Semantic classification of email

Data

The data is comprised of a pool of 1 million emails from the Enron Corporation that were made available to the public. From these, N=1805 emails were tagged with semantic labels. The labels are in the domain of meetings.
Each email can have multiple labels; this makes the classification problem more interesting, the idea is to try to retrieve correctly all the labels assigned to a particular email. Most of the emails transcribed are labeled **else,** leaving relatively few classes with enough training samples to produce significant results**.**

add-agenda	add-agenda-items	add-attendees	attend	cancel-active	cancel-passive	**else**	organize	prepare-materials	remove-attendees	request-info	reschedule
2	4	158	326	2	30	1419	16	5	14	4	9

Table 1. Number of emails containing each label.

I left out the classes with less than 9 training examples: add-agenda, add-agenda-items, cancel-active, prepare-materials and request-info.
The data was split into train and test sets with 70% and 30% of the data respectively. Since there are some classes with few positive samples I used 3-fold cross validation on the training data to tune parameters.

Feature Extraction

The first step is to extract relevant features from the 1805 emails. The header of the emails was removed, except for the **Subject:** line which contains relevant information about that can help in the classification problem. The rest of the header contains information like "**From:"** that might improve training error but will not generalize to a different test set.
The training feature is the frequency of the word in the email normalized by the total number of words in the email. A dictionary of features is created on the training data; this dictionary is used in both training and testing, OOVs are ignored.

Pre-processing
The body of the email was pre-processed to parse strings for dates, times numbers and phone numbers into generic tags, <date>, <time>, <month>, <year>, <number>, and <extension>. All capitalization and punctuation was also removed.

Training

I trained 7 SVM classifiers, one for each label. The classifier used was SVM light [1].

SVM regularization parameter **C** was tuned (using 3-fold cross validation) on the training set, individually for each class, on each new training run, as well as the threshold **T**. The weight on the errors of positive samples **j** was set for each class individually.

To tune **C** I retrained the models on 66% of the training data and tested on the remaining 33% for each of the three folds, and chose the **C** that maximized the average F-score of the three folds.
SVM light computes a good default value for $C_d = 1/(avg\ x * x)$**,** I use a multiplier **m** (500, 200, 100, 50, 10, 1, 0.1, 0.01) on this default value. After that I fix **C=m*Cd** and tune **T** by computing the F-score on **score -T** for **T** from +5 to -5 in increments of 0.1.

In the data set the classes are unbalanced; there are many more negative examples than positive examples. SVM light provides a parameter **j** that multiplies the sum of the errors of the positive examples in the objective function [2].

$$\min_{y,a,b} \ \frac{1}{2}\|w\|^2 + C\left[\sum_{i:y_i=-1}\varepsilon_i + j\sum_{i:y_i=+1}\varepsilon_i\right]$$
$$\text{s.t. } y_i\left(w^T x_i + b\right) \geq 1-\varepsilon_i \ , \ \forall i$$
$$\varepsilon_i \geq 0, \ \forall i$$

The **j** parameter can be set to $j = \frac{N_-}{N_+}$ where N_- is the number of negative samples.

Testing

There are two metrics that I used to evaluate the performance of the system.

1. The 1-Best F-score (harmonic mean of the precision and recall).
2. The N-Best F-score.

For 1-Best I count as true/false positives/negatives the highest scoring class (i.e.: I sort the classes in decreasing order according to the SVM score and choose the first class). For the N-Best F-score I count all the positive scoring or positive label classes. For a given email TP will be the sum of the classes that have positive labels and positive SVM score, FP the sum of the classes that have negative labels and positive scores, and FN the sum of the classes that have positive labels and negative scores.
The N-Best score will give an idea of how many correct labels we are assigning to each email, whereas the 1-Best score will only look at the highest scoring class and can be misleading regarding the performance of the system, since we are interested in retrieving all the labels for a given email.

Experimental Results

I will consider the baseline to be the case in which we classify all emails as **else** (i.e.: the email doesn't contain any meeting information). For this case I set all the samples as **else** in the test data (score +10 for **else** and -10 for the other classes) and compute the scores.

1-Best:	TP: 426 ,FP: 150	Precision:	73.9 %	F-score: **84.9**
	FN: 0 ,TN: 0	Recall:	100 %	
N Best:	TP: 426 , FP: 150	Precision:	73.9 %	F-score: **68.4**
	FN: 241 ,TN: 6095	Recall:	63.8 %	

If I train the SVM without tuning **C** or **j** and I don't tune **T** I get:

1 Best:	TP: 453 FP: 99	Precision:	82.0 %	F-score: 89.3
	FN: 8 TN: 16	Recall:	98.2 %	
N Best:	TP: 460 FP: 100	Precision:	82.1 %	F-score: 74.9
	FN: 207 TN: 6145	Recall:	68.9 %	

In the following I'll show only the F-score measure.
After optimizing the threshold **T** for each class:
1-Best : 92.8 %
N-Best: 67.2 %

The N-Best score degrades by 7.7 % absolute, this seems counter intuitive. After examining the results I see that the class **reschedule** is producing a lot of FP with low positive score. This class has only 9 positive samples so it's the most unbalanced class, so tuning **j** would probably fix this problem.

After re-training the models using **j** as described above for each class and re-tuning **T:**
1-Best: 93.9
N-Best: 86.9

We get a nice improvement of 1.1% absolute for the 1-Best score and 19.7% absolute for the N-Best score. This seems to fix the problem with the reschedule class.

Tuning **C** (as **m*** **Cd**), re-training the models and re-tuning **T** afterwards I get:
1-Best: 93.8
N-Best: 86.2

Tuning **C** doesn't seem to improve the score.

After doing some error analysis I observed that there are cases in which else and other labels are selected as positive results, this is not logical since the class should be either else or any other class. Also there can be cases in which a sample doesn't get any positive scores and is considered a TN so it doesn't belong to any class, but it must belong to a class because **else** plus the other classes is the universe.

Changing the approach to train SVMs only for the classes that are not else, and consider an email to be **else** if all the scores are negative, that is if all the SVMs classified the sample as negative. This solves the logic problems in the experimental setup.

After re-tuning **C** and **T** I get:
1-Best: 94.2
N-Best: 86.5

I get a small improvement of 0.3% absolute for both scores.

For feature selection I tried removing features in the dictionary that have frequency 1. These features could be considered noise, and removing them can decrease the feature dimension significantly. After re-tuning parameters and re-training I get:
1-Best: 93.6
N-Best: 86.8

I observe a small improvement of 0.3% for the N-Best score and degradation of 0.6% for the 1-Best score.

One idea that improves performance considerably in Speaker ID systems that use SVM classifiers is to do variance normalization on the features. I computed the standard deviation of the features on the training data and normalized both the train and test features. It doesn't seem to improve performance.
1-Best: 92.9
N-Best: 86.3

System	1-Best F-score	N-Best F-score
baseline (all else)	84.9	68.4
default C, j=1, T=0	89.3	74.9
default C, j=1, tune T	92.8	67.2
default C, tune j, T	93.9	86.9
tune C, j, T	93.8	86.2
remove else SVM, tune C,j,T	94.2	86.5
remove freq. 1 words	93.6	86.8
variance normalization	92.9	86.3

Table 2. Systems performance

Conclusions

An email classification system was developed that tries to predict a semantic label in the context of meetings. The system improves on the performance of a baseline that classifies all emails as irrelevant (else), based on a metric (N-Best F-score) that was defined to asses performance in the case that multiple labels (with the same weight) are assigned to a sample. The biggest improvements in performance are obtained by properly tuning parameters **j** and the thresholds **T.**

References

[1] Thorsten Joachims, *Learning to Classify Text Using Support Vector Machines*. Dissertation, Kluwer, 2002.

[2] K. Morik, P. Brockhausen, and T. Joachims, Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. International Conference on Machine Learning (ICML), 1999.