Using Part-of-Speech Information for Transfer in Text Classification

Jason D. M. Rennie
jrennie@csail.mit.edu

December 17, 2003

Abstract

Consider the problem of text classification where there are very few training examples for the target task. An idea popularized recently is to use unsupervised examples to gain a better understanding of the example space. This can, in turn, be used to improve classification performance on the target task. The problem of transfer adds a new twist. It supposes that there is available large amounts of supervised data, pertaining to a set of "reference" tasks. The interesting case is where these reference tasks are only loosely related to the main task, such as the tasks all being topic classification problems. We propose to use part-of-speech as a vehicle for transfering information from the reference tasks to the target task. Different types of classification problems use parts-of-speech differently. We give two algorithms. The first treats the regularization term of a classifier as a Gaussian prior and learns a covariance matrix from the reference data. The second treats the supervision for the reference data as being cluster labels; spectral clustering is used to learn a kernel matrix from the reference data. We flesh-out in full the first idea and discuss many issues pertaining to Regularized Least Squares, the classification algorithm we use. We present experiments that give negative results, suggesting that the Gaussian prior idea may not inappropriate for the part-of-speech/transfer problem.

1 Utilizing Part-of-Speech for Transfer

The idea of using unsupervised data for classification has received much attention of late (e.g. [1, 2]). However, often it is easy to obtain large quantities of labeled, or clustered data. The difficulty is that this data may not be labeled in a way that directly pertains to the task at hand. This is the problem that we call transfer. The "target" task is the classification task that we would like to solve. We posit that there exist many other classification tasks (call them "reference" tasks) that share some similarity to the target task; we assume that labeled data for these reference tasks is freely available. This problem of transfer is interesting when the connection between the reference tasks and the target task is loose. In this case, features that are useful for the reference task may be useless for the target task. We propose that there may still be a way to reduce error on the target task via transfer.

We think that different types of text classification problems use parts-of-speech in different ways. For example, when discriminating between topics, nouns may be the most useful part-of-speech; when classifying authors, articles and determiners may be more important. We propose that if there is variation in the way different parts-of-speech are used, then we can take advantage of this. First, we consider a motivating example. In Table 1, we give the top words for each of the categories in the 20 Newsgroups data set. We tagged all words in the data set with their part-of-speech (POS) using Ratnaparki's tagger and then trained a one-versus-all Regularized Least Squares Classifier (RLSC). The words listed had the largest weights for the corresponding binary classifier. The table includes 55 nouns, 4 adjectives (one of which may have been mis-labeled, "keith_jj") and one verb. 20 Newsgroups is a topic classification task. Each of 20 newsgroups represents a topic and the classifier's job is to assign an e-mail to the newsgroup to which it was posted. From this list of top words, it seems that nouns are highly useful in identifying topics in documents. Of course, it is possible that we are simply observing the distribution of different parts-of-speech. There are more unique nouns than other part-of-speech classes. But, if we were simply observing the distribution of unique parts-of-speech found in the data set, we would expect to see 38 nouns, 9 verbs, 8 adjectives, 2 cardinal numbers, 1 adverb and 2 other parts of speech[1]. So, there is certainly some way in which nouns (and possibly adjectives) are in some way more useful for 20 Newsgroups than other parts of speech. The question is, how do we take advantage of this information?

2 Regularized Least Squares Classification

Before we discuss the problem of using part-of-speech information for transfer, we introduce the classification algorithm that we plan to use for this work. Many algorithms have been used for the problem of text classification, including Naive Bayes [3], Decision Trees [4], k-Nearest Neighbor [5], rule learning [6] and Neural Networks [7]. However, there is a class of algorithms, here called regularized linear classifiers, which have performed as well or better than all other algorithms for the problem of text classification. Regularized linear classifiers can be written as the sum of two terms: loss and regularization. The loss term is a function of how well the model fits the data; the regularization term penalizes more complex models. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of training examples; let $\{y_1, \ldots, y_n\}$ be their binary ($\{+1, -1\}$) labels. Let $\mathbf{w}$ define the decision boundary. Using the squared L2-norm for regularization (as is common), a regularized

[1]We counted 132930 unique tokens: 84349 nouns, 19020 verbs, 18545 adjectives, 3659 cardinal numbers, 2482 adverbs and 4917 other parts of speech

Newsgroup	1st	2nd	3rd
alt.atheism	benedikt_nnp	tammy_nnp	keith_jj
comp.graphics	images_nns	Dd_jj	animation_nn
comp.os.ms-windows.misc	windows_nns	windows_nnp	ftp.cica..._nn
comp.sys.ibm.pc.hardware	gateway_nnp	orchid_nnp	cmos_nnp
comp.sys.mac.hardware	apple_nnp	mac_nnp	powerbook_nn
comp.windows.x	motif_nnp	widget_nn	xterm_nn
misc.forsale	sell_vb	offers_nns	sale_nn
rec.autos	car_nn	cars_nns	driving_nn
rec.motorcycles	dod_nnp	bike_nn	motorcycle_nn
rec.sport.baseball	baseball_nn	phillies_nns	stadium_nn
rec.sport.hockey	hockey_nn	hockey_nnp	nhl_nn
sci.crypt	cryptography_nn	nsa_nnp	encryption_nn
sci.electronics	circuit_nn	voltage_nn	electronics_nns
sci.med	doctor_nn	disease_nn	symptoms_nns
sci.space	space_nnp	launch_nn	orbit_nn
soc.religion.christian	clh_nn	christ_nnp	scripture_nn
talk.politics.guns	guns_nns	gun_nn	firearms_nns
talk.politics.mideast	israeli_jj	armenians_nnps	turkish_jj
talk.politics.misc	deficit_nn	phill_nnp	teel_nnp
talk.religion.misc	koresh_nnp	hudson_nnp	decenso_nnp

Table 1: For each category in the 20 Newsgroups data set, the words with the three largest weights are listed. The underscore and suffix is not part of the word; it indicates the word's part-of-speech. Benedikt, Tammy and Keith are names of common alt.atheism posters. "DoD" stands for Denziens of Doom, the name of a motorcycle group. "clh" is the signature of the moderator of the soc.religion.christian newsgroup. Phill and Teel are names of common talk.politics.misc posters. Hudson and Decenso are names of common talk.religion.misc posters.

linear classifier minimizes

$$\sum_i L(y_i \mathbf{x}_i^T \mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}. \tag{1}$$

$L(\cdot)$ is the loss function. The regularization parameter, λ, weighs the importance of the compexity penalty. The Support Vector Machine (SVM), Regularized Least Squares Classification (RLSC) and Regularized Logistic Regression (RLR) all use this form. For the SVM, $L(z) = (1-z)_+$. For RLSC, $L(z) = (1-z)^2$. For RLR, $L(z) = (1 + e^{-z})^{-1}$. [8] gives a discussion of these and other classifiers that fit this form. Such classifiers have come to dominate recent empirical evaluations of text classification algorithms [9, 10, 8, 11]. Regularized Least Squares Classification (RLSC) [12], also known as Ridge Regression [13], has the advantage of being especially easy to optimize since its objective is quadratic. Results [12, 8, 11] show that RLSC performs comparably and sometimes slightly better than other regularized linear classifiers.

2.1 Optimizing RLSC

The first step in making use of RLSC is optimizing the parameter vector that defines the decision boundary, $\mathbf{w}$. Much of the following discussion can be found in [12]. We procede in the usual fashion, taking the gradient of the objective and setting it equal to zero. The manipulations are simplest in matrix-vector form. Let $X = \begin{bmatrix} -\mathbf{x}_1- \\ \vdots \\ -\mathbf{x}_n- \end{bmatrix}$; let $\mathbf{y} = [y_1 \cdots y_n]^T$. The rewritten objective is

$$\begin{aligned} J &= (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}, \qquad (2) \\ &= \mathbf{w}^T X^T X \mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y} + \lambda \mathbf{w}^T \mathbf{w}. \qquad (3) \end{aligned}$$

The gradient is

$$\frac{\partial J}{\partial \mathbf{w}} = 2X^T X \mathbf{w} - 2X^T \mathbf{y} + 2\lambda \mathbf{w}. \tag{4}$$

Setting this to the zero vector, we get

$$(X^T X + \lambda I)\mathbf{w} = X^T \mathbf{y}. \tag{5}$$

If the number of dimensions is small, we can simply invert $(X^T X + \lambda I)$ and left multiply on both sides to get the solution for $\mathbf{w}$. For larger problems, iterative, gradient-descent algorithms can be used. We used Conjugate Gradients [14], which requires that new search directions are perpendicular to ones used in previous iterations. It has the advantage of being guaranteed to converge in no more than d iterations, where d is the number of dimensions. Also, in practice it converges much faster than simpler, gradient-based approaches.

A useful insight made by [12] is that RLSC can be kernelized, much like the SVM and other algorithms. That is, instead of representing the classifier in terms of parameters of the decision boundary, we can represent it in terms of weights on the examples. Let $\mathbf{c}$ be a n-length vector, where n is the number of training examples. Consider the RLSC solution (equation 5). Making the substitution $\mathbf{w} = X^T\mathbf{c}$, we get

$$X^T X X^T \mathbf{c} + \lambda X^T \mathbf{c} = X^T \mathbf{y}. \tag{6}$$

The Representer Theorem (Appendix B in [12]) shows that this substitution does not reduce the expressiveness of the classifier. Note that removing the X^T from the left of each term does not affect the validity of the solution,

$$X X^T \mathbf{c} + \lambda \mathbf{c} = \mathbf{y}. \tag{7}$$

Note that XX^T is a matrix of dot-products between example vectors. In other words, $(XX^T)_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$. Here we show how RLSC can solve for non-linear decision boundaries without explicitly representing the higher-dimensional data points. Consider projecting the data to a higher-dimesnsional space. Let $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ be such a projection. Let's rename XX^T to K and replace each entry with $K_{ij} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Now, solving

$$(K + \lambda I)\mathbf{c} = \mathbf{y} \tag{8}$$

gives us the RLSC classifier. We use $K = XX^T$ to find a linear decision boundary and compute dot-products in a projected space to find a linear boundary in that projected space (and a non-linear boundary in the original space). Note that the classification output for a new example is the dot-products of all the training examples with the new example, weighted by $\mathbf{c}$. To reconcile notation with common practice, we write $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Then, the classification output for a new example, $\mathbf{x}$, is

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i K(\mathbf{x}_i, \mathbf{x}). \tag{9}$$

We noted before the Conjugate Gradients is guaranteed to converge in at most d iterations. By showing that the classifier can be represented in terms of weights on the examples, we have shown that Conjugate Gradients is also guaranteed to converge in at most n iterations. In other words, it will converge in $\leq \min(n, d)$ iterations.

2.2 Selecting the Regularization Parameter

We have discussed how to learn parameters for the RLSC decision boundary. But, there is one parameter we have yet to discuss: the regularization parameter, λ. Unlike the decision boundary parameters, we cannot simply minimize the objective since we would always choose $\lambda = 0$, which is rarely the best choice. A popular choice to determine the regularization parameter is cross-validation. The training set is broken up into N sets. Labels for each set are given by a classifier trained on the remaining sets. Cross-validation gives an approximation of the generalization error. A larger N leads to a better approximation. When $N = n$ (one set per training example), this becomes leave-one-out cross-validation (LOOCV), where each training example is labeled by a classifier trained on all of the other training examples. An advantage of RLSC is that the LOOCV error can be calculated analytically. [12] shows how this can be done, but there is a mistake in the formula. We give the corrected formula and proof, along with additional intuition.

We show that we can determine the LOOCV classification outputs for all training examples with a single matrix inverse as our most expensive operation. Let $f_i(\mathbf{x})$ be the classification output for $\mathbf{x}$ with the classifier trained on all examples but $\mathbf{x}_i$. Define $G = (K + \lambda I)^{-1}$. We will show that

$$f_i(\mathbf{x}_i) = \frac{f(\mathbf{x}_i) - (KG)_{ii} y_i}{1 - (KG)_{ii}} \tag{10}$$

It is worth taking a moment to explain what this means. Recall that the solution to RLSC is the vector $\mathbf{c}$ that solves $(K + \lambda I)\mathbf{c} = \mathbf{y}$. So, $\mathbf{c} = G\mathbf{y}$. The fact that we can calculate LOOCV outputs with a single matrix inverse stems from the fact RLSC can be solved with a single matrix inverse. $K\mathbf{c} = KG\mathbf{y}$ and $K\mathbf{c}$ is the vector of classification outputs (the i^{th} entry of $K\mathbf{c}$ is $f(\mathbf{x}_i)$). The output for the i^{th} example is computed via dot-product of the i^{th} row of KG with the column vector $\mathbf{y}$. $(KG)_{ii} y_i$ is the component of that dot-product that comes from the i^{th} example. So, one view is that the LOOCV calculation takes the output from the fully trained classifier, removes the part due to the i^{th} example, then renormalizes to get the LOOCV output. Next, we give the proof of the LOOCV formula.

Define $\mathbf{Y}^i$ to be vector where $Y^i_j = y_j$ for $j \neq i$ and $Y^i_i = f_i(\mathbf{x}_i)$. Since $f_i(\cdot)$ is the RLSC classifier trained on all examples except the i^{th}, it minimizes

$$\sum_{j \neq i} (Y^i_j - f_i(\mathbf{x}_j))^2 + \lambda \mathbf{c}^T K \mathbf{c}. \tag{11}$$

Since $Y^i_i = f_i(\mathbf{x}_i)$ (by definition), it also minimizes

$$\sum_{i=1}^{n} (Y^i_j - f_i(\mathbf{x}_j))^2 + \lambda \mathbf{c}^T K \mathbf{c}, \tag{12}$$

where the sum is now over all training examples. Thus, f_i is the solution to a modified RLSC problem where the set of training example labels is $\mathbf{Y}^i$. In other words, $K\mathbf{c} = KG\mathbf{Y}^i$, or

$$\begin{aligned} f_i(\mathbf{x}_i) &= \sum_j (KG)_{ij} \mathbf{Y}^i_j, & (13)\\ &= f(\mathbf{x}_i) - (KG)_{ii} y_i + (KG)_{ii} f_i(\mathbf{x}_i), & (14)\\ &= \frac{f(\mathbf{x}_i) - (KG)_{ii} y_i}{1 - (KG)_{ii}}, & (15) \end{aligned}$$

as we wanted to show.

Being able to calculate the LOOCV error with a single matrix inverse can be a significant computational advantage. For moderate size problems, where taking the inverse is not too burdensome, we can train a classifier on the full set of training examples *and* calculate the LOOCV error with a single matrix inverse.

2.3 Weakness of LOOCV Bounds

The calculation of LOOCV error becomes difficult for large problems (such as when the number of training examples numbers in the tens of thousands). [15] proved bounds on the LOOCV error for a certain class of kernel classifiers. [12] extended those to RLSC. We applied these bounds to a text classification problem with the intent of using them to select the regularization parameter [16]. We found that the bound was accurate for large values of the regularization parameter, but quickly decayed as we decreased λ, greatly over-estimating the error for smaller values. As a result, the bound always selected a regularization parameter that was too large. We did not see how the bound could be used as part of a method for selecting a reasonable regularization parameter. It seems that until better bounds or approximations are put forward, it is still necessary to use cross-validation for the determination of the regularization parameter on large problems.

3 Making use of POS Information

Now we return to the problem of using part-of-speech information for transfer. As we have discussed, we believe that different types of text classification problems may use parts-of-speech in different ways. Nouns may be more useful for topic classification while determiners may be more useful for author identification. This section addresses the question, *how do we transfer knowledge from the reference tasks to the target task?* We discuss two ideas. The first interprets the regularization term for a regularized linear classifier as a Gaussian prior and learns a diagonal covariance matrix based on classifiers trained on reference task data. We show that this is equivalent to a certain pre-processing of the data. In the next section, we give experimental results. The second idea involves a direct modification of the kernel. We provide an outline of how we might modify the kernel and give pointers to relevant publications.

3.1 Learning a Regularization Prior

The regularization term on a regularized linear classifier can be interpreted as the negative log-likelihood of a zero-mean Gaussian. The analogy is particularly compelling for Regularized Logistic Regresion [17]. We modify the objective by inserting the inverse covariance matrix, Λ^{-1}, into the regularization term. The revised RLSC objective is

$$J = (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \Lambda^{-1} \mathbf{w}. \tag{16}$$

The standard form of RLSC can be seen as using an identity inverse covariance matrix.

We propose to use data from reference tasks to learn entries in the inverse covariance matrix. The logic is as follows. Let $f : \mathbb{R}^{n \times d} \times \mathbb{R}^n \rightarrow \mathbb{R}^d$ be a classification algorithm that learns a linear decision boundary. Given a set of training data, examples X and binary $\mathbf{y}$, it returns a vector of decision boundary weights, $\mathbf{w}$. Let $p : \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R}$ specify a distribution on examples and labels. We assume that each classification task has a unique p of its own; we also assume that all possible classification tasks can be indexed, $1, \dots, N$. Let p_i be the example/label distribution for the i^{th} classification task. Let $-i$ represent the i^{th} task with its labels reversed. Finally, let $q : \mathbb{Z} \rightarrow \mathbb{R}$ specify a distribution on tasks. We assume that $q(i) = q(-i)$. The examples for a task are drawn by

- selecting n, the number of examples to be drawn,
- drawing a task, i, from the task distribution, q, and
- drawing n examples/labels from the distribution for task i, p_i.

The classification function, f, maps a set of examples to a weight vector. Hence, the above model gives us a joint distribution for weights over all possible features. Note that the distribution is centered—the mean is $\mathbf{0}$ (due to our $q(i) = q(-i)$ assumption). We model the distribution on weights as a Gaussian. Since our mean is zero, the covariance matrix specifies the distribution.

We now have a model for classification weights. Furthermore, the reference data can provide us with samples from this distribution. We are given sets of examples and their labels. We can apply the classifier to get out weight vectors. We treat each weight vector as a sample from the weight vector distribution. This would work great... if only we had enough samples. It is unlikely that we would be able to amass sufficiently many labeled data sets to reasonaly estimate a joint (or even marginal!) distribution across the full set of features. Note that the weight distribution for a feature can be very peaked. It is not uncommon for a feature to have very small weights for most tasks and large weights for a handful of tasks. This is where the part-of-speech (POS) information comes in. We can use the POS information to usefully group together sets of features. While there is only one feature for the word "baseball", there may be thousands of unique nouns. So, with only a few data sets, we can quickly gather tens- or hundreds-of-thousands of weight samples for nouns. In lieu of trying to learn a full covariance matrix, or even the full diagonal of a covariance matrix, we learn a diagonal covariance matrix where words that are of the same POS are constrained to have the same value. To estimate the covariance for a POS class, we average squared weights over all words of that POS and all reference tasks.

In the next section, we show that using a non-identity covariance matrix is equivalent to pre-processing the input data. Later we give experimental results using this technique.

3.2 Equivalence of Regularization Prior to Example Pre-Processing

To gain a better understanding of how the addition of the inverse covariance matrix affects a regularized linear classifier, we compare it to another modification, pre-processing of the example vectors. We consider the general form of regularized linear classifiers. Recall that the objective is

$$J = \sum_{i=1}^{n} L(y_i \mathbf{x}^T \mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}. \tag{17}$$

Interpreting the regularization term as a Gaussian prior and inserting a non-identity covariance matrix, we get

$$J = \sum_{i=1}^{n} L(y_i \mathbf{x}^T \mathbf{w}) + \lambda \mathbf{w}^T \Lambda^{-1} \mathbf{w}. \tag{18}$$

Note that since Λ is a covariance matrix, it is positive definite, which means that it's and it's inverse's square roots exist. Let $L = \Lambda^{1/2}$. Consider the substitution $\mathbf{w} = \Lambda^{1/2}\mathbf{v}$. The objective becomes

$$J = \sum_{i=1}^{n} L(y_i \mathbf{x}^T \Lambda^{1/2} \mathbf{v}) + \lambda \mathbf{v}^T \mathbf{v}. \tag{19}$$

So, introducing a non-identity covariance matrix is equivalent to using $\mathbf{x}' = \mathbf{x}^T \Lambda^{1/2}$ as a data pre-processing step.

3.3 Modifying the Kernel Matrix

Viewed in the light of the regularization term representing a Gaussian prior on the weight vector, the insertion of the inverse covariance matrix seems sensible. However, from a kernel perspective it is quite odd. In this section, we discuss the idea of directly modifying the kernel matrix. First, we show that a non-identity covariance matrix is *not* in any way equivalent to a modification to the kernel matrix. Then, we discuss some related spectral clustering work which may be used to transfer part-of-speech information from reference tasks to target task by learning a special kernel matrix. This idea is much less developed, so we only touch on the high points and do not get to mathematical details.

To show that the Gaussian prior interpretation is not equivalent to a kernel modification, we return to the kernel version of RLSC. We make the substitutions $\mathbf{w} = X^T \mathbf{c}$ and $K = XX^T$. The objective is

$$J = \mathbf{c}^T K K \mathbf{c} - 2\mathbf{c}^T K \mathbf{y} + \mathbf{y}^T \mathbf{y} + \lambda \mathbf{c}^T K \mathbf{c}. \tag{20}$$

Taking the Gaussian prior interpretation and adding an inverse covariance matrix would involve modifying the kernel matrix, K, in the last term only. Pre-processing the input data as described above would inolve modifying the kernel matrix, K, in only the first two terms. Neither change is equivalent to making a substitution for K.

We now know that a kernel matrix modification is different from the other ideas we have discussed. A modification to the kernel matrix effectively projects the data into an alternate space. The classifier then learns a decision boundary in that alternate space. What we need is a way to learn a kernel matrix given a set of reference task problems. Whereas with the covariance matrix, it was sensible to simply build classifiers on the reference task data and average weight values, there is no clear justification for doing something similar with the kernel matrix.

We consider spectral clustering as a way to modify the kernel matrix to transfer POS information from the reference task(s). Spectral clustering considers the problem of clustering together data points based on their eigenspace. Given an affinity (similarity) matrix, K, it associates the points into clusters so as to minimize the similarities between points in different clusters. This is the traditional framing of the problem. Recently, [18], considered the reverse problem. Given a clustering, can one learn a kernel matrix that minimizes the similarities between points in different clusters. Though this may seem strange, it is very similar to the problem of classification; most classification algorithms use different loss functions than the normalized cut objective that is often used for spectral clustering. But, the input is identical—examples grouped into clusters or classes. Hence, our reference data can serve as input to such an algorithm.

What remains to be discussed, and, what we have yet to figure out, is how to incorporate part-of-speech information. Similar to the reasoning behind not learning a full covariance matrix, it may be overkill to try to learn an unconstrained kernel matrix over the reference task data. Whereas in other types of applications, individual features may usefully transfer from one problem to another, in text classification, the words that are useful in one task are rarely useful in another (unless there is topic overlap, which we see as a less interesting problem). [18] considers a kernel matrix of the form

$$k_\alpha(x, y) = \exp(-(x-y)^T \mathrm{diag}(\alpha)(x-y)), \tag{21}$$

where x and y are data points and α serves as weights on the feature values. In learning the kernel, we might restrict entries of α of the same parts-of-speech to be equal to each other.

4 Experiments

We ran experiments to evaluate the idea of interpreting the regularization term as a Gaussian prior and learning a covariance matrix. The results are poor. We were not able to reduce error on the target task, even when we tried exponentiating the weights trained on reference task classifiers and using a very small training set for the target task.

To evaluate our idea, we ran text classification experiments on the 20 Newsgroups data set. 20 Newsgroups (as the title suggests) is a collection of newsgroup posts from 20 different newsgroups from the mid 1990s. There are approximately 1000 posts per newsgroup. We use the "20news-bydate" version, which has dupliates removed, posts are sorted within each newsgroup by date into train/test sets and newsgroup-identifying headers are discarded. Before passing the documents to Adwait Ratnaparkhi's POS tagger[2], we (1) removed lines of UUEncoded text, (2) removed quote characters from the beginning of lines, and (3) removed periods and at-signs from e-mail addresses (as they tended to confuse the tagger). We then (4) ran Ratnaparkhi's tagger and further processed output from the tagger. We (5) lower-cased all alphabetic characters, (3) discarded tokens of length 25 or greater, (4) converted all digits to 'D', and (5) removed non-alphanumeric beginning and ending characters. We computed a document vector for each post, consisting of the number of times each word occured in that post. We transformed each count by $\log(x+1)$ and then normalized each vector to (L2-norm) length 1. We did not stem or use a stop list.

To create a problem suitable for transfer, we grouped 19 of the newsgroups into 5 distinct sets, religion, computers, recreation, science and politics. For each set, we took as the target task the problem of classifying posts within that set. For example, the politics set includes talk.politics.misc, talk.politics.guns and talk.politics.mideast. The politics target task was to assign a post to one of the three politics newsgroups. We used the remaining newsgroups as reference data. This division ensured that the target topics were sufficiently different from the reference topics; training examples from the reference tasks would be of little or no direct use for the target task (though we did not test this, e.g., by using the output of the reference task classifiers as input for the target task classfier). We used 10 training examples per class for the target task.

We first used the reference data to select a regularization parameter, λ. For each of 10 rounds, we chose 10 training examples/class from the reference data and computed LOOCV error (using the formula discussed in section 2.2. We chose the regularization parameter with the lowest average LOOCV error over those trials. Common choices were $\lambda = 10^{-2}$ and $\lambda = 10^{-3}$. We took the squared weights learned for the selected value of λ, rescaled them so that the average value was 1 and then averaged within POS groupings. We used a diagonal inverse covariance matrix and set the entry for each feature to $1/v^p$ where v is the average weight-square value for the feature's POS. We used p to scale the entries of the inverse covariance matrix.

We found that the performance was quite poor. We never saw a significant improvement compared to using the identity covariance matrix. The results are given in Table 2. The reported experiments used the full set of possible POS tags. We give the square-weight values that we learned from the reference data in Table 3. These are somewhat surprising. Many parts-of-speech that are not usually associated with topic classification problems, such as wh-pronouns, possessive pronouns and determiners are given very large weights. This may be due to the fact that the 20 Newsgroup problem is somewhat of an author

[2]Available from `http://www.cis.upenn.edu/∼adwait/statnlp.html`.

	religion	computers	recreation	science	politics
$p = 0$	0.5371	0.6389	0.4534	0.5108	0.4796
$p = 0.25$	0.5374	0.6392	0.4535	0.5108	0.4796
$p = 0.5$	0.5371	0.6393	0.4550	0.5109	0.4796
$p = 1$	0.5369	0.6393	0.4603	0.5109	0.4795
$p = 2$	0.5336	0.6430	0.4796	0.5113	0.4794

Table 2: Shown are averaged classification errors for the target task. The high errors are due to the fact that the classes are closely related and we use only 10 training examples per class. p indicates to what power covariance estimates from the reference data set were taken. $p = 0$ corresponds to the identity covaraince matrix, which is equivalent to the standard RLSC formulation. No setting of p yields any significant improvement over the baseline. Performance quickly degraded when we tried larger values of p (not shown).

identification task, since each newsgroup has its own style of writing. It also suggests that there may yet be value in the idea of learning a covariance matrix, but that we may need to group parts-of-speech to see a benefit. For a POS such as WP$ (whose), our attempts to learn a covariance appear to be fruitless because they vary so much across different training sets. Whereas personal pronouns (PRP) are given weights within a narrow range, so the covariance learned on the reference tasks is more likely to transfer to the target task. Unfortunately, we did not have time to run additional experiments, but we plan to experiment with the idea of restricting the covariance estimation to groups of POS which show less weight variation between tasks.

5 Summary

We have described a method for transfering part-of-speech information from a set of reference tasks to a target task. Experimentally we found it to perform poorly. But, an analysis of the weights learned on reference tasks indicates that there is room for improvement; some POS classes show consistency in learned weights across reference tasks. We may find that the technique works better when we group parts-of-speech based on weight variability. We are also very interested in the idea of learning a kernel matrix for the target task based on reference task data. This may be a more natural approach since it directly modifies the space in which the examples lie and it scaling should not be an issue like it is for the Gaussian prior idea. We would also like to consider other natural language information, such as sentence structure.

References

[1] Martin Szummer and Tommi Jaakkola. Information regularization with partially labeled data. In *Neural Information Processing Systems 15*, 2003.

POS	religion	computers	recreation	science	politics
cc	15.68	20.18	22.91	15.04	15.08
cd	3.69	3.78	3.00	3.44	3.46
dt	27.00	31.61	27.94	30.43	29.83
ex	0.07	18.26	1.69	0.14	3.61
fw	0.14	0.05	0.13	0.06	0.23
in	16.48	17.16	16.94	16.63	14.54
jj	0.13	0.14	0.13	0.16	0.20
jjr	0.36	0.28	0.15	0.19	0.23
jjs	0.27	0.24	0.33	0.20	0.24
md	6.02	5.24	3.12	5.13	4.85
nn	0.38	0.26	0.29	0.34	0.43
nnp	0.28	0.18	0.32	0.26	0.31
nnps	0.13	0.06	0.07	0.07	0.12
nns	0.45	0.25	0.29	0.23	0.30
pdt	3.55	5.96	0.48	3.06	3.88
pos	6.10	5.53	4.70	7.26	8.68
prp	78.30	74.75	79.09	75.71	72.12
prp$	47.10	110.09	49.76	29.25	49.37
rb	0.82	0.93	0.54	0.83	0.85
rbr	0.18	0.46	0.72	1.39	0.23
rbs	1.40	0.10	0.10	0.57	0.13
rp	2.49	1.31	1.72	2.52	3.07
to	18.63	21.05	21.98	21.46	18.01
uh	0.09	0.11	0.09	0.05	0.09
vb	0.47	0.33	0.35	0.41	0.37
vbd	0.27	0.22	0.18	0.21	0.17
vbg	0.30	0.23	0.27	0.25	0.29
vbn	0.22	0.13	0.12	0.18	0.14
vbp	0.87	0.77	1.07	0.88	0.61
vbz	1.17	2.13	1.89	1.66	1.56
wdt	2.14	0.57	0.17	2.27	3.64
wp	19.54	33.46	28.16	14.68	8.72
wp$	1.30	0.02	4.05	1.17	2.31
wrb	8.65	13.13	7.55	13.21	10.03

Table 3: Shown are average squared weights for different parts-of-speech trained on the reference task data. Many large weights are found on parts-of-speech that are not usually associated with topic classification tasks, such as wh-pronouns (WP), possessive pronouns (PRP$) and determiners (DT). Note that these have been rescaled so that the average feature (not POS) has a squared weight of 1.

[2] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning: Proceedings of the Twentieth International Conference*, 2003.

[3] Jason D. M. Rennie. Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology, 2001.

[4] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[5] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[6] William W. Cohen. Fast, effective rule induction. In *Procedings of the Twelfth International Machine Learning Conference (ICML-95)*, 1995.

[7] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR-95)*, 1995.

[8] Fan Li and Yiming Yang. A loss function analysis for classification methods in text categorization. In *Proceedings of the Twentith International Conference on Machine Learning*, pages 472–479, 2003.

[9] Jason D. M. Rennie and Ryan Rifkin. Improving multiclass text classification with the Support Vector Machine. Technical Report AIM-2001-026, Massachusetts Insititute of Technology, Artificial Intelligence Laboratory, 2001.

[10] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

[11] Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.

[12] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning.* PhD thesis, Massachusetts Institute of Technology, 2002.

[13] A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[14] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/∼jrs/jrspapers.html, 1994.

[15] Tommi Jaakkola and David Haussler. Probabilistic kernel regression models. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1998.

[16] Jason D. M. Rennie. On the value of leave-one-out cross validation bounds. http://www.ai.mit.edu/∼jrennie/writing/loocv.ps.gz, December 2003.

[17] Jason D. M. Rennie. On L2-norm regularization and the Gaussian prior. http://www.ai.mit.edu/∼jrennie/writing/l2gaussian.ps.gz, May 2003.

[18] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2004.