Collective Classification of Posts to Internet Forums

Pádraig Ó Duinn and Derek Bridge*

Insight Centre for Data Analytics,
School of Computer Science and Information Technology,
University College Cork, Ireland
padraig.oduinn|derek.bridge@insight-centre.org

Abstract. We investigate automatic classification of posts to Internet forums. We use collective classification methods, which simultaneously classify related objects — in our case, the posts in a thread. Specifically, we compare the Iterative Classification Algorithm (ICA) with Conditional Random Fields and with conventional classifiers (k-Nearest Neighbours and Support Vector Machines). The ICA algorithm invokes a local classifier, for which we use the kNN classifier. Our main contributions are two-fold. First, we define experimental protocols that we believe are suitable for offline evaluation in this domain. Second, by using these protocols to run experiments on two datasets, we show that ICA with kNN has significantly higher accuracy across most of the experimental conditions.

Keywords: collective classification, Internet forums, incremental

1 Introduction

Internet forums are places for debate and discussion, for the sharing of experience, and for collaborative problem-solving. Users assume a variety of roles. They be *consumers*, seeking information from existing posts; as *contributors*, they might initiate new threads or add posts to existing threads; and as *moderators*, they might monitor discussions and intervene to enforce forum policies.

Forums display the posts within a thread in a linear order, most commonly in chronological order. But a thread's argument structure may not be linear. A post may respond to a post other than the one that immediately precedes it. As the thread becomes longer, contributors, who may be unwilling to read all previous posts, may repeat previous posts, insert unrelated posts, or even start new threads that replicate existing threads. This makes it more difficult for the different categories of users to achieve their goals.

We are interested in forum management software that infers the argument structure and uses it to assist users. For example, for consumers, the software

* The authors gratefully acknowledge the financial support of the Irish Research Council. This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

might summarise a whole thread, or present it in a hierarchical fashion, or suppress redundant or irrelevant posts. For contributors, it might provide guidance for increasing the usefulness of a contribution. For moderators, it might help identify such things as: spam posts, misplaced posts, flame wars, troll users, or threads that need locking, merging or deleting.

In this paper we are investigating post classification. We wish to classify each post using a label that denotes the post's role within its thread. In a trouble-shooting forum, the labels might include 'question', 'answer', 'clarification', and so on. We make the simplifying assumption in this work that each post has a single label. This is fairly reasonable for a trouble-shooting forum, but more questionable for forums where threads are more discursive. We may lift this assumption in future work. We believe that classifying posts by their role in the thread is a useful precursor to the kinds of tasks we mentioned in the previous paragraph, for example inferring the argument structure so that threads can be presented hierarchically.

In traditional classification, objects are considered in isolation, and classified independently. However, it may be the case that objects are inter-related. In this case, classification accuracy can often be improved by taking into account information from the related objects. When this information includes the *predicted labels* of the related objects, the term *collective classification* is used.

We compare two collective classifiers. One is a Conditional Random Fields (CRF) classifier, and this approach has already appeared in the literature [9]. The other is the Iterative Classification Algorithm (ICA) [14]. ICA has not to our knowledge been previously used for post classification. We expect collective classifiers to obtain higher classification accuracy than traditional classifiers.

The Iterative Classification Algorithm, which we explain fully in Section 2.1, repeatedly invokes another, 'local' classifier [14]. In principle, the local classifier can be any classification algorithm. In practice, the local classifier must be cheap to train because it is re-trained repeatedly on a changing training set. The training set is updated on each iteration with the latest predicted labels of the related objects. For our local classifier, we use a k-nearest neighbours case-based classifier. Case-based classifiers are lazy learners and, as such, have negligible training costs, making them ideal for use within ICA.

In Section 2, we describe the task of post classification in general and we describe the two collective classifiers that we are comparing in this paper (ICA and CRF). Section 3 describes the two datasets which we use in our experiments. Section 4 defines our experimental protocols. Section 5 presents the results of some of the experiments that we have run. The final sections present related work (Section 6) and conclusions (Section 7).

2 Post Classification

A forum comprises a set of threads T. Each thread $t \in T$ comprises a sequence of posts, $t = \langle p_1, p_2, \ldots, p_n \rangle$. Each post has a label $c(p_i, t)$ which is drawn from a finite set of labels C. Note that the label may depend on both the post, p_i, and the

thread in which it appears, t: in principle, an identical post in a different thread might receive a different label. The task of the classifier is to predict the label, $\hat{c}(p_i, t)$. It might seem more natural to assume that the label and predicted label of a post p_i both depend only on the preceding posts in the thread, $\{p_j | j < i\}$, rather than the whole thread, t. Certainly, in the experiments we report in this paper, we only use the preceding posts. But, a formalism that allows the label to depend on the whole thread leaves open the possibility that classification can depend on reactions to a post — which come later in the thread. This is something we may experiment with in the future.

Each post p_i will be described by attributes. Three categories of attributes may be used:

- Attributes derived from observations of p_i itself, e.g. author information and lexical attributes such as word counts and counts of the number of punctuation symbols.
- Attributes derived from observations of other posts in the thread, $p_j \in t, j \neq i$, e.g. p_j's author information and lexical attributes, or the degree of similarity between the titles of p_i and p_j. In some cases, it may be possible to *observe* the *labels* of other posts in the thread, e.g. if they are given as part of training data, in which case attributes derived from these would also fall into this category. But, for most posts we cannot observe their labels; we can only *predict* them, and so their labels fall into the third category.
- Attributes derived from the *predicted labels* of other posts in the thread, $\hat{c}(p_j, t)$ where $p_j \in t, j \neq i$, e.g. the predicted label of the previous post, or a count of the number of previous posts that are predicted to have a certain label.

Attributes in the first two categories are called *observed attributes*, whereas attributes in the third category are called *unobserved attributes*. Alternatively, attributes in the first category are *non-relational attributes*, whereas attributes in the last two categories are *relational attributes*.

A classifier that uses only the first of these categories is said to be a *non-relational classifier*; one that uses the second of these is said to be a *relational classifier*; and one that uses the third is said to be a *collective classifier* [14].

Collective classifiers are more usually formalised for the case of a graph. The job is to predict labels for a subset of the nodes in the graph — those whose labels are not already known. In addition to a node's attributes, the classifier may use relational attributes, i.e. the attributes and labels of other nodes. Usually, there is a definition of neighbourhood within the graph, and a node's relational attributes come only from the other nodes in its neighbourhood. In this paper, we are taking the special case where the graph is a chain — a linear arrangement of posts.

Collective classifiers fall into two main types: those that make use of a local classifier, and those that define and optimize a set of weights in a global objective function [15]. We will explain one representative of the first of these two approaches (ICA) and one representative of the second approach (CRF).

Algorithm 1 Iterative Classification of Forum Posts

t = thread of posts $\langle p_1, p_2, \ldots, p_n \rangle$
m = the index of the first unlabelled post in t, $m \in [1, n+1]$
M_O = local classifier that uses observed attributes only
M_U = local classifier that uses observed and unobserved attributes

// Step 1: Initial classification
// Step 1a: Derive observed attributes
for all $i \leftarrow 1$ to n **do**
  $\boldsymbol{o_i} \leftarrow p_i$'s observed attributes
end for
// Step 1b: Classify unlabelled posts using observed attributes only
for all $i \leftarrow m$ to n **do**
  $c_i \leftarrow M_O(\boldsymbol{o_i})$
end for
// Step 2: Iterative classification
repeat
  // Step 2a: Derive unobserved attributes
  for all $i \leftarrow m$ to n **do**
    $\boldsymbol{u_i} \leftarrow p_i$'s unobserved attributes
  end for
  // Step 2b: Re-classify unlabelled posts using all attributes
  for all $i \leftarrow m$ to n **do**
    $c_i \leftarrow M_U(\boldsymbol{o_i}, \boldsymbol{u_i})$
  end for
until class labels have stabilized or a certain number of iterations has elapsed
return c_i for each p_i, $i \geq m$

2.1 Iterative Classification Algorithm

The Iterative Classification Algorithm (ICA) is probably the simplest collective classifier. Algorithm 1 presents the version that we use for forum post classification. We allow for the possibility that we already know the labels of some of the posts in thread t; for example, where they are supplied in a training set. We use index $m \in [1, n+1]$ to designate the first thread whose label is not known. In other words, we know the labels of posts p_i for $i < m$; and we must predict the labels of posts p_j for $j \geq m$. If $m = 1$, then we have not been given any true labels in this thread; if $m = n+1$; all posts are already labelled, and there is no work to be done.

Step 1 of ICA is to establish initial labels for the unlabelled posts. This is done using a local classifier which only considers the post's observed attributes. Step 2 continually re-classifies posts, but now using all attributes, until predictions have stabilized or, as in our experiments, a certain number of iterations (in this paper, 10) has been performed. Specifically, Step 2a computes the unobserved attributes. These may simply be the currently predicted labels of the other posts, or attributes that are derived from these. In Step 2b, posts are re-classified using both observed and unobserved attributes.

Note that we make the assumption that posts are processed by the algorithm in chronological order (for-loops run *up* to n). More sophisticated versions of ICA include steps for choosing the order of processing, e.g. based on the degree of uncertainty of the currently predicted label (e.g. [14]).

For both local classifiers, M_O and M_U, we use a k-nearest-neighbours case-based classifier [12]. Its case base contains posts, described by a full set of attributes. Each post is stored with its label. To classify a new post, we retrieve its k-nearest neighbours. The local distance measure we use to compare one attribute-value with another is the range-normalised absolute difference [20]; the global distance measure is simply an unweighted average of the values of the local distance measures for each attribute. M_O, however, only computes distances over the new post's observed attributes, whereas M_U computes distances over all attributes. The predicted class is obtained by a distance-weighted vote of the nearest neighbours. In Section 1, we explained why a lazy learner, such as a case-based classifier, is especially well-suited for use in ICA.

2.2 Conditional Random Fields Classification

The main alternative way to carry out collective classification seeks to optimize a global objective function, e.g., one that weights individual feature functions. In collective classification, techniques include Loopy Belief Propagation (LBP) [18] and Mean-Field Relaxation Labelling (MF) [19]. Although rarely presented as collective classification, Conditional Random Fields (CRFs) are similar [10]; indeed, LBPs are partly inspired by CRFs [18].

However, post classification does not require the full generality of these algorithms, since the posts form a chain, and not an arbitrary graph. The problem simplifies to a sequence labelling problem. For sequence labelling, classifiers include Maximum Entropy Markov Models (MEMMs) [11] and Linear Chain Conditional Random Fields. We will use a Linear Chain CRF, since it overcomes the label bias problem exhibited by MEMMs [10]. Specifically, we use CRF++, which is an open source Linear Chain CRF.[1]

There is another reason for choosing to use Linear Chain CRFs: they were the best-performing approach in an earlier post classification study by Kim et al. [9], where they were compared with a conventional Maximum Entropy learner (ME) and a Structural Support Vector Machine (HMM-SVM) [8]. Here, we are able to compare CRFs to a more competitive approach (ICA) and to do so using the experimental protocols that we define in Section 4, which we believe give more reliable results.

3 Datasets

Our goal is to compare post classification using ICA with the Linear Chain CRF that was used in [9] and with conventional classifiers, namely k-nearest-

[1] `http://crfpp.sourceforge.net/`

neighbours (kNN) and Support Vector Machines (SVM). We use two datasets, which we describe here.

The first, the CNET dataset, is the one created by Kim et al. for the work they describe in [9]. It comprises 320 threads but five of them include some unlabelled posts, so we use only the remaining 315 threads; these contain a total of 1332 posts from CNET forums that deal with trouble-shooting for operating systems, software, hardware and web development.[2]

For the CNET dataset, the set C contains twelve labels. The two main labels, *Question* and *Answer*, are sub-divided into sub-categories, having double-barrelled names. For example, a post whose label is *Question-question* contains a new question. A post whose label is *Question-add* contains information which supplements some other question, e.g. providing additional information or asking a follow-up. Similarly, *Answer-answer* offers solutions to problems or answers to questions posed in question posts, and *Answer-add* provides information that supplements some other answer. The other labels are *Question-confirmation*, *Question-correction*, *Answer- confirmation*, *Answer-correction*, *Answer-objection*, *Resolution*, *Reproduction* and *Other*. Note that one label (*Answer-correction*) never occurs.

The true labels of each post were assigned by two human annotators, with adjudication in the case of disagreement. In the original dataset, 65 posts were assigned multiple labels. It is not clear how [9] handles these. For each of these, we made a judgement about which label was best and discarded other labels.

The second dataset, the FIRE dataset, is also from a forum dealing with trouble-shooting for computers[3], which was used by Bhatia et al. in [1]. It comprises 100 threads, containing a total of 556 posts. Bhatia et al. use a set C of just eight labels: *Question*, *Repeat Question*, *Clarification*, *Further Details*, *Solution*, *Positive Feedback*, *Negative Feedback* and *Junk*.

We extracted a set of attributes from each post in the two datasets. We based the attributes on those described in [9]. All attributes are numeric-valued. The observed non-relational attributes are:

- **Punctuation:** Punctuation marks can help identify the nature of a post. The number of explanation marks in a post and the number of question marks in a post are used as two attributes.
- **URLs:** Posts which contain URLs are more likely to be in one of the *Answer* classes. The number of URLs in a post is used as an attribute.
- **Thread Initiator:** The user who starts a thread is less likely to post answers in that thread. We use a binary feature which indicates whether the post's author is also the person who initiated the thread.
- **Post Position:** The position of a post in a thread is important, not least since the first post is always a question. This attribute records the relative position of the current post in its thread, i.e. the ratio of its position to the number of posts in the thread.

[2] `http://forums.cnet.com/`
[3] `http://www.isical.ac.in/~fire/`

– **Author Profile:** Some users ask more questions than they answer, and vice versa. We use the class priors for that user. In other words, we have $|C|$ attributes, one for each class label, where each is the probability that this user creates posts of this class, calculated from the labels of the user's posts in the training set.

The observed relational attributes are:

– **Title Similarity:** A post whose title is similar to the title of a previous post is often some form of reply to that previous post. We compute the cosine similarity between the post's title and the titles of all previous posts. We find the previous post whose title is most similar to that of the current post. The relative position for that post is used as the attribute. (This attribute is not used in the experiments with the FIRE dataset because its posts do not have titles.)
– **Post Similarity:** Similarity of content is also a sign of one post responding to another. This attribute is the same as the previous one, but this time based on cosine similarity of post content, rather than post title.

The unobserved relational attributes are:

– **Previous Post**: This attribute records the label of the previous post in the thread. To make it numeric-valued, we use one-hot encoding, i.e. we use $|C|$ attributes, one for each class label, setting them all to zero except for the one that corresponds to the class of the previous post, which is set to one.
– **Previous Post from Same User:** This attribute records the label of the previous post in the thread by the author of the current post. Here again we used one-hot encoding.
– **Full History:** This attribute records the labels of all previous posts in the thread. Again there are $|C|$ attributes, but this time they are counts: the number of previous posts having each label.
– **User History:** This attribute records the labels of all previous posts of the author of the current post. Like Full History, there are $|C|$ attributes, represented as counts: the number of posts having each label.

Notice that we have no lexical attributes (such as unigram and bigram frequencies). In their experiments, Kim et al. did not run the CRF with lexical attributes because there were too many such attributes. They claim the CRF does not scale well to large numbers of attributes. We confirmed this: CRF++ crashed on 1000 attributes. To keep the comparison fair, we do not use such attributes for the ICA even though the local classifiers that we are using cope reasonably well with high-dimensional data. In any case, Kim et al. found that their other classifiers (ME and HMM-SVM) were not very accurate when trained on lexical attributes, possibly because the amount of training data (just 1332 posts of fewer than 100 words each) was too small to allow useful generalization.

Our experimental protocols (next section) work in a chronological fashion, thus requiring that each post have a timestamp. These were missing from the CNET dataset, so we scraped them from the original forum.

4 Experiments

Both Kim et al. and Bhattia et al. use a 10-fold cross-validation methodology. While this gives robust accuracy estimates, it fails to model the way that threads grow over time and the fact that a later post must be classified using the possibly erroneous class labels of previous posts in the same thread. We define *incremental* protocols instead, giving a more realistic setting for the experiments.

In each protocol, we train classifiers on all posts with their true labels from the first 100 days of the dataset. We then ask the classifier to predict labels for all of the next day's posts, and we measure accuracy against the true labels. We repeat this on a daily basis until the dataset is exhausted. Thus we have an accuracy figure for each day subsequent to the 100th day.

Where the protocols differ is whether and how they retrain the classifiers at the end of each day.

Static (S): In this protocol, classifiers are not re-trained. The classifiers that are built from the first 100 days of the training set are used unchanged to classify each of the remaining days' posts.

Supervised Incremental (SI): In this protocol, at the end of each simulated day (after the classifiers have made predictions for that day's posts), the classifiers are re-trained on a dataset that is extended by all of that day's posts with their *true labels*. In other words, when classifying posts submitted to the forum on day $i + 1$, the classifiers will have been trained on all posts up to day i. Re-training includes updating the priors for the Author Profile attribute so that they reflect the distribution of labels in the extended training set.

Semi-Supervised Incremental (SSI): This protocol is similar to the previous one except that the training sets are extended each day with that day's posts but with their *predicted labels*, rather than their true labels. In other words, when classifying posts submitted on day $i+1$, the classifiers have been trained on posts from the first 100 days with their true labels and posts from the next $i - 100$ days with their predicted labels.

Careful Semi-Supervised Incremental (CSSI): This protocol is similar to SSI, however not all posts are included when retraining. Only when the classifier's confidence in its prediction exceeds a threshold does that post and its predicted label get added to the training set. Posts from day i for which this threshold is not met are reclassified on day $i + 1$ along with the new posts from day $i+1$. Each of the classifiers can output a probability that its prediction is correct, and it is this that we compare with the threshold to decide what will happen with the post.

The SI protocol models the situation where forum moderators make decisions about each new post on a daily basis: a classifier might advise the moderator but the moderator always makes the final decision. Hence, the training set only ever contains posts with true labels. By contrast, the SSI protocol models the situation where there is no on-going moderator intervention: the classifier's decision

Protocol	Dataset	Unobserved attributes used
S	CNET	Previous Post from Same User
S	FIRE	Previous Post from Same User
SI	CNET	Previous Post, Full History, User History
SI	FIRE	Previous Post, Previous Post from Same User, User History
SSI	CNET	Previous Post, Full History, User History
SSI	FIRE	User History
CSSI	CNET	Previous Post, Full History, User History
CSSI	FIRE	Previous Post

Table 1: Unobserved relational attributes used by ICA in different experiments

is always final. Hence, training set posts up to day 100 have their true labels but subsequent posts have their predicted labels. The CSSI protocol attempts to reduce the misclassification rate of SSI by taking a more cautious approach.

It is informative to investigate the results of these somewhat extreme protocols. In practice, on a real forum performance will lie somewhere between the extremes. For example, there might be a triage system: if the classifier has low confidence in its decision, a moderator may be asked to review it.

One other aspect of the methodology should be mentioned. Before running these protocols, we use a model selection step. For the conventional kNN classifier, model selection chooses a good value for k (from $k = 5, 7, 11, 15$); for ICA using kNN as its local classifier, model selection chooses a good value for k and also decides which (non-empty) subset of the unobserved attributes to use; for CRF, model selection chooses between two optimization methods and parameters for optimization and regularization; and for the SVM, model selection chooses a penalty coefficient and a coefficient for the kernel (for which we use the Radial Basis Function). For all classifiers, model selection also chooses the best confidence threshold to use in the CSSI protocol.

Model selection uses the first 100 days' posts (with their true labels). We train different versions of the classifiers (e.g. kNN with different k) on approximately the first 70% of these days' posts, we measure accuracy of predictions on the remaining posts, and then we select the best version of the classifier for use in the protocols described above.

Table 1 shows, for each experiment (protocol plus dataset), which unobserved attributes model selection chooses for ICA.

5 Results

Figure 1 shows the results for the Static protocol — where there is no re-training. On the CNET dataset, ICA outperforms all other classifiers tested; in the case of the FIRE dataset, ICA performs best, but in this instance, its performance is matched by kNN.

By way of comparison, we calculated the accuracy of a majority-class classifier, taking the majority class from the 100 days of training data and seeing

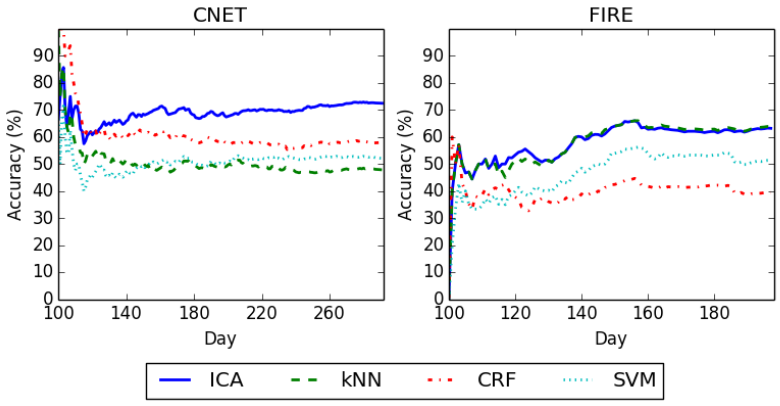

Fig. 1: Accuracy for the Static protocol

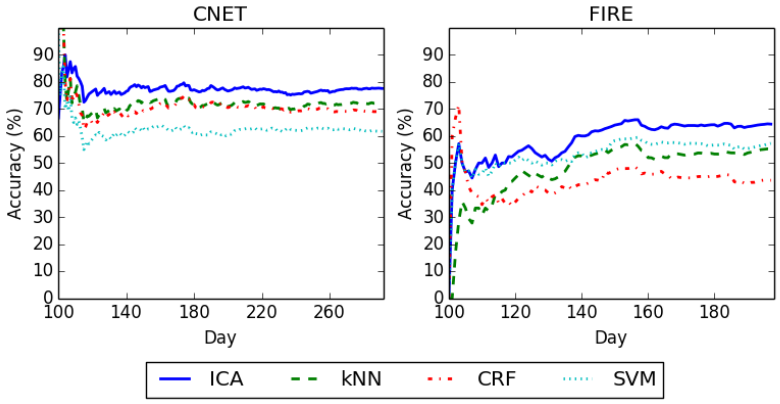

Fig. 2: Accuracy for the Supervised Incremental protocol

how often this would be the correct prediction for the remaining posts. The majority classifier has an accuracy of 40.74% for the CNET dataset and 36.87% for the FIRE dataset. ICA is about 70% and 60% accurate on these datasets respectively.

Figure 2 presents the results from experiments with the Supervised Incremental protocol — where daily re-training uses the true labels of the new posts. On the CNET dataset, ICA is again the best performing technique. However, with this protocol there is an increase in the accuracy of all the classifiers, and the difference in performance is smaller than that seen for the CNET dataset using the S protocol. This is to be expected as the classifiers have access to more properly labelled training data for each day. We can make similar observations about the FIRE dataset and this time kNN is not matching ICA.

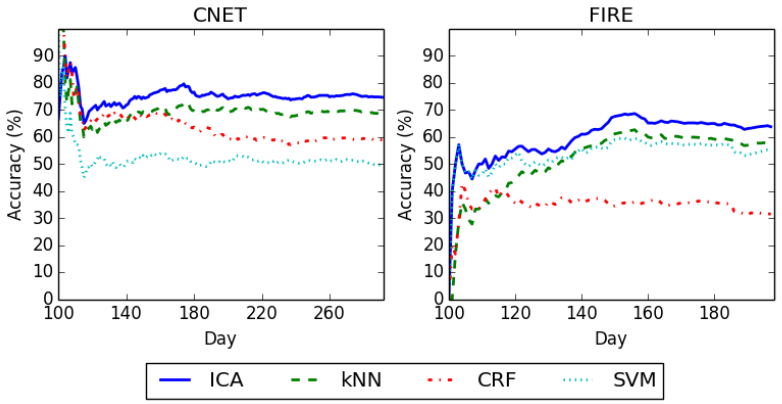

Fig. 3: Accuracy for the Semi-Supervised Incremental protocol

Figure 3 presents the results for the Semi-Supervised Incremental protocol — where daily re-training uses the predicted labels of the new posts. For the CNET dataset, ICA continues to achieve the best performance. In this case, kNN is also able to achieve good accuracy, but it still falls short of ICA's. The results for the FIRE dataset are similar, with ICA performing best, followed by kNN and, in this case, SVM close behind.

As we would expect, accuracies with this protocol (SSI) are lower than the figures we saw with the SI protocol. Although the training data grows over time, there is no guarantee that the labels in the new training instances are correct: they are predicted labels. It is also the case that, with the exception of kNN, the accuracies of the classifiers see little if any improvement over the accuracies obtained in the S protocol experiment, which implies that the benefits of the extra training data are largely outweighed by the error in the predicted labels.

Finally, the results for the Careful Semi-Supervised Incremental protocol are in Figure 4. Once again, ICA achieves the best results on the CNET dataset. The results for the other classifiers are a little improved relative to the SSI protocol experiment. The results for the FIRE set are more unusual. In this case, SVM performs very well initially. However towards the end, ICA, kNN and SVM all achieve similar accuracy.

Looking across the experiments, we see that ICA is generally the most accurate classifier for forum posts. We tested for significance using McNemar's Test [7] with a value of 1 for the continuity correction term [5] and $p = 0.05$. We did this on just the final day's test data. The tests confirm that ICA is significantly better than the other classifiers for all experiments except one: there is no significant difference between ICA and kNN on the FIRE dataset for the S protocol.

ICA is also the most robust classifier: variations in the experimental protocol have little effect on its accuracy. The other classifiers have room to improve their

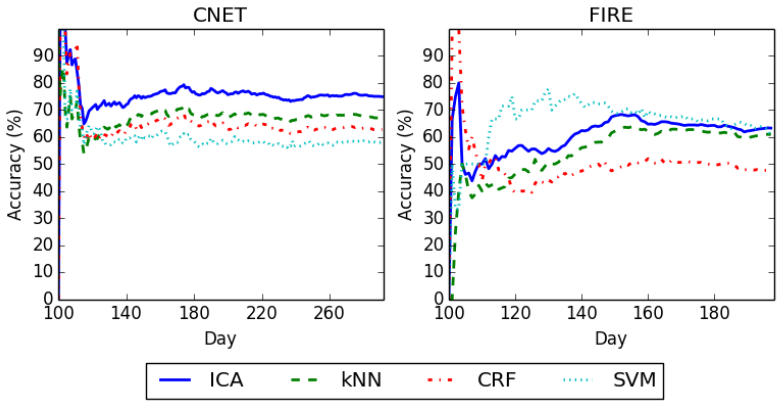

Fig. 4: Accuracy for the Careful Semi-Supervised Incremental protocol

CNET	ICA	kNN	CRF	SVM
S	**72.47**	47.82	57.83	52.09
SI	**77.57**	72.01	68.95	61.72
SSI	**75.81**	68.67	58.94	49.58
CSSI	**74.91**	67.01	62.76	57.97

FIRE	ICA	kNN	CRF	SVM
S	63.20	63.80	39.47	51.34
SI	**64.39**	55.19	43.62	57.27
SSI	**63.80**	57.27	31.45	54.90
CSSI	63.28	60.91	47.86	60.30

Table 2: Average accuracy across all test instances

accuracy when given more training data and, as we would expect, in general being given new training instances with true labels (SI) is better than being given new instances with confidently-predicted labels (CSSI) which, in turn, is better than being given new instances with predicted labels without regard to prediction confidence (SSI).

We have also summarized the results by computing average accuracy across *all* test instances: see Table 2. Again we tested for significance using McNemar's Test. In the Table, figures in bold are significantly better than all others on that protocol.

From the Table, we observe that, in nearly all cases, ICA is significantly better than the other classifiers. The exceptions are found on the FIRE dataset using the S protocol, where ICA is not significantly better than kNN, and on the FIRE dataset using the CSSI protocol, where there is no significant difference between ICA, kNN and SVM.

It is interesting too to note from the graphs and table the relatively poor performance of CRFs across our experiments. In [9], Kim et al. used a 10-fold cross-validation methodology to compare CRFs with a Maximum Entropy learner and a Structural SVM on their dataset (CNET). They found CRFs to have the highest accuracy. They did not compare with ICA. We repeated their experiment, using their methodology, to compare CRF with ICA: in this experiment, we found ICA and CRF accuracy to be very similar. However, we believe that Kim

et al.'s methodology lacks sufficient fidelity to the way that forum post classification happens in practice. Their test set contains randomly-chosen *threads*. This means that entire thread structure is preserved also in the training set. This enables the CRF to build a good model. For the more realistic chronological protocols that we have described in this paper, CRF loses its advantage and ICA becomes the most accurate.

We should add a word of warning about detailed comparisons of classifiers across different graphs in this paper. Although each graph plots accuracy for, e.g., a kNN classifier, the kNN classifier in one graph is not strictly comparable with the kNN classifier in another graph since they may be using different values for k. This is due to the model selection phase in our experimental methodology (described in Section 4). The same applies to all the other classifiers, e.g. ICA may be using a different value for k but also different unobserved relational attributes (Table 1).

6 Related Work

The most closely related piece of work is by Kim et al. [9], since we set out to compare collective classifiers that use local classifiers (ICA) with CRFs, which were the best-performing classifiers in their experiments. We also use their dataset and, as best we could, the same attributes that they used. Their paper also applies CRF to link classification, i.e. the task of predicting thread structure.

Bhatia et al. also report work on post classification [1], and we use the same dataset that they use (the FIRE dataset). However, they do not use collective classification: all their attributes are observed attributes. They experiment with several classifiers from the Weka machine learning tool-kit [6] and report the results of just the logit model classifier (72.02% accuracy).

There has been work on classification for online debates and spoken debates. For example, Somasundaran & Wiebe classify stance (pro or con) in online debates [16, 17]. In [2], transcripts from the US congress are classified based on support or opposition to a topic. This paper makes use of collective classification techniques.

A related topic is the classification of speech acts in email conversation. This is explored in [4] and then revisited in [3] using collective classification techniques. The collective classifiers were able to achieve better results.

More generally, there are several overviews and comparisons using collective classification, e.g. [15, 12]. The accuracy of the collective classifiers that use local classifiers can often be improved by using 'cautious' variants of the algorithms, which take account of the uncertainty in the predicted labels. These cautious collective classifiers are presented in, e.g., [14, 13].

Accordingly, we too used a cautious variant of ICA (the one that [14, 13] designates ICA_C) in our experiments. However, in our experimental protocols, we did not find ICA_C to be better than ICA. We omitted the results from this paper to reduce clutter in our graphs.

7 Conclusions

In this paper, we have investigated the task of classifying posts to Internet forums. We compared ICA —a collective classifier that uses a local classifier (in our case, kNN)— with Linear Chain CRFs —a classifier that optimizes global probability functions— and with some conventional classifiers (kNN and SVM). We compared them on two datasets, and we defined a number of experimental protocols for robust testing of these classifiers. ICA out-performed all other classification techniques tested across all the protocols for the CNET dataset. It out-performed all other classification techniques for all but two of the protocols for the FIRE dataset: it was matched by kNN in one protocol; and in another, there was a period during which SVM out-performed ICA, but their end-of-period accuracies were the same.

There are many ways in which we can improve ICA. We can try other attributes, including lexical attributes and also relational attributes that are based on later posts in the thread as well as those that are based on earlier posts in the thread. We can improve the kNN local classifier; for example, we can use attribute weights in the distance computation [12]. We can try other local classifiers, such as Naïve Bayes, or even ensembles of classifiers. Additionally, Gibbs Sampling is another collective classification technique which uses local classifiers. It is possible that this will further improve the accuracy seen in our experiments. However, Gibbs Sampling comes with a considerable time cost compared to ICA, and it remains to be seen whether any improvements would be worth the added expense. We can also trying collective classifiers that do not use local classifiers (other than CRF), such as Loopy Belief Propagation.

More generally, we plan to look at identifying argument structure in threads and then techniques that can be incorporated into the kind of forum management software that we described in Section 1.

References

1. Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. Classifying user messages for managing web forum data. In Zachary G. Ives and Yannis Velegrakis, editors, *Procs. of the 15th International Workshop on the Web and Databases*, pages 13–18, 2012.
2. Clinton Burfoot, Steven Bird, and Timothy Baldwin. Collective classification of congressional floor-debate transcripts. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Procs. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515. ACL, 2011.
3. Vitor R. Carvalho. On the collective classification of email speech acts. In Ricardo A. Baeza-Yates et al., editors, *Procs. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352. ACM Press, 2005.
4. William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. Learning to classify email into "speech acts". In *Procs. of the Conference on Empirical Methods in Natural Language Processing*, pages 309–316. ACL, 2004.

5. Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.
6. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.
7. Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms*. Cambridge University Press, 2011.
8. Thorsten Joachims, Thomas Finley, and Chun-Nam J. Yu. Cutting-plane training of structural SVMs. *Machine Learning Journal*, 77(1):27–59, 2009.
9. Su Nam Kim, Li Wang, and Timothy Baldwin. Tagging and linking web forum posts. In *Procs. of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202. ACL, 2010.
10. John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Procs. of the 18th International Conference on Machine Learning*, pages 282–289, 2001.
11. Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In Pat Langley, editor, *Procs. of the 17th International Conference on Machine Learning*, pages 591–598, 2000.
12. Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Case-based collective classification. In David Wilson and Geoff Sutcliffe, editors, *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference*, pages 399–404. AAAI Press, 2007.
13. Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious inference in collective classification. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 596–601. AAAI Press, 2007.
14. Luke McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious collective classification. *Journal of Machine Learning Research*, 10:2777–2836, 2009.
15. Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
16. Swapna Somasundaran and Janyce Wiebe. Recognizing stances in online debates. In Keh-Yih Su, Jian Su, and Janyce Wiebe, editors, *Procs. of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 226–234. ACL, 2009.
17. Swapna Somasundaran and Janyce Wiebe. Recognizing stances in ideological online debates. In *Procs. of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. ACL, 2010.
18. Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In Adnan Darwiche and Nir Friedman, editors, *Procs. of the 18th Annual Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.
19. Yair Weiss. Comparing the mean field method and belief propagation for approximate inference in MRFs. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods*. The MIT Press, 2001.
20. D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.