Combining Probability-Based Rankers for Action-Item Detection

Paul N. Bennett
Microsoft Research[*]
One Microsoft Way
Redmond, WA 98052
paul.n.bennett@microsoft.com

Jaime G. Carbonell
Language Technologies Institute, Carnegie Mellon
5000 Forbes Ave
Pittsburgh, PA 15213
jgc@cs.cmu.edu

Abstract

This paper studies methods that automatically detect *action-items* in e-mail, an important category for assisting users in identifying new tasks, tracking ongoing ones, and searching for completed ones. Since action-items consist of a short span of text, classifiers that detect action-items can be built from a document-level or a sentence-level view. Rather than commit to either view, we adapt a context-sensitive metaclassification framework to this problem to combine the *rankings* produced by different algorithms as well as different views. While this framework is known to work well for standard classification, its suitability for fusing rankers has not been studied. In an empirical evaluation, the resulting approach yields improved rankings that are less sensitive to training set variation, and furthermore, the theoretically-motivated reliability indicators we introduce enable the metaclassifier to now be applicable in any problem where the base classifiers are used.

1 Introduction

From business people to the everyday person, e-mail plays an increasingly central role in a modern lifestyle. With this shift, e-mail users desire improved tools to help process, search, and organize the information present in their ever-expanding inboxes. A system that ranks e-mails according to the likelihood of containing "to-do" or *action-items* can alleviate a user's time burden and is the subject of ongoing research throughout the literature.

From: Henry Hutchins <hhutchins@innovative.company.com>
To: Sara Smith; Joe Johnson; William Woolings
Subject: meeting with prospective customers
Hi All,
I'd like to remind all of you that the group from GRTY will be visiting us next Friday at 4:30 p.m. The schedule is:
+ 9:30 a.m. Informal Breakfast and Discussion in Cafeteria
+ 10:30 a.m. Company Overview
+ 11:00 a.m. Individual Meetings (Continue Over Lunch)
+ 2:00 p.m. Tour of Facilities
+ 3:00 p.m. Sales Pitch
In order to have this go off smoothly, I would like to practice the presentation well in advance. *As a result, I will need each of your parts by Wednesday.*
Keep up the good work!
–Henry

Figure 1: An E-mail with Action-Item (italics added).

In particular, an e-mail user may not always process all e-mails, but even when one does, some emails are likely to be of greater response urgency than others. These messages often contain action-items. Thus, while importance and urgency are not equal to action-item content, an effective action-item detection system can form one prominent subcomponent in a larger prioritization system.

Action-item detection differs from standard text classification in two important ways. First, the user is interested both in detecting whether an email contains action-items and in locating exactly where these action-item requests are contained within the email body. Second, action-item detection attempts

[*]This work was performed primarily while the first author was supported by Carnegie Mellon University.

to recover the sender's intent — whether she means to elicit response or action on the part of the receiver.

In this paper, we focus on the primary problem of presenting e-mails in a *ranked order* according to their likelihood of containing an action-item. Since action-items typically consist of a short text span — a phrase, sentence, or small passage — supervised input to a learning system can either come at the *document-level* where an e-mail is labeled yes/no as to whether it contains an action-item or at the *sentence-level* where each span that is an action-item is explicitly identified. Then, a corresponding document-level classifier or aggregated predictions from a sentence-level classifier can be used to estimate the overall likelihood for the e-mail.

Rather than commit to either view, we use a combination technique to capture the information each viewpoint has to offer on the current example. The STRIVE approach (Bennett *et al.*, 2005) has been shown to provide robust combinations of heterogeneous models for standard topic classification by capturing areas of high and low reliability via the use of reliability indicators.

However, using STRIVE in order to produce improved rankings has not been previously studied. Furthermore, while they introduce some reliability indicators that are general for text classification problems as well as ones specifically tied to naïve Bayes models, they do not address other classification models. We introduce a series of reliability indicators connected to areas of high/low reliability in kNN, SVMs, and decision trees to allow the combination model to include such factors as the sparseness of training example neighbors around the current example being classified. In addition, we provide a more formal motivation for the role these variables play in the resulting metaclassification model.

Empirical evidence demonstrates that the resulting approach yields a context-sensitive combination model that improves the quality of rankings generated as well as reducing the variance of the ranking quality across training splits.

2 Problem Approach

In contrast to related combination work, we focus on improving *rankings* through the use of a metaclassification framework. In addition, rather than simply focusing on combining models from different classification algorithms, we also examine combining models that have different *views*, in that both the qualitative nature of the labeled data and the application of the learned base models differ. Furthermore, we improve upon work on context-sensitive combination by introducing reliability indicators which model the sensitivity of a classifier's output around the current prediction point. Finally, we focus on the application of these methods to action-item data — a growing area of interest which has been demonstrated to behave differently than more standard text classification problems (*e.g. topic*) in the literature (Bennett and Carbonell, 2005).

2.1 Action-Item Detection

There are three basic problems for action-item detection. (1) *Document detection*: Classify an e-mail as to whether or not it contains an action-item. (2) *Document ranking*: Rank the e-mails such that all e-mail containing action-items occur as high as possible in the ranking. (3) *Sentence detection*: Classify each sentence in an e-mail as to whether or not it is an action-item.

Here we focus on the document ranking problem. Improving the overall ranking not only helps users find e-mails with action-items quicker (Bennett and Carbonell, 2005) but can decrease response times and help ensure that key e-mails are not overlooked.

Since a typical user will eventually process all received mail, we assume that producing a quality ranking will more directly measure the impact on the user than accuracy or F1. Therefore, we focus on ROC curves and area under the curve (AUC) since both reflect the quality of the ranking produced.

2.2 Combining Classifiers with Metaclassifiers

One of the most common approaches to classifier combination is stacking (Wolpert, 1992). In this approach, a metaclassifier observes a past history of classifier predictions to learn how to weight the classifiers according to their demonstrated accuracies and interactions. To build the history, cross-validation over the *training* set is used to obtain predictions from each base classifier. Next, a metalevel representation of the training set is constructed where each example consists of the class label and the predictions of the base classifiers. Finally, a metaclassifier is trained on the metalevel representation to learn a model of how to combine the base classifiers.

However, it might be useful to augment the history with information other than the predicted probabilities. For example, during peer review, reviewers

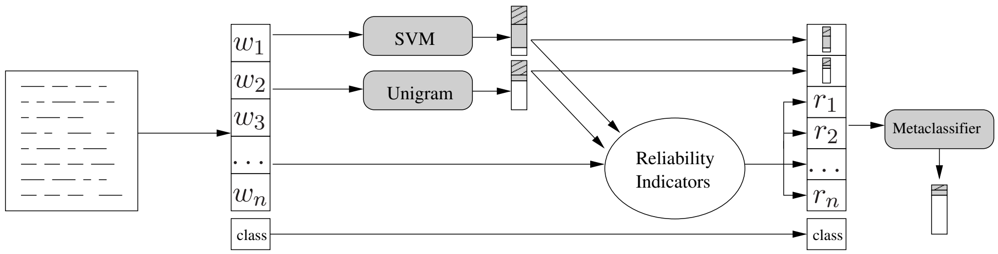

Figure 2: Architecture of *STRIVE*. In *STRIVE*, an additional layer of learning is added where the metaclassifier can use the context established by the reliability indicators and the output of the base classifiers to make an improved decision.

typically provide both a 1-5 acceptance rating and a 1-5 confidence. The first of these is related to an estimate of class membership, $P(\text{"accept"} \mid \text{paper})$, but the second is closer to a measure of expertise or a self-assessment of the reviewer's reliability on an example-by-example basis.

Automatically deriving such self-assessments for classification algorithms is non-trivial. The **St**acked **R**eliability **I**ndicator **V**ariable **E**nsemble framework, or *STRIVE*, demonstrates how to extend stacking by incorporating such self-assessments as a layer of reliability indicators and introduces a candidate set of functions (Bennett *et al.*, 2005).

The *STRIVE* architecture is depicted in Figure 2. From left to right: (1) a bag-of-words representation of the document is extracted and used by the base classifiers to predict class probabilities; (2) reliability indicator functions use the predicted probabilities and the features of the document to characterize whether this document falls within the "expertise" of the classifiers; (3) a metalevel classifier uses the base classifier predictions and the reliability indicators to make a more reliable combined prediction.

From the perspective of improving action-item rankings, we are interested in whether stacking *or* striving can improve the quality of rankings. However, we hypothesize that striving will perform better since it can learn a model that varies the combination rule based on the current example and thus, better capture when a particular classifier at the document-level or sentence-level, bag-of-words or n-gram representation, *etc.* will produce a reliable prediction.

2.3 Formally Motivating Reliability Indicators

While *STRIVE* has been shown to provide robust combination for topic classification, a formal motivation is lacking for the type of reliability indicators that are the most useful in classifier combination.

Assume we restrict our choice of metaclassifier to a linear model. One natural choice is to rank the e-mails according to the estimated posterior probability, $\hat{P}(\text{class} = \text{action item} \mid \mathbf{x})$, but in a linear combination framework it is actually more convenient to work with the estimated log-odds or logit transform which is monotone in the posterior, $\hat{\lambda} = \log \frac{\hat{P}(\text{class=action item}|\mathbf{x})}{1-\hat{P}(\text{class=action item}|\mathbf{x})}$ (Kahn, 2004).

Now, consider applying a metaclassifier to a single base classifier. Given only a classifier's probability estimates, a metaclassifier cannot improve on the estimates if they are well-calibrated (DeGroot and Fienberg, 1986). Thus a metaclassifier applied to a single base classifier corresponds to recalibration (Kahn, 2004).

Assume each of the n base models gives an uncalibrated log-odds estimate $\hat{\lambda}_i$. Then the combination model would have the form $\hat{\lambda}^*(\mathbf{x}) = W_0(\mathbf{x}) + \sum_{i=1}^{n} W_i(\mathbf{x})\hat{\lambda}_i(\mathbf{x})$ where the W_i are example dependent weight functions that the combination model learns. The obvious implication is that our reliability indicators can be informed by the optimal values for the weighting functions.

We can determine the optimal weights in a simplified case with a single base classifier by assuming we are given "true" log-odds values, λ, and a family of distributions $\Delta_{\mathbf{x}}$ such that $\Delta_{\mathbf{x}} = p(\mathbf{z} \mid \mathbf{x})$ encodes what is local to $\mathbf{x}$ by giving the probability of drawing a point $\mathbf{z}$ near to $\mathbf{x}$. We use Δ instead of $\Delta_{\mathbf{x}}$ for notational simplicity. Since Δ encodes the example dependent nature of the weights, we can drop $\mathbf{x}$ from the weight functions. To find weights that minimize the squared difference between the true log-odds and the estimated log-odds in the Δ vicinity of $\mathbf{x}$, we can solve a standard regression problem, $\operatorname{argmin}_{w_0,w_1} E_{\Delta}\left[\left(w_1\,\hat{\lambda} + w_0 - \lambda\right)^2\right]$. Under the assumption $\mathrm{VAR}_{\Delta}\left[\hat{\lambda}\right] \neq 0$, this yields:

$$w_0 = E_\Delta[\lambda] - w_1 E_\Delta\left[\hat{\lambda}\right] \quad (1)$$

$$w_1 = \frac{\mathrm{COV}_\Delta\left[\hat{\lambda}, \lambda\right]}{\mathrm{VAR}_\Delta\left[\hat{\lambda}\right]} = \frac{\sigma_\lambda}{\sigma_{\hat{\lambda}}}\rho_{\lambda,\hat{\lambda}} \quad (2)$$

where σ and ρ are the stdev and correlation coefficient under Δ, respectively. The first parameter is a measure of calibration that addresses the question, "How far off on average is the estimated log-odds from the true log-odds in the local context?" The second is a measure of correlation, "How closely does the estimated log-odds vary with the true log-odds?" Note that the second parameter depends on the local sensitivity of the base classifier, $\mathrm{VAR}_\Delta^{1/2}\left[\hat{\lambda}\right] = \sigma_{\hat{\lambda}}$. Although we do not have true log-odds, we *can* introduce local density models to estimate the local sensitivity of the model.

In particular, we introduce a series of reliability indicators by first defining a Δ distribution and either computing $\mathrm{VAR}_\Delta\left[\hat{\lambda}\right]$, $E_\Delta\left[\hat{\lambda}\right]$ or the closely related terms $\mathrm{VAR}_\Delta\left[\hat{\lambda}(\mathbf{z}) - \hat{\lambda}(\mathbf{x})\right]$, $E_\Delta\left[\hat{\lambda}(\mathbf{z}) - \hat{\lambda}(\mathbf{x})\right]$. We use the resulting values for an example as features for a linear metaclassifier. Thus we use a context-dependent bias term but leave the more general model for future work.

2.4 Model-Based Reliability Indicators

As discussed in Section 2.3, we wish to define local distributions in order to compute the local sensitivity and similar terms for the base classification models. To do so, we define local distributions that have the same "flavor" as the base classification model.

First, consider the kNN classifier. Since we are concerned with how the decision function would change as we move locally around the current prediction point, it is natural to consider a set of shifts defined by the k neighbors. In particular, let $\mathbf{d}_i$ denote the document that has been shifted by a factor β_i toward the ith neighbor, *i.e.* $\mathbf{d}_i = \mathbf{d} + \beta_i(\mathbf{n}_i - \mathbf{d})$. We use the largest β_i such that the closest neighbor to the new point is the original document, *i.e.* the boundary of the Voronoi cell (see Figure 3). Clearly, β_i will not exceed 0.5, and we can find it efficiently using a simple bisection algorithm. Now, let Δ be a uniform point-mass distribution over the shifted points and $\hat{\lambda}_{k\mathrm{NN}}$, the output score of the kNN model.

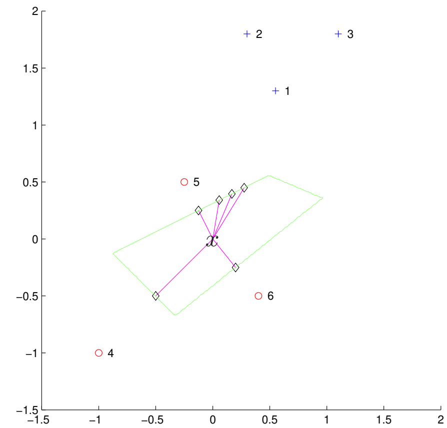

Figure 3: Illustration of the kNN shifts produced for a prediction point x using the numbered points as its neighborhood.

Given this definition of Δ, it is now straightforward to compute the kNN based reliability indicators: $E_\Delta[\hat{\lambda}_{k\mathrm{NN}}(\mathbf{z}) - \hat{\lambda}_{k\mathrm{NN}}(\mathbf{x})]$ and $\mathrm{Var}_\Delta^{1/2}[\hat{\lambda}_{k\mathrm{NN}}(\mathbf{z}) - \hat{\lambda}_{k\mathrm{NN}}(\mathbf{x})]$.

Similarly, we define variables for the SVM classifier by considering a document's locality in terms of nearby support vectors from the set of support vectors, $\mathcal{V}$. To determine β_i, we define it in terms of the closest support vector in $\mathcal{V}$ to $\mathbf{d}$. Let ϵ be half the distance to the nearest point in $\mathcal{V}$, *i.e.* $\epsilon = \frac{1}{2}\min_{\mathbf{v}\in\mathcal{V}} \|\mathbf{v} - \mathbf{d}\|$. Then $\beta_i = \frac{\epsilon}{\|\mathbf{v}_i - \mathbf{d}\|}$.[1] Thus, the shift vectors are all rescaled to have the same length. Now, we must define a probability for the shift. We use a simple exponential based on ϵ and the relative distance from the document to the support vector defining this shift. Let $\mathbf{d}_i \sim \Delta$ where $P_\Delta(\mathbf{d}_i) \propto \exp(-\|\mathbf{v}_i - \mathbf{d}\| + 2\epsilon)$ and $\sum_{i=1}^{V} P_\Delta(\mathbf{d}_i) = 1$.[2]

Given this definition of Δ, we compute the SVM based reliability indicators: $E_\Delta[\hat{\lambda}_{\mathrm{SVM}}(\mathbf{z}) - \hat{\lambda}_{\mathrm{SVM}}(\mathbf{x})]$ and $\mathrm{Var}_\Delta^{1/2}[\hat{\lambda}_{\mathrm{SVM}}(\mathbf{z}) - \hat{\lambda}_{\mathrm{SVM}}(\mathbf{x})]$.

Space prevents us from presenting all the derivations here. However, we also define decision-tree based variables where the locality distribution gives high probability to documents that would land in nearby leaves. For a multinomial naïve Bayes model (NB), we define a distribution of documents identical to the prediction document except having an occurrence of a single feature deleted. For the multivariate Bernoulli naïve Bayes (MBNB) model

[1] We assume that the minimum distance is not zero. If it is zero, then we return zero for all of the variables.

[2] As is standard to handle different document lengths, we take the distance between documents after they have been normalized to the unit sphere.

that models feature presence/absence, we use a distribution over all documents that has one presence/absence bit flipped from the prediction document. It is interesting to note that the variables from the naïve Bayes models can be shown to be equivalent to variables introduced by Bennett *et al.* (2005) — although those were derived in a different fashion by analyzing the weight a single feature carries with respect to the overall prediction.

Furthermore, from this starting point, we go on to define similar variables of possible interest. Including the two for each model described here, we define 10 kNN variables, 5 SVM variables, 2 decision-tree variables, 6 NB model based variables, and 6 MBNB variables. We describe these variables as well as implementation details and computational complexity results in (Bennett, 2006).

3 Experimental Analysis

3.1 Data

Our corpus consists of e-mails obtained from volunteers at an educational institution and covers subjects such as: organizing a research workshop, arranging for job-candidate interviews, publishing proceedings, and talk announcements. After eliminating duplicate e-mails, the corpus contains 744 messages with a total of 6301 automatically segmented sentences. A human panel labeled each phrase or sentence that contained an explicit request for information or action. 416 e-mails have no action-items and 328 e-mails contain action-items. Additional information such as annotator agreement, distribution of message length, *etc.* can be found in (Bennett and Carbonell, 2005). An anonymized corpus is available at http://www.cs.cmu.edu/˜pbennett/action-item-dataset.html.

3.2 Feature Representation

We use two types of feature representation: a bag-of-words representation which uses all unigram tokens as the feature pool; and a bag-of-n-grams where n includes all n-grams where $n \leq 4$. For both representations at both the document-level and sentence-level, we used only the top 300 features by the chi-squared statistic.

3.3 Document-Level Classifiers

kNN

We used a s-cut variant of kNN common in text classification (Yang, 1999) and a tfidf-weighting of the terms with a distance-weighted vote of the neighbors to compute the output. k was set to be $2(\lceil \log_2 N \rceil + 1)$ where N is the number of training points. [3] The score used as the uncalibrated log-odds estimate of being an action-item is:

$$\hat{\lambda}_{k\mathrm{NN}}(\mathbf{x}) = \sum_{\mathbf{n} \in k\mathrm{NN}(\mathbf{x}) | c(\mathbf{n}) = {\mathrm{action} \atop \mathrm{item}}} \cos(\mathbf{x}, \mathbf{n}) \quad - \sum_{\mathbf{n} \in k\mathrm{NN}(\mathbf{x}) | c(\mathbf{n}) \neq {\mathrm{action} \atop \mathrm{item}}} \cos(\mathbf{x}, \mathbf{n}).$$

SVM

We used a linear SVM as implemented in the SVM*light* package v6.01 (Joachims, 1999) with a tfidf feature representation and L2-norm. All default settings were used. SVM's margin score, $\sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$, has been shown to empirically behave like an uncalibrated log-odds estimate (Platt, 1999).

Decision Trees

For the decision-tree implementation, we used the WinMine toolkit and refer to this as *Dnet* below (Microsoft Corporation, 2001). Dnet builds decision trees using a Bayesian machine learning algorithm (Chickering *et al.*, 1997; Heckerman *et al.*, 2000). The estimated log-odds is computed from a Laplace correction to the empirical probability at a leaf node.

Naïve Bayes

We use a multinomial naïve Bayes (NB) and a multivariate Bernoulli naïve Bayes classifier (MBNB) (McCallum and Nigam, 1998). For these classifiers, we smoothed word and class probabilities using a Bayesian estimate (with the word prior) and a Laplace m-estimate, respectively. Since these are probabilistic, they issue log-odds estimates directly.

3.4 Sentence-Level Classifiers

Each e-mail is automatically segmented into sentences using RASP (Carroll, 2002). Since the corpus has fine grained labels, we can train classifiers to classify a sentence. Each classifier in Section 3.3 is also used to learn a sentence classifier. However, we then must make a document-level prediction.

In order to produce a ranking score, the confidence that the document contains an action-item is:

$$\hat{\lambda}(\mathbf{d}) = \begin{cases} \frac{1}{n(\mathbf{d})} \sum_{s \in \mathbf{d} | \pi(s) = 1} \hat{\lambda}(s), & \exists s \in \mathbf{d} | \pi(s) = 1 \\ \frac{1}{n(\mathbf{d})} \max_{s \in \mathbf{d}} \hat{\lambda}(s) & o.w. \end{cases}$$

[3] This rule is not guaranteed be optimal for a particular value of N but is motivated by theoretical results which show such a rule converges to the optimal classifier as the number of training points increases (Devroye *et al.*, 1996).

where s is a sentence in document $\mathbf{d}$, π is the classifier's 1/0 prediction, $\hat{\lambda}$ is the score the classifier assigns as its confidence that $\pi(s) = 1$, and $n(\mathbf{d})$ is the greater of 1 and the number of (unigram) tokens in the document. In other words, when any sentence is predicted positive, the document score is the length normalized sum of the sentence scores above threshold. When no sentence is predicted positive, the document score is the maximum sentence score normalized by length. The length normalization compensates for the fact that we are more likely to emit a false positive the longer a document is.

3.5 Stacking

To examine the hypothesis that the reliability indicators provide utility beyond the information present in the output of the 20 base classifiers ($2 \text{ representations} * 2 \text{ views} * 5 \text{ classifiers}$), we construct a linear stacking model which uses only the base classifier outputs and no reliability indicators as a baseline. For the implementation, we use SVM^{light} with default settings. The inputs to this classifier are normalized to have zero mean and a scaled variance.

3.6 Striving

Since we are constructing base classifiers for both the bag-of-words and bag-of-n-grams representations, this gives 58 reliability indicators from computing the variables in Section 2.4 for the document-level classifiers ($58 = 2 * [6 + 6 + 10 + 5 + 2]$).

Although the model-based indicators are defined for each sentence prediction, to use them at the document-level we must somehow combine the reliability indicators over each sentence. The simplest method is to average each classifier-based indicator across the sentences in the document. We do so and thus obtain another 58 reliability indicators.

Furthermore, our model might benefit from some of the structure a sentence-level classifier offers when combining document predictions. Analogous to the sensitivity of each base model, we can consider such indicators as the mean and standard deviation of the classifier confidences across the sentences within a document. For each sentence-level base classifier, these become two more indicators which we can benefit from when combining document predictions. This introduces 20 more variables ($20 = 2 \text{ representations} * 2 * 5 \text{ classifiers}$).

Finally, we include the 2 basic voting statistic reliability-indicators (*PercentPredictingPositive* and *PercentAgreeWBest*) that Bennett *et al.* (2005) found useful for topic classification. This yields a total of 138 reliability-indicators ($138 = 58 + 20 + 58 + 2$). With the 20 classifier outputs, there are a total of 158 input features for striving to handle.

As with stacking, we use SVM^{light} with default settings and normalize the inputs to this classifier to have zero mean and a scaled variance.

3.7 Performance Measures

We wish to improve the rankings of the e-mails in the inbox such that action-item e-mails occur higher in the inbox. Therefore, we use the area under the curve (AUC) of an ROC curve as a measure of ranking performance. AUC is a measure of overall model and ranking quality that has gained wider adoption recently and is equivalent to the Mann-Whitney-Wilcoxon sum of ranks test (Hanley and McNeil, 1982). To put improvement in perspective, we can write our relative reduction in residual area (RRA) as $\frac{1-\text{AUC}}{1-\text{AUC}_{\text{baseline}}}$. We present gains relative to the best AUC performer (bRRA), and relative to perfect dynamic selection performance, (dRRA), which assumes we could accurately *dynamically* choose the best classifier per cross-validation run.

The F1 measure is the harmonic mean of precision and recall and is common throughout text classification (Yang and Liu, 1999). Although we are not concerned with F1 performance here, some users of the system might be interested in improving ranking while having negligible negative effect on F1. Therefore, we examine F1 to ensure that an improvement in ranking will not come at the cost of a statistically significant decrease in F1.

3.8 Experimental Methodology

To evaluate performance of the combination systems, we perform 10-fold cross-validation and compute the average performance. For significance tests, we use a two-tailed t-test (Yang and Liu, 1999) to compare the values obtained during each cross-validation fold with a p-value of 0.05.

We examine two hypotheses: Stacking will outperform all of the base classifiers; Striving will outperform all the base classifiers and stacking.

3.9 Results & Discussion

Table 1 presents the summary of results. The best performer in each column is in bold. If a combination method statistically significantly outperforms all base classifiers, it is underlined.

	F1	AUC	bRRA	dRRA
Document-Level, Bag-of-Words Representation				
Dnet	0.7398	0.8423	1.41	1.78
NB	0.6905	0.7537	2.27	2.91
MBNB	0.6729	0.7745	2.00	2.49
SVM	0.6918	0.8367	1.48	1.87
kNN	0.6695	0.7669	2.17	2.74
Document-Level, Ngram Representation				
Dnet	0.7412	0.8473	1.38	1.77
NB	0.7361	0.8114	1.75	2.23
MBNB	0.7534	0.8537	1.30	1.61
SVM	0.7392	0.8640	1.24	1.59
kNN	0.7021	0.8244	1.62	2.01
Sentence-Level, Bag-of-Words Representation				
Dnet	0.7793	0.8885	1.00	1.27
NB	0.7731	0.8645	1.21	1.50
MBNB	0.7888	0.8699	1.14	1.42
SVM	0.6985	0.8548	1.34	1.70
kNN	0.6328	0.6823	2.98	3.88
Sentence-Level, Ngram Representation				
Dnet	0.7521	0.8723	1.13	1.42
NB	**0.8012**	0.8723	1.15	1.46
MBNB	0.8010	0.8777	1.10	1.38
SVM	0.7842	0.8620	1.23	1.58
kNN	0.6811	0.8078	1.76	2.29
Metaclassifiers				
Stacking	0.7765	0.8996	0.88	1.12
STRIVE	0.7813	**0.9145**	**0.76**	**0.94**

Table 1: Base classifier and combiner performance

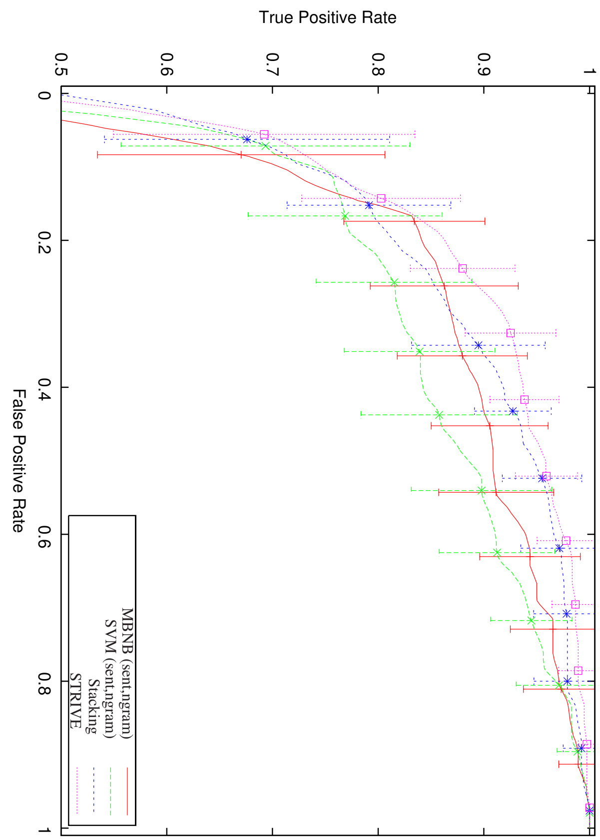

Figure 4: ROC curves (rotated).

Now, we turn to the issue of whether combination improves the ranking of the documents. Examining the results in Table 1, we see that *STRIVE* statistically significantly beats every other classifier according to AUC. Stacking outperforms the base classifiers with respect to AUC but not statistically significantly.

Examining F1, we see that neither combination method outperforms the best base classifier, *NB (sent,ngram)*. If we examine the hypothesis of whether this base classifier significantly outperforms either combination method, the hypothesis is rejected. Thus, STRIVE improves the overall ranking with a negligible effect on F1.

Finally, we compare the ROC curves of striving, stacking, and two of the most competitive base classifiers in Figure 4. We see that striving loses by a slight amount to stacking early in the curve but still beats the base classifiers. Later in the curve, it dominates all the classifiers. If we examine the curves using error bars, we see that the variance of *STRIVE* drops faster than the other classifiers as we move further along the x-axis. Thus, *STRIVE*'s ranking quality varies less with changes to the training set.

4 Related Work

Several researchers have considered text classification tasks similar to action-item detection. Cohen *et al.* (2004) describe an ontology of "speech acts", such as "Propose a Meeting", and attempt to predict when an e-mail contains one of these speech acts. Corston-Oliver *et al.* (2004) consider detecting items in e-mail to "Put on a To-Do List" using a sentence-level classifier. In earlier work (Bennett and Carbonell, 2005), we demonstrated that sentence-level classifiers typically outperform document-level classifiers on this problem and examined the underlying reasons why this was the case. Furthermore, we presented user studies demonstrating that users identify action-items more rapidly when using the system.

In terms of classifier combination, a wide variety of work has been done in the arena. The STRIVE metaclassification approach (Bennett *et al.*, 2005) extended Wolpert's stacking framework (Wolpert, 1992) to use reliability indicators. In recent work, Lee *et al.* (2006) derive variance estimates for naïve Bayes and tree-augmented naïve Bayes and use them in the combination model. Our work complements theirs by laying groundwork for how to compute variance estimates for models such as kNN that have no obvious probabilistic component.

5 Future Work and Conclusion

While there are many interesting directions for future work, the most interesting is to directly integrate the sensitivity and calibration quantities derived into the more general model discussed in Section 2.3.

In this paper, we took an existing approach to context-dependent combination, STRIVE, that used many *ad hoc* reliability indicators and derived a formal motivation for classifier model-based local sensitivity indicators. These new reliability indicators are efficiently computable, and the resulting combination outperformed a vast array of alternative base classifiers for ranking in an action-item detection task. Furthermore, the combination results yielded a more robust performance relative to variation in the training sets. Finally, we demonstrated that the STRIVE method could be successfully applied to ranking.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

Paul N. Bennett and Jaime Carbonell. 2005. Feature representation for effective action-item detection. In *SIGIR '05, Beyond Bag-of-Words Workshop*.

Paul N. Bennett, Susan T. Dumais, and Eric Horvitz. 2005. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8:67–100.

Paul N. Bennett. 2006. *Building Reliable Metaclassifiers for Text Learning*. Ph.D. thesis, CMU. CMU-CS-06-121.

John Carroll. 2002. High precision extraction of grammatical relations. In *COLING '02*.

D.M. Chickering, D. Heckerman, and C. Meek. 1997. A Bayesian approach to learning Bayesian networks with local structure. In *UAI '97*.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *EMNLP '04*.

Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *Text Summarization Branches Out: Proceedings of the ACL '04 Workshop*.

Morris H. DeGroot and Stephen E. Fienberg. 1986. Comparing probability forecasters: Basic binary concepts and multivariate extensions. In P. Goel and A. Zellner, editors, *Bayesian Inference and Decision Techniques*. Elsevier.

Luc Devroye, László Györfi, and Gábor Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, NY.

James A. Hanley and Barbara J. McNeil. 1982. The meaning and use of the area under a recever operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In Bernhard Schölkopf, Christopher J. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Joseph M. Kahn. 2004. *Bayesian Aggregation of Probability Forecasts on Categorical Events*. Ph.D. thesis, Stanford University, June.

Chi-Hoon Lee, Russ Greiner, and Shaojun Wang. 2006. Using query-specific variance estimates to combine bayesian classifiers. In *ICML '06*.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI '98, Workshops*. TR WS-98-05.

Microsoft Corporation. 2001. WinMine Toolkit v1.0. http://research.microsoft.com/~dmax/WinMine/ContactInfo.html.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter Bartlett, Bernhard Scholkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *SIGIR '99*.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):67–88.