AN ABSTRACT OF THE THESIS OF

Ian Oberst for the degree of Master of Science in Computer Science presented on June 7, 2010.

Title:

On Feature Relevance Feedback Methods: Incorporating Labeled User Features

Abstract approved: _____

Weng-Keen Wong

In text classification, labeling features is often less time consuming than labeling entire documents. In situations where very little labeled training data is available, feature relevance feedback has the potential to dramatically increase classification performance. We review previous work on incorporating feature relevance feedback in the form of labeled features and introduce a new method, the *Feature Contrast Method*, for using feature relevance feedback with locally weighted logistic regression. We show that our method is responsive to user feedback and significantly outperforms previously developed feature relevance feedback methods while remaining robust to noise in feature and training data. We also highlight several key issues that affect the performance of feature feedback methods.

On Feature Relevance Feedback Methods: Incorporating Labeled User Features

by

Ian Oberst

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 7, 2010
Commencement June 2011

Master of Science thesis of Ian Oberst presented on June 7, 2010.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electric Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Ian Oberst, Author

ACKNOWLEDGEMENTS

I would like to thank Dr. Weng-Keen Wong for his guidance and wisdom as I traveled through the Wonderland that is graduate school. Despite my fumbling, he managed to instill no small amount of knowledge into my head and entertained my often wild and excited moments of inspiration.

My sincere thanks to Professor Paulson, whose encouragement, not only as a teacher but also as a friend, has made my journey through education more rewarding that I had ever thought possible. I firmly believe that he is one of the most valuable teachers Oregon State University has ever had.

Many thanks to Dr. Margaret Burnett and the ML Feedback team, who have been excellent research partners over the years and broke me into the world of feature feedback. Thanks for the many wonderful and interesting research projects.

My deepest gratitude to my family, who have always been there for me and helped support me emotionally and financially. Thanks to my father, who was always willing to "talk shop" with me whenever I had something I was excited about, and to my mother, who may not have always understood what I was

doing but was always there with unfailing support.

My appreciation to my brothers at Antioch who helped keep me grounded in my faith when everything else in life seemed to be pulling me away.

Finally, my deepest thanks to Dr. Jon Herlocker for hiring me as an untested undergraduate research assistant. You opened the first of many doors that I have walked through during my time at Oregon State University.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDIX TABLES

LIST OF APPENDIX TABLES (Continued)

LIST OF APPENDIX TABLES (Continued)

DEDICATION

To my Creator, with Whom nothing is impossible and to Whom I owe thanks for every talent and faculty that was brought to bear in producing this research.

To my wife, whose steadfast support and faith helped me through my doubts and moments of uncertainty. I love you.

To Matthew, Lane, and Andy: I wouldn't be who I am today without you three. I am truly the luckiest man alive to have friends like you.

Chapter 1 – Introduction

Applications tailored to the individual needs of users have the potential to dramatically increase productivity while reducing the investment of users who desire a particular end result. Such highly customized programs are difficult from a learning standpoint as each application must learn one user's distinct preferences, meaning that training data is unique to that one individual. Requiring each user to label large amounts of data is intractable from a usability standpoint, as it constitutes a huge time investment for each user, and such an investment could be greater than the time needed to perform a particular task manually.

Within such a customized framework, incorporating domain knowledge in a way that users can understand while minimizing the time investment lends itself to producing better performing applications that are successful in helping users complete their desired tasks. One method of incorporating domain information is through the use of feature relevance feedback, a method that allows users to denote certain features as important for a particular class or for the training data as a whole. In particular, we focus on feature relevance feedback in the form of labeled user features. A *labeled user feature* is a feature that is associated with a single label that represents the class that it is most representative of. For example, given the feature "touchdown," we might assign it the label "football."

Given time constraints and the option of feedback through just labeled instances

or using feature relevance in addition to labeled instances, feature relevance is more attractive as "on average humans take 5 times longer to label one document than to label one feature [10]." Additionally, previous work has shown that incorporating feature relevance feedback can improve the performance of classifiers in less time than focusing only on labeling documents alone [3]. Work by Raghavan & Allan introduced an algorithm for incorporating feature relevance feedback that has been well cited in related work [9]. In their study they showed that the use of feature relevance in the form of labeled features significantly increased the performance of their SVM-based methods in a series of 1-vs-All tests. While these results are encouraging, they leave several important questions regarding feature relevance unanswered:

- Does feature relevance work well outside of binary classification and, if so, how well does it scale with the number of classes?
- How does the quality of the initial labeled training points affect performance?
- Does the quality of features given as feedback affect the performance of feature relevance?

In this thesis we attempt to begin to answer these questions, focusing specifically on the effects of labeled feature feedback on the performance of feature relevance methods, apart from any form of active learning. Additionally, we contribute a new method of incorporating feature relevance feedback into locally weighted logistic regression (LWLR) [2]. LWLR is a modification of logistic regression that

weights training points with respect to a query through the use of a distance function. This allows training points that are more similar to the query to have a greater influence during classification than less similar points. Our method, the *Feature Contrast Method* (FCM), is a distance function that allows feature relevance feedback into LWLR.

To answer our research questions, we extended Raghavan & Allan's earlier work by analyzing their methods in a multi-class setting alongside our Feature Contrast Method. This idealized setting allowed us to more accurately assess the effect of feature feedback on the feature relevance methods we test. Finally, we address situations that are closer to how end-users behave by testing all methods on real user-feature feedback taken from our earlier Autocoder study [8].

Chapter 2 – Related Work

2.1 Introduction

Previous work on incorporating feature relevance feedback can be divided into two categories. The first category consists of methods that directly incorporate feature relevance into classifiers in a supervised setting. This means that the feature relevance feedback is used to modify the classifier within the context of only the training data. The second category incorporates feature feedback in a semi-supervised fashion, leveraging information from both the labeled training data and the unlabeled test data.

In this thesis we focus on the first category of feature feedback, which we denote as Supervised Feature Relevance feedback, as opposed to the second category, Semi-supervised Feature Relevance feedback, which we propose to explore in future work. In the following sections, we outline related work divided into these two categories.

2.2 Supervised Feature Relevance Methods

Haghighi & Klein introduced *prototype-driven learning*, a method that, given a set of "prototype" features for each label, optimizes the joint marginal likelihood of the labeled data [4]. This method allows for a simplified and declarative method

for incorporating feedback through labeled feature examples. In their paper, they show that this method of incorporating feature feedback resulted in a reduction in error for several of their induction tasks.

Tandem Learning, developed by Raghavan & Allan, incorporates both label feedback and feature feedback into SVMs to improve classifier performance [9]. In their paper, they show methods of incorporating user feedback via an active learning loop using three different feedback mechanisms. Using a combination of these methods in conjunction with uncertainty sampling and an oracle, Raghavan & Allan demonstrated a significant improvement in classifier performance over their baseline.

Raghavan & Allan's first method, scaling, involved directly modifying the counts of features based on the feature relevance feedback. For all instances in the labeled and unlabeled data, any features that were marked as relevant were scaled by some constant a, while all others were scaled by b. Their second method, feature pseudo documents, involved adding pseudo documents, consisting of only a feature and a label, to the training data. A parameter, r, is introduced which controls the distance of the pseudo documents to the margin. Values of $r \leq 1$ allow the pseudo documents to exert a greater influence on the margin than the support vectors. Raghavan & Allan's third method, pseudo relevance feedback, soft-labels the unlabeled training data by computing the similarity between the unlabeled instances and the set of terms the user has associated with each class. This is done by adding a new slack variable for each unlabeled instance. For each class, the sum of these slack variables is computed and each slack variable is weighted by the

similarity of its respective instance to the features associated with that class.

We note that Raghavan & Allan's third method falls under the category of Semi-supervised Feature Relevance methods, but we list it here as their previous two methods are supervised methods. As noted earlier, we wish to focus on the effect of feature relevance in a supervised setting, so we exclude their third method from our experiments.

2.3 Semi-supervised Feature Relevance Methods

Much previous work with Semi-supervised Feature Relevance methods involves soft-labeling the unlabeled data using feature feedback and then using these soft-labeled instances as additional training data. Early work by Liu, et al. [5] obtained labeled features using human annotators and then soft-labeled based on the cosine similarity between the unlabeled data and pseudo-instances that consisted only of labeled features. They showed that gathering feature relevance feedback required less effort than labeling instances, and classifiers trained using feature relevance feedback with a small training set produced results similar to classifiers trained using only a larger set of training data.

Wu & Srihari [14] provided soft-labeled instances with confidence scores obtained by computing the number of features, associated with some label l, that appear in an instance divided by the total number of features associated with the label l. These weighted instances were then used as training data for a Weighted Margin SVM, which considered the weights when selecting support vectors. They

demonstrated that incorporating feature relevance feedback in this fashion produced considerably better performance than SVMs using only labeled data.

Stumpf, et al. introduced a modification to co-training called *user co-training* [12]. Co-training is a semi-supervised learning method that utilizes two classifiers that work on the same data using independent sets of features with the assumption that each classifier produces the same classification despite the differing feature sets. User co-training treats the user as the second classifier involved in co-training. They do this by introducing a *user feedback classifier* that represents the user by treating the feature feedback as the set of features for the labels the user associates them with. In addition to user co-training, they also considered a constrained version of Naïve Bayes where user feedback was introduced as constraints for the maximum likelihood estimation of parameters during training. While the use of constrained Naïve Bayes did not increase performance, they showed that incorporating keyword-based feedback with user co-training resulted in a significant increase in performance.

Druck, Mann, & McCallum [3] used a generalized expectation (GE) criterion, a term in an objective function that incorporates preferences about model expectations, to train a discriminative probabilistic model where the labeled features directly constrained the model's predictions on the unlabeled data. To do this, they constructed a GE criterion that penalized the model's predicted distribution based on its KL-divergence from a reference distribution. The reference distribution was estimated from feature relevance feedback, obtained via an oracle or latent Dirichlet allocation (LDA), in conjunction with the unlabeled data. Druck,

Mann, & McCallum compared their method to the work of Wu & Srihari and Raghavan & Allan. In both cases, their comparisons suggest that their method of incorporating feature relevance through GE criteria outperforms these previously formulated methods.

GE is very general and can be applied to many different probabilistic models. Additionally, several types of parameter estimation can be shown to be special cases of GE, such as maximum likelihood estimation. Using this, it is possible to demonstrate that classifiers based on these methods are also special cases of GE, which we demonstrate is the case for LWLR in Section 4.4.

Chapter 3 – Nomenclature

Bold variables are used to represent vectors.

$\boldsymbol{Y}$	:	The set of all J labels.
$\boldsymbol{X}$	:	The set of all features, consisting of M features.
$\boldsymbol{D}$	:	The set of all training data, consisting of N instances.
$(\boldsymbol{x}_i, y_i)$	:	An instance, consisting of a non-sparse vector of M features and a label
x_i^j	:	The value of the jth feature in the ith instance.
$\boldsymbol{R}$	:	The feature relevance feedback for a single user, represented as a vector of M elements where the ith element is the label given as feedback for the ith feature in $\boldsymbol{X}$. Features for which no feedback is given are assigned a special symbol, $\emptyset$, which represents the absence of a label.
R_i	:	The ith label in $\boldsymbol{R}$.
$\boldsymbol{Q}$	:	The set of all queries.
$\boldsymbol{x}_q$	:	A query.
$\boldsymbol{x}_q^i$	:	The value of the ith feature in the query.
$\boldsymbol{\Theta}$	:	The set of parameters for a model.
$\boldsymbol{\Theta}(y_k)$	:	The set of parameters associated with the label y_k.

$\theta(y_k, i)$: The ith parameter from those associated with the label y_k.

$\cos(\boldsymbol{u}, \boldsymbol{v})$: A function that computes the cosine similarity between two instances.

$[P]$: Iverson bracket notation, defined as

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise.} \end{cases}$$

where P is a statement that can be true or false.

$\boldsymbol{u} \circ \boldsymbol{v}$: The Hadamard product of two vectors of identical length where $(\boldsymbol{u} \circ \boldsymbol{v})_i = \boldsymbol{u}_i \boldsymbol{v}_i$.

Chapter 4 – Locally Weighted Logistic Regression

4.1 Introduction

LWLR is a variation of logistic regression that assigns a weight to each training point with respect to a query, $\boldsymbol{x}_q$, through the use of a distance function. Using weighted training points allows LWLR to give more influence to points that are closer to the query than other points, which enables LWLR to outperform logistic regression in many cases. This advantage comes with a sacrifice in speed, as LWLR is a lazy algorithm that must compute these weights each time a query is made. In addition, an appropriate kernel value k must be selected to achieve good performance using LWLR.

The use of a distance function in calculating the weights for LWLR provides an excellent opportunity to incorporate feature relevance feedback in a natural way. The distance function only considers two points at a time, the query and a training point, which allows us modify the resulting weight by considering the feature feedback that is relevant to only these points. By doing so, we can obtain a more accurate picture of how related these two points are according to the end user.

We begin with a brief mathematical description of LWLR, followed by a derivation showing that LWLR is a special case of GE. Finally, we conclude with a dis-

cussion of the use of cosine similarity as a baseline distance function for LWLR and our feature relevance method.

4.2 Description of Locally Weighted Logistic Regression

We begin with logistic regression. Given a set of features derived from documents, $\boldsymbol{X}$, labels, $\boldsymbol{Y}$, and the set of parameters, $\boldsymbol{\Theta}$, the probability of a label y_k, given the input $\boldsymbol{x}_i$ is:

$$p_{\boldsymbol{\Theta}}(y_k|\boldsymbol{x}_i) = \frac{\exp\left\{\theta(y_k,0) + \sum_{j=1}^{M}\theta(y_k,j)\boldsymbol{x}_i^j\right\}}{Z(\boldsymbol{x}_i)},$$

where

$$Z(\boldsymbol{x}_i) = \sum_{k=1}^{J}\exp\left\{\theta(y_k,0) + \sum_{j=1}^{M}\theta(y_k,j)\boldsymbol{x}_i^j\right\}.$$

For simplicity in later derivations, we introduce a new feature into each instance, denoted as $\boldsymbol{x}_i^0$, which allows us to include the parameter $\theta(y_k,0)$ in the summation. This feature always has a value of 1. Incorporating this, we get

$$p_{\boldsymbol{\Theta}}(y_k|\boldsymbol{x}_i) = \frac{\exp\left\{\sum_{j=0}^{M}\theta(y_k,j)\boldsymbol{x}_i^j\right\}}{Z(\boldsymbol{x}_i)},$$

where

$$Z(\boldsymbol{x}_i) = \sum_{k=1}^{J}\exp\left\{\sum_{j=0}^{M}\theta(y_k,j)\boldsymbol{x}_i^j\right\}.$$

This gives us the following likelihood for logistic regression:

$$L(\boldsymbol{\Theta}) = \prod_{i=1}^{N} p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)$$

LWLR modifies the likelihood by adding a weight term, $w(\boldsymbol{x}_q, \boldsymbol{x}_i)$, which is the weight for the ith training point with respect to $\boldsymbol{x}_q$. Incorporating this we get

$$L(\boldsymbol{\Theta}) = \prod_{i=1}^{N} p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)^{w(\boldsymbol{x}_q, \boldsymbol{x}_i)}, \tag{4.2.1}$$

where

$$w(\boldsymbol{x}_q, \boldsymbol{x}_i) = \exp\left\{-\frac{\boldsymbol{f}(\boldsymbol{x}_q, \boldsymbol{x}_i)^2}{k^2}\right\},$$

k is the kernel width, and F is a function of the Euclidean distance from the ith data point to the query.

4.3 Training Locally Weighted Logistic Regression

LWLR sets the weight for each training instance with respect to the query, which is the current instance we wish to classify. As a result, LWLR must be trained for each query we wish to classify. To do this, we perform maximum likelihood estimation on the parameters $\boldsymbol{\Theta}$ using the log-likelihood of LWLR, which is

$$l_w(\boldsymbol{\Theta}) = \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i). \tag{4.3.1}$$

If we consider a single parameter, $\theta(y_a, b)$, the derivative of the log-likelihood with respect to $\theta(y_a, b)$, given by l'_w, is:

$$
\begin{aligned}
l'_w = & \frac{\delta}{\delta\theta(y_a,b)} \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\frac{\delta}{\delta\theta(y_a,b)} \log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i) \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\frac{\delta}{\delta\theta(y_a,b)} \log \left(\frac{\exp\left\{\sum_{j=0}^{M} \theta(y_i,j)\boldsymbol{x}_i^j\right\}}{Z(\boldsymbol{x}_i)} \right) \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\frac{\delta}{\delta\theta(y_a,b)} \left(\sum_{j=0}^{M} \theta(y_i,j)\boldsymbol{x}_i^j - \log Z(\boldsymbol{x}_i) \right) \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\frac{\delta}{\delta\theta(y_a,b)} \sum_{j=0}^{M} \theta(y_i,j)\boldsymbol{x}_i^j - \frac{\delta}{\delta\theta(y_a,b)} \log Z(\boldsymbol{x}_i) \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\boldsymbol{x}_i^b[y_a = y_i] - \frac{1}{Z(\boldsymbol{x}_i)} \frac{\delta}{\delta\theta(y_a,b)} Z(\boldsymbol{x}_i) \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\boldsymbol{x}_i^b[y_a = y_i] - \frac{1}{Z(\boldsymbol{x}_i)} \frac{\delta}{\delta\theta(y_a,b)} \sum_{k=1}^{J} \exp\left\{ \sum_{j=0}^{M} \theta(y_k,j)\boldsymbol{x}_i^j \right\} \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\boldsymbol{x}_i^b[y_a = y_i] - \frac{1}{Z(\boldsymbol{x}_i)} \sum_{k=1}^{J} \frac{\delta}{\delta\theta(y_a,b)} \exp\left\{ \sum_{j=0}^{M} \theta(y_k,j)\boldsymbol{x}_i^j \right\} \right) \\
= & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\boldsymbol{x}_i^b[y_a = y_i] - \frac{1}{Z(\boldsymbol{x}_i)} \right. \\
& \left. * \sum_{k=1}^{J} \left(\exp\left\{ \sum_{j=0}^{M} \theta(y_k,j)\boldsymbol{x}_i^j \right\} \frac{\delta}{\delta\theta(y_a,b)} \sum_{j=0}^{M} \theta(y_k,j)\boldsymbol{x}_i^j \right) \right)
\end{aligned}
$$

$$= \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \left(\boldsymbol{x}_i^b[y_a = y_i] - \frac{1}{Z(\boldsymbol{x}_i)} * \sum_{k=1}^{J} \left(\exp \left\{ \sum_{j=0}^{M} \theta(y_k, j) \boldsymbol{x}_i^j \right\} \boldsymbol{x}_i^b[y_a = y_k] \right) \right)$$

Using the derivative of l_w with respect to $\theta(y_a, b)$, we can construct the full derivative of l_w with respect to $\boldsymbol{\Theta}$. We then train a LWLR model using an optimization routine like L-BFGS [6] that finds the values of $\boldsymbol{\Theta}$ that maximize the log likelihood.

4.4 Locally Weighted Logistic Regression as a Special Case of Generalized Expectation

In Mann & McCallum's work, a GE criterion is a function, represented as G, which expresses some preference about a model's expectations of some set of values [7]. It represents the model's predicted distribution over the values of some function, f, as a scalar, where f is a function of the model's variables. Formally, this is denoted as

$$G(E[f(X)]) \rightarrow \mathbb{R}$$

The scalar value produced by G is added as a term to the objective function used for parameter estimation.

It has been shown that maximum likelihood is a special case of generalized expectation [7]. We extend that derivation to show that LWLR is also a special case of GE. We begin with the likelihood for LWLR, shown earlier in equation 4.3.1.

Let $p_{\boldsymbol{\Theta}}(Y|X)$ be the conditional probability distribution over Y given X parameterized by $\boldsymbol{\Theta}$. Let $\boldsymbol{f}(\boldsymbol{x}_i, y)$ be a vector indicator function that returns a vector of length N containing zeros in all locations except at the ith position where it contains a 1 if $\boldsymbol{x}_i$ has label y. Let $\hat{\boldsymbol{f}}$ be the empirical distribution of the vector indicator function applied to X in the training data $\boldsymbol{D}$ and Y in the labels $\boldsymbol{Y}$, denoted as

$$\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in \boldsymbol{Y}} \boldsymbol{f}(\boldsymbol{x}_i, y),$$

which returns a vector of N values in which the ith element represents the probability of selecting the ith data point from the training data.

To represent LWLR, our GE criterion, G, is the negative cross entropy between the elements of $E_{\boldsymbol{\Theta}}[\boldsymbol{f}(X, Y)|\boldsymbol{X}]$ and $\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})$, multiplied by the weight $w(\boldsymbol{x}_q, \boldsymbol{x}_i)$, where

$$E_{\boldsymbol{\Theta}}[\boldsymbol{f}(X, Y)|\boldsymbol{X}] = \frac{1}{N} \sum_{i=1}^{N} \sum_{y \in \boldsymbol{Y}} p_{\boldsymbol{\Theta}}(y|\boldsymbol{x}_i) \boldsymbol{f}(\boldsymbol{x}_i, y)$$

which is the model's expectation of the vector indicator function given the training data. Given this and $\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})$, we get

$$G = \quad -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})_i \log(E_{\boldsymbol{\Theta}}[\boldsymbol{f}(X, Y)|\boldsymbol{X}]_i) \tag{4.4.1}$$

where the subscripts i index into the dimensions of the vector indicator function. Taking this, we simplify until we reach something equivalent to equation 4.3.1,

which is the log-likelihood of LWLR.

$$\begin{aligned} G = & -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})_i \log(E_{\boldsymbol{\Theta}}[\boldsymbol{f}(X, Y)|\boldsymbol{X}]_i) \\ = & -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})_i \log\left(\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{y\in\boldsymbol{Y}} p_{\boldsymbol{\Theta}}(y|\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}, y)\right)_i\right) \end{aligned}$$

From the expansion of $E_{\boldsymbol{\Theta}}[\boldsymbol{f}(X, Y)|\boldsymbol{X}]$ we see that it returns a vector of probabilities, multiplied by $\frac{1}{N}$, where the ith entry corresponds to $\frac{1}{N}p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)$. This gives:

$$\begin{aligned} G = & -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\hat{\boldsymbol{f}}(\boldsymbol{X}, \boldsymbol{Y})_i \log\left(\frac{1}{N}p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) \\ = & -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\left(\frac{1}{N}\sum_{j=1}^{N}\sum_{y\in\boldsymbol{Y}}\boldsymbol{f}(\boldsymbol{x}_j, y)\right)_i \log\left(\frac{1}{N}p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) \\ = & -\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\frac{1}{N}\log\left(\frac{1}{N}p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) \\ = & -\frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log\left(\frac{1}{N}p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) \\ = & -\frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\left(\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i) - \log N\right) \\ = & -\frac{1}{N}\sum_{i=1}^{N}\left(w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i) - w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log N\right) \\ = & -\frac{1}{N}\sum_{i=1}^{N}\left(w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) + \frac{1}{N}\sum_{i=1}^{N}\left(w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log N\right) \\ = & -\frac{1}{N}\sum_{i=1}^{N}\left(w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) + \frac{\log N}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \end{aligned} \tag{4.4.2}$$

Both $\frac{1}{N}$ and $\frac{\log N}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)$ will not affect the outcome of the maximum likelihood estimation for the parameters $\boldsymbol{\Theta}$. Removing them from equation 4.4.2, we get

$$\begin{aligned} G = & -\frac{1}{N}\sum_{i=1}^{N}\left(w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)\right) + \frac{\log N}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i) \\ = & \sum_{i=1}^{N} w(\boldsymbol{x}_q, \boldsymbol{x}_i)\log p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i) \end{aligned}$$

which is the log-likelihood for LWLR. This shows that equation 4.4.1 is equivalent to equation 4.3.1 in terms of the maximum likelihood estimation for the parameters $\boldsymbol{\Theta}$. Thus, LWLR is a special case of generalized expectation.

4.5 Cosine Similarity

The LWLR classifier uses a distance function to determine the weights of the training data with respect to a query. The closer a particular training point is to the query, the more weight it is given during training. Cosine similarity (COSIM) is a method used for comparing two vectors by finding the cosine angle between them. In text classification it is used to compare the similarity of two documents by using the vector of TF-IDF values associated with each document's features. Since cosine similarity is used frequently in text classification problems and does not incorporate any feature feedback, it lends itself to being used as the baseline distance function for LWLR.

The distance function used by LWLR requires that higher distance values indi-

cate greater dissimilarity between the query and the training point. In its original form, cosine similarity behaves in the opposite fashion, where 1 indicates that two points are identical and 0 indicates that they are independent. We can incorporate it into a distance function using by defining the following function:

$$COSIM(\boldsymbol{x}_q, \boldsymbol{x}) = 1 - \cos(\boldsymbol{x}_q, \boldsymbol{x}) \tag{4.5.1}$$

This function inverts the output of cosine similarity, giving results ranging from 0 (identical) to 1 (independent). Note that using cosine similarity as a distance function restrict points from being any further apart than a distance of 1.

Chapter 5 – Feature Relevance Feedback in Locally Weighted Logistic Regression: Feature Contrast Method

5.1 Introduction

Locally weighted logistic regression compares a query to training data to set the weight of each training point through the use of a distance function. Training points that are weighted more highly have a greater effect on classification than those with lower weights. The particular distance metric used directly affects the resulting weights. Our baseline distance metric, cosine similarity, only shows how similar two instances are and ignores other information that may indicate that the two instances belong to the same class, even if they do not appear to be similar.

We introduce a new distance metric called the Feature Contrast Method (FCM). In addition to using cosine similarity to consider how similar two instances are, FCM utilizes the labels assigned to user features that are present in the instances. By comparing and contrasting user features and their labels, FCM is able to better determine if two instances belong to the same class. We begin with a formal definition of FCM, followed by a detailed discussion of its construction.

5.2 Formal Definition

We begin with the likelihood for LWLR and modify the distance function to include the feature relevance feedback, which gives

$$L(\boldsymbol{\Theta}) = \prod_{i}^{N} p_{\boldsymbol{\Theta}}(y_i|\boldsymbol{x}_i)^{w(\boldsymbol{x}_q,\boldsymbol{x}_i,\boldsymbol{R})}$$

where $w(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{R})$ is the weight for instance $\boldsymbol{x}_i$ with respect to the query $\boldsymbol{x}_q$ and the feature relevance feedback. This weight is defined as

$$w(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{R}) = \exp\left\{-\frac{F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})^2}{k^2}\right\}$$

where k is the kernel width. Let $F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})$ be

$$F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = \max\left[0, COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i) * (1 - G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}))\right],$$

where $G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})$ is

$$G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) - \frac{PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})}{J-1}.$$

Let $BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})$ be

$$BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = W(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})$$

and let $PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i)$ be

$$PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = W(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}).$$

$W(\boldsymbol{x}_q, \boldsymbol{x}, \boldsymbol{\mathcal{S}}, \boldsymbol{R})$ is a function that computes the cosine similarity between $\boldsymbol{x}_q$ and $\boldsymbol{x}$ using only those two points and only the features whose corresponding labels in $\boldsymbol{R}$ exist in the set of labels $\boldsymbol{\mathcal{S}}$.

Let $\boldsymbol{h}(\boldsymbol{R}, \boldsymbol{\mathcal{S}})$ be a vector indicator function that returns a vector of length M where the ith entry is 1 if $R_i \in \boldsymbol{\mathcal{S}}$ and $R_i \neq \emptyset$, and 0 otherwise. Let $\hat{\boldsymbol{h}}(\boldsymbol{x}_i)$ be a vector indicator function that returns a vector of length M where the jth entry is 1 if $\boldsymbol{x}_i^j = 0$ and 0 otherwise. Given $\boldsymbol{h}$ and $\hat{\boldsymbol{h}}$, C is defined as

$$C(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{\mathcal{S}}, \boldsymbol{R}) = \alpha\left(\boldsymbol{h}(\boldsymbol{R}, \boldsymbol{\mathcal{S}}) \circ \boldsymbol{x}_q\right) \bullet \hat{\boldsymbol{h}}(\boldsymbol{x}_i),$$

where α is the normalization constant for the vector produced by $\boldsymbol{h}(\boldsymbol{R}, \boldsymbol{\mathcal{S}}) \circ \boldsymbol{x}_q$. $\alpha\left(\boldsymbol{h}(\boldsymbol{R}, \boldsymbol{\mathcal{S}}) \circ \boldsymbol{x}_q\right)$ results in a vector of normalized feature values where only those features in $\boldsymbol{x}_q$ that have feature relevance feedback whose label in $\boldsymbol{R}$ is also in $\boldsymbol{\mathcal{S}}$ have non-zero values. The vector created from $\hat{\boldsymbol{h}}(\boldsymbol{x}_i)$ is used as a mask through which we pass the normalized feature values so that only those features with zero values in the $\boldsymbol{x}_i$ are considered. The dot product with $\hat{\boldsymbol{h}}(\boldsymbol{x}_i)$ results in C producing the sum of the normalized feature values in $\alpha\left(\boldsymbol{h}(\boldsymbol{R}, \boldsymbol{\mathcal{S}}) \circ \boldsymbol{x}_q\right)$ for only those features that have non-zero feature values in $\boldsymbol{x}_i$.

5.3 Creating a Distance Function that Incorporates Feature Relevance Feedback

We begin with equation 4.5.1, the COSIM function, which does not take into account any information from the user features, as the basis for our distance function, Using this, our distance function is

$$F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i).$$

Now suppose we have a set of features, $\boldsymbol{R}$, which we consider to be more important than other features in determining the label of a data point. In order to incorporate information from the user features in $\boldsymbol{R}$, we need a function that returns a value that represents the contribution of those features to deciding if the training point in question is similar to $\boldsymbol{x}_q$. For now, we call this function $G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})$ and leave it undefined.

We note that if two points are very similar to each other via cosine similarity, using the additional information provided by G affects the resulting weight very little, as cosine similarity is already returning a value very close to 0. If the two points are not similar, then it may be that they do not belong in the same class, or they do belong together but they simply do not share many similar features. In the second case, using the additional information from G may help us determine whether or not the two data points actually share a label. Since it is only useful to include this information when the points are not similar, we need to make sure

that it is only included when the cosine similarity is close to one. We can do this by modifying the distance function as follows:

$$\begin{aligned} F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = \quad & COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i) - COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i) * G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) \\ = \quad & COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i) * (1 - G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})) \end{aligned}$$

This weights the contribution of $1 - G$ by the cosine similarity of $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$, using more information from G as the similarity of $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ decreases.

5.3.1 Defining G

Cosine similarity allows us to consider all the feature to see if $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ are similar. If a lot of the information represented by G is being used, this may indicate that leveraging information other than that of just the unlabeled user features is necessary, and using the feature relevance information may help us in deciding if $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ are more or less similar than what cosine similarity indicated. When thinking about how to use this information, some questions we might consider are:

- Do the instances share any user features?
- Does the query have any user features whose class is the same as that of the training point?
- Does the query have any user features whose class is different from that of the training point?

If the two points share many user features, then this may indicate that they are similar. Even if they do not share many user features, they still may belong to the same class if the query contains many user features whose class is the same as the training point. Since we have already considered $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ in terms how similar they are, we now look at how they are different.

An instance containing user features is not guaranteed to contain only those user features associated with the label that the instance should have. This means that the instance may contain user features whose associated label contradicts the correct label for it. When considering the query, we are interested in discovering if the label associated with the training point is the correct label for the query. In terms of the user features in the query, those that do not share a label with the training point indicate that the query may not be similar to the training instance, while those user features that do share a label with the training data indicate the opposite.

There are two types of user features we can consider, *positive user features*, or those that share a label with the training instance, and *negative user features*, those that do not. We need a method of representing the contribution from both types, and one way of doing so is through the use of feature values.

Intuitively, a feature that shows up more in an instance is more important than one that shows up fewer times. This may not be true for very common words, but we assume that for user features the user chooses features that represent only those that are useful in determining the class of an instance. We incorporate this into G

as

$$G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = C(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) - C(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}).$$

The function C returns the sum of the normalized feature values for all user features with non-zero values in $\boldsymbol{x}_q$, but zero values in $\boldsymbol{x}_i$, that are associated with a label that exists in the set of labels passed as an argument. In this case, the first use of C returns the sum of the normalized positive user feature values and the second use returns the normalized negative user feature values. We normalize the values to prevent a bias towards instances with a greater number of user features.

G now returns a value that represents the information from the feature relevance feedback in terms of how $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ are different. While useful, this neglects to consider how they are similar in terms of user features. This is especially important if they are completely similar in terms of the user features, as this would mean we could get no information from looking at how they were different.

In order to balance between the information from contrasting the instance in terms of the user features versus the information from comparing them, we weight each use of C. To accomplish this, we use the previously defined function W to modify the distance function as follows:

$$\begin{aligned} G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = \quad & W(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) \\ & -W(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}). \end{aligned}$$

W represents a modified version of cosine similarity where the similarity is computed from the TF-IDF values for $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ calculated using only those two

points rather than the entire corpus. Additionally, W only takes into consideration the user features when computing the cosine similarity. This allows us to use less of the information from C when $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ are more similar in terms of the user features, and visa versa.

The two products in G accomplish two different functions. One product represents information that increases the similarity between $\boldsymbol{x}_q$ and $\boldsymbol{x}_i$ while the other product decreases the similarity. For ease in referring to these two products, we denote them as follows:

$$\begin{aligned} BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = &\quad W(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) \\ PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = &\quad W(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}) * C(\boldsymbol{x}_q, \boldsymbol{x}_i, \boldsymbol{Y} \setminus y_i, \boldsymbol{R}) \end{aligned}$$

Using these, we get

$$G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) - PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}).$$

We note that if our data has more than two classes the $PENALTY$ term will represent more than a single label, meaning that it will carry more weight than the $BONUS$ term in these cases. To prevent this, we divide the $PENALTY$ term by $J - 1$, which averages its contribution over the number of labels it represents. Using this, we arrive at our final representation of G:

$$G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = BONUS(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) - \frac{PENALTY(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R})}{J - 1}.$$

5.3.2 Incorporating G

In its current form, G will return a value that modifies the similarity between the two points in question. In the case of a negative value, which will push the points further apart, there is no restriction on how far apart the points can be. A positive value is different, as the closest two points can be is 0. Too large of a positive value will cause the distance to become negative, which will artificially push the points apart. In order to prevent this, we modify our distance metric in the following way:

$$F(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}) = \max\left[0, COSIM(\boldsymbol{x}_q, \boldsymbol{x}_i) * (1 - G(\boldsymbol{x}_q, \boldsymbol{x}_i, y_i, \boldsymbol{R}))\right]$$

This prevents G from causing the points from being "more similar" than 0.

Chapter 6 – Feature Relevance in a Multi-Class Experiment

6.1 Introduction

Raghavan & Allan's original experiment was performed in a 1-vs-All setting using feature relevance feedback selected by an oracle. The oracle selected the features to be given as feedback by choosing those features whose information gain was greater than or equal to the average information gain of the top thirty features from the corpus. After the features were chosen, each feature was assigned the label that had the highest probability of occurring with it. The 1-vs-All experiment environment, along with their method of selecting the feature relevance feedback, made it difficult to answer our research questions using their original setup. The large number of classes in each of the datasets causes a 1-vs-All framework to have inherent class imbalance, which can affect the performance of classifiers and make determining the benefits of various feature relevance methods difficult. Additionally, the selection of the user feedback, as it was not balanced between classes, caused some classes to lack feedback, making it difficult to assess the effects of features on feature relevance methods.

Extending their previous work into a multi-class environment not only made it possible to remove issues of class imbalance, but also allowed us to analyze their feature relevance methods in a more realistic, and difficult, test setting. A more

realistic setting enabled us to assess the usefulness of each method we tested in terms of problems that are more similar to those that a real user might encounter. To answer our research questions, we constructed a multi-class experiment with two different sampling methods for each dataset we use. This allowed us to answer our question on the effect of the initial training data. We then selected a balanced number of top features to represent the feature relevance feedback for each class in the experiments, as well as a random selection of features. Comparing the results from both sets of features enabled us to answer our question regarding the effects of feature quality. Finally, we varied the number of features we gave as feature relevance feedback for each class to determine how increasing the amount of feedback affected each method.

In this section we explain in detail the setup of our multi-class experiment and then show the results of using both Raghavan & Allan's supervised methods in addition to our FCM method. We discuss the effects that each sampling method had on the training data with respect to the distribution of user features. Finally, we conclude with a discussion of the results with respect to our research questions.

6.2 Experiment

We began by choosing 5 classes from both the 20 Newsgroups and Modapte datasets. Using these 5 classes, we created 4 sub-datasets of each original dataset, each sub-dataset consisting of n classes, where n ranged from 2 to 5. Using each sub-dataset, we created 30 pools consisting of 1000 random instances, sampled

so that each class had as close to an equal number of instances in the pool as possible. After each pool was created, we then created a corresponding validation set consisting of 50 labeled instances per class represented in the pool. From the pools we selected 4 instances from each class to be training data, and the remaining instances were used as test data. For clarity, we refer to these training/testing sets as the Balanced/Random (BR) datasets, respectively, for either the 20 Newsgroups (BR20News) or Modapte (BRModapte) corpus.

In addition to the BR sets, we used an oracle to create a list of user features for each sub-dataset. For each class in a particular sub-dataset, the oracle selected the top m features for that class according to information gain. Labels were associated with the features by computing the probability of occurrence of the feature with the labels in the sub-dataset. This was done for values of m from 1 to 10, creating 10 sets of the "best" user features, as selected by the oracle. We also created 10 sets of randomly chosen features, each set a duplicate of the previous with a randomly selected feature added for each class in the sub-dataset.

Raghavan & Allan's SVM methods were modified to work in a multi-class setting. Using the pools of 1000 training instances, we ran methods 1 and 2 using uncertainty sampling and the 10 sets of "best" user features. For each run, we saved the training and testing splits generated by uncertainty sampling on each of the pools in conjunction with the list of user features. The data from the method (either 1 or 2) that performed the best was then used to create new training/testing datasets for use with our methods. For clarity, we refer to these training/testing sets as the Uncertainty (UN) datasets for either 20 Newsgroups (UN20News) or

Modapte (UNModapte). This process is illustrated in Figure 6.1.

Table 6.1: Parameters for each dataset, determined using validation sets.

	20 Newsgroups		**Modapte**	
	Uncertainty	BR	Uncertainty	BR
Method 1	$a = 15$, $b = 1$	$a = 3$, $b = 1$	$a = 4$, $b = 1$	$a = 2$, $b = 1$
Method 2	$r = .4$	$r = 3$	$r = .6$	$r = 2$
FCM	$k^2 = .1$	$k^2 = .2$	$k^2 = .1$	$k^2 = .3$
COSSIM	$k^2 = .1$	$k^2 = .15$	$k^2 = .2$	$k^2 = .35$

We ran LWLR using our FCM distance metric on each of the Balanced/Random datasets and the Uncertain datasets. In addition, we modified Raghavan & Allan's methods to use the training/testing splits in the Balanced/Random datasets rather than uncertain sampling, and ran methods 1 and 2 on Balanced/Random datasets. The parameters for each classifier/dataset pair were tuned using the lists of "best" user features, given as feature relevance feedback, and the validation sets. These parameters are shown in Table 6.1

We computed the average macro-F1 and micro-F1 scores over all tests for a particular method/dataset along with a confidence interval, which was computed over the set of average macro-/micro-F1 scores used to calculate the overall average. We focus on the macro-F1 scores, but micro-F1 scores are available in Appendix A.

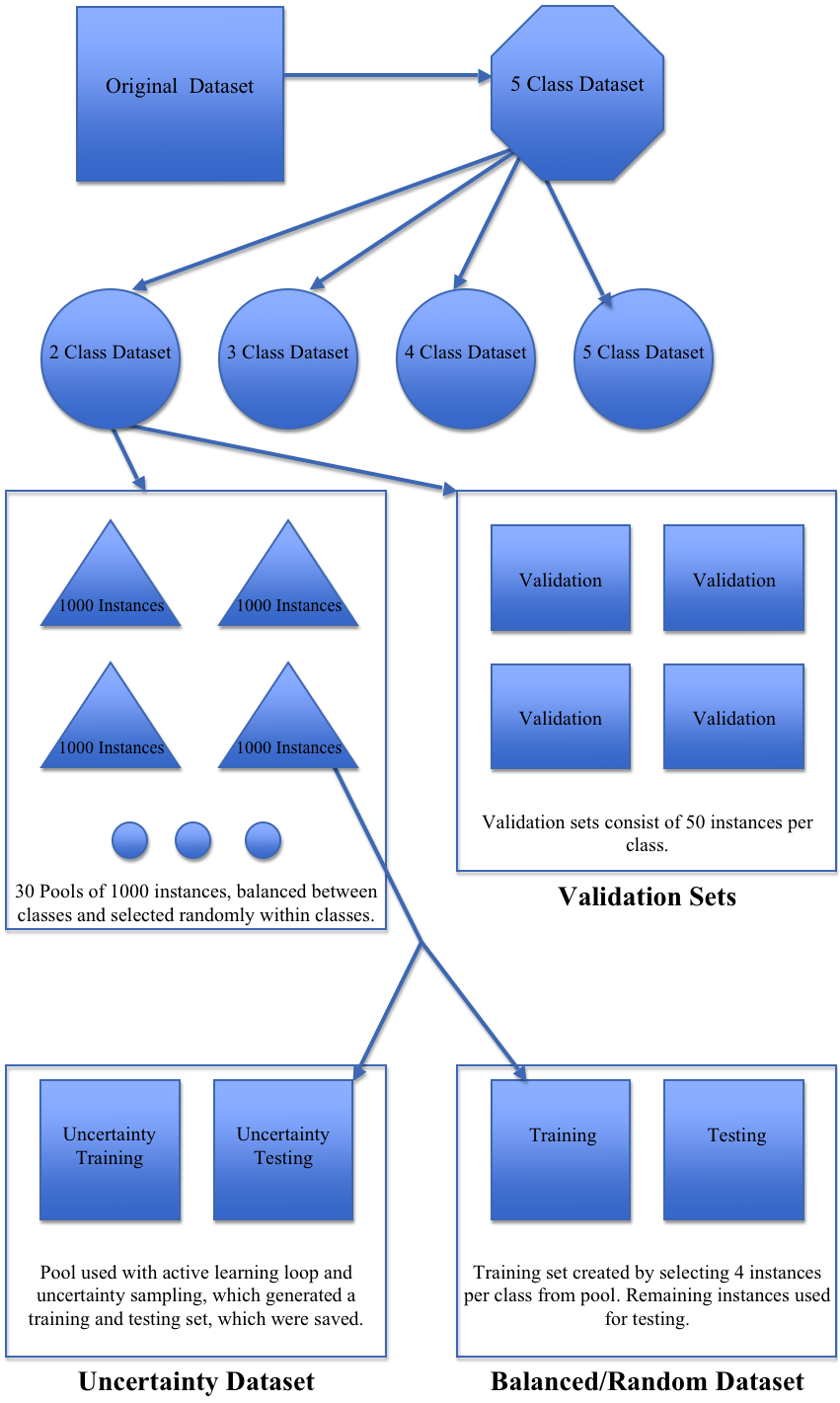

Figure 6.1: An illustration showing the creation of the validation and datasets for the multi-class experiments.

6.3 Results

6.3.1 Balanced/Random Datasets

We begin by looking at the performance of classifiers on the 20 Newsgroups Balanced/Random dataset, which is shown in Figures 6.2 - 6.5. For all class sizes, LWLR with FCM, using the best user features, performs significantly better than both its baseline and Raghavan & Allan's methods. It also shows a responsiveness to the addition of user features. Use of random features instead of the best features cause the classifier to perform with no significant difference from the baseline.

Both of Raghavan & Allan's methods show no significant advantage of one over the other, though method 1 is typically more significant than the baseline. Except for the 3 - and 5 - class tests using 4 or more of the best user features, method 2 fails to perform significantly better than a standard SVM. Method 2 shows responsiveness to the increase in the number of best user features, albeit only slightly. Using random user features, both methods show no significant difference from the baseline.

Looking at the performance of the classifiers on the Modapte Balanced/Random dataset, shown in Figures 6.6 - 6.9, we note that once again LWLR with FCM tends to significantly outperform the baseline as well as Raghavan & Allan's methods. For the 2 - class tests, both methods 1 and 2 show no significant difference compared to LWLR with FCM. However, FCM retains its performance much better as the number of classes increase, significantly outperforming all other methods as the number of best user features increases. Again, FCM shows a responsiveness to

increasing the number of best user features.

With the Modapte Balanced/Random dataset, method 2 starts to perform significantly better than a plain SVM as the number of classes and best user features increase. It also shows more responsiveness to increasing the number of best user features as the number of classes increase. Method 1, using the best user features, shows no significant improvement over using random user features. As before, all classifiers' performance using random features shows no significant difference from their respective baselines.

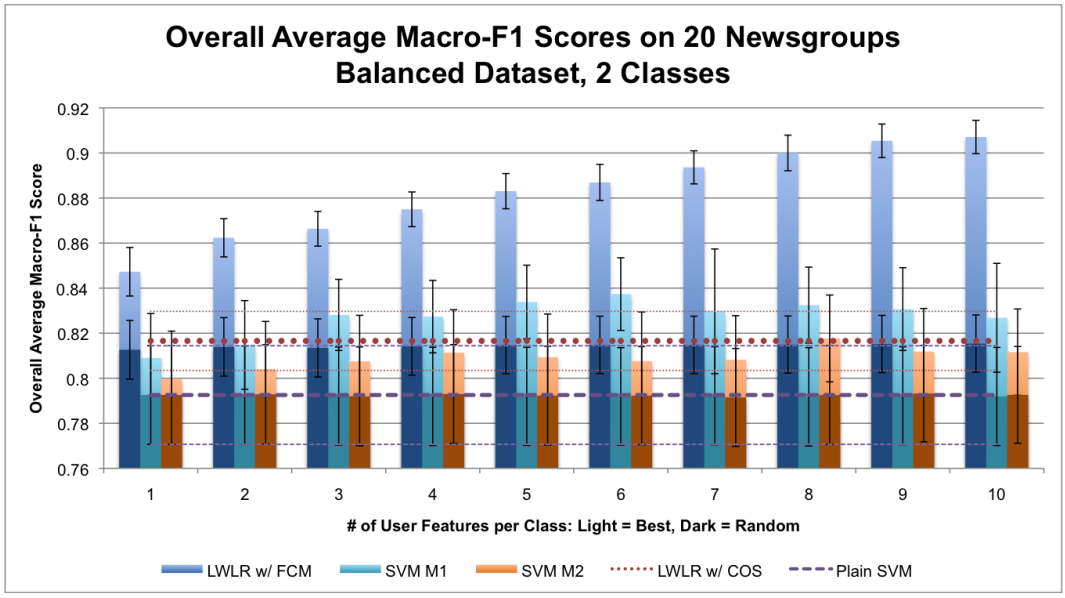

Figure 6.2: The overall average macro-F1 scores on the 20 Newsgroups Balanced/Random dataset, 2 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

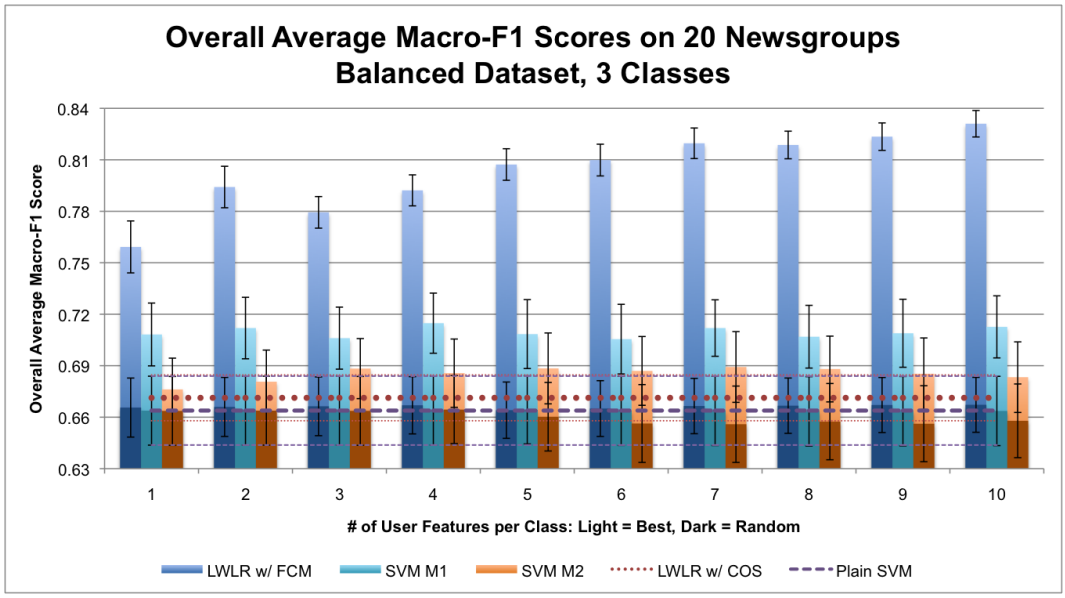

Figure 6.3: The overall average macro-F1 scores on the 20 Newsgroups Balanced/Random dataset, 3 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

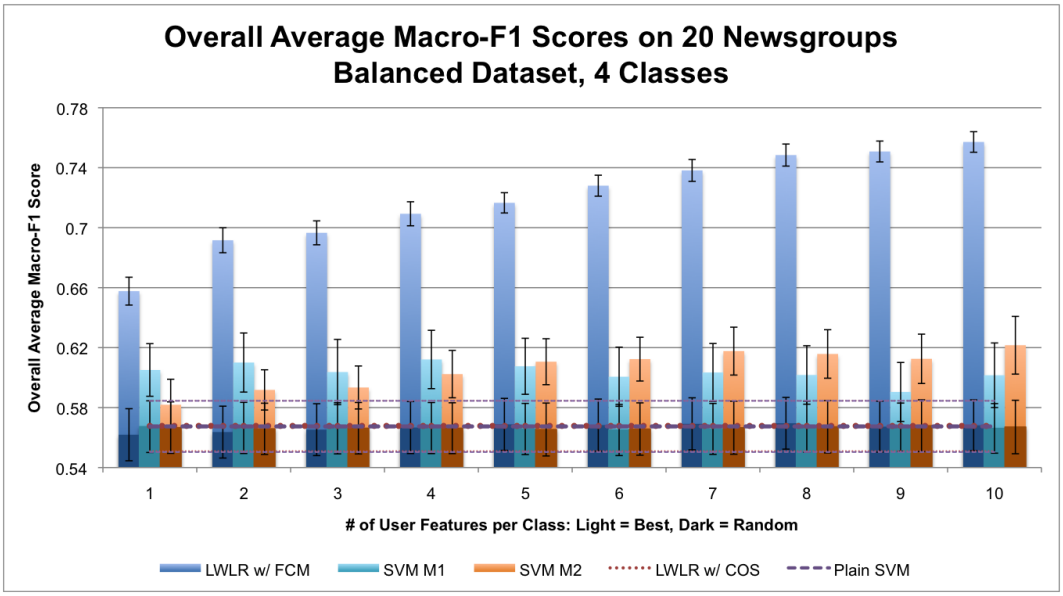

Figure 6.4: The overall average macro-F1 scores on the 20 Newsgroups Balanced/Random dataset, 4 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

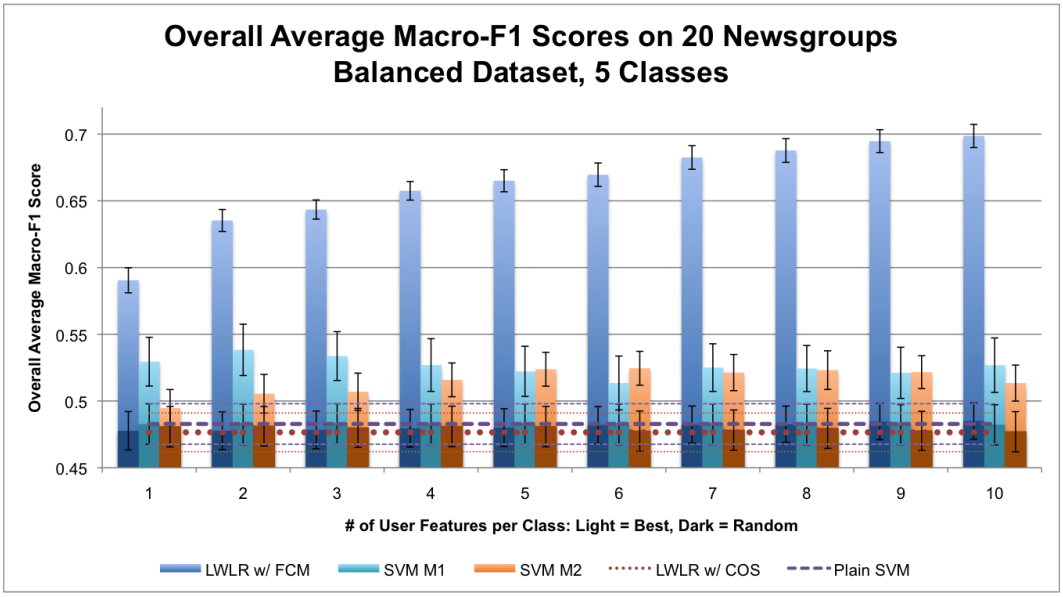

Figure 6.5: The overall average macro-F1 scores on the 20 Newsgroups Balanced/Random dataset, 5 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

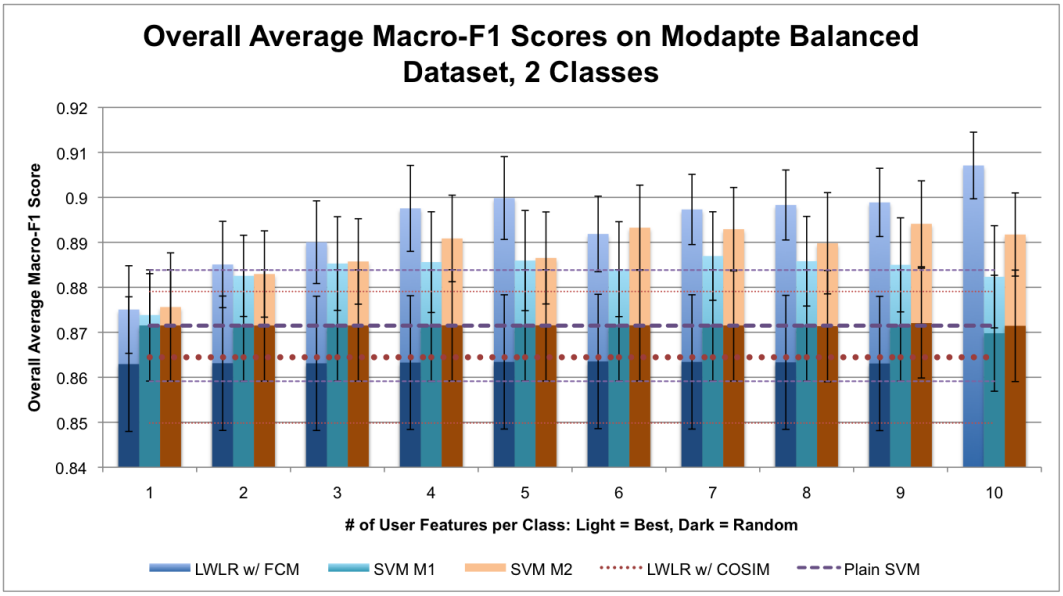

Figure 6.6: The overall average macro-F1 scores on the Modapte Balanced/Random dataset, 2 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

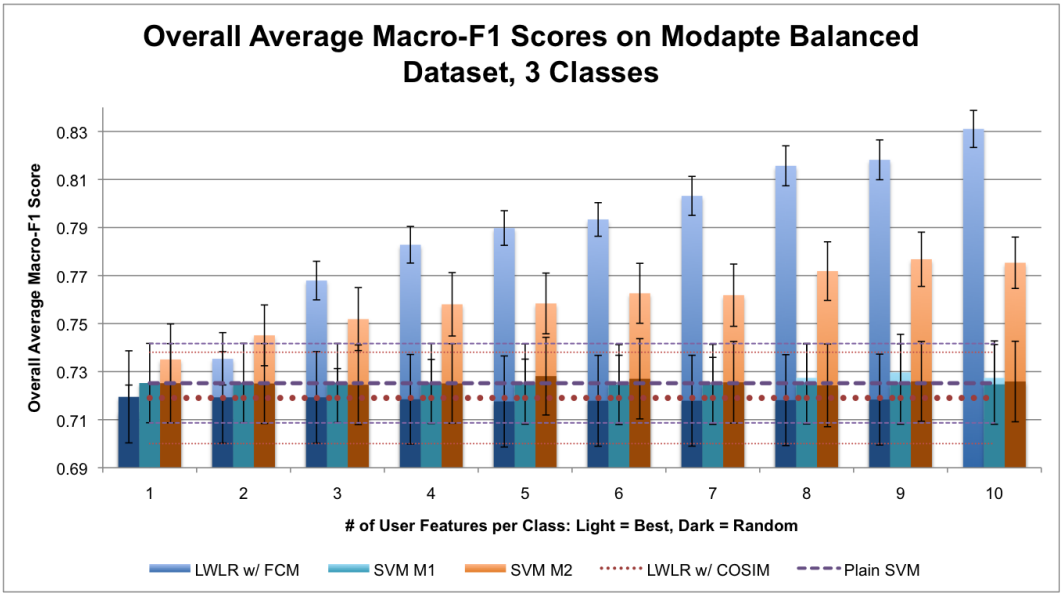

Figure 6.7: The overall average macro-F1 scores on the Modapte Balanced/Random dataset, 3 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

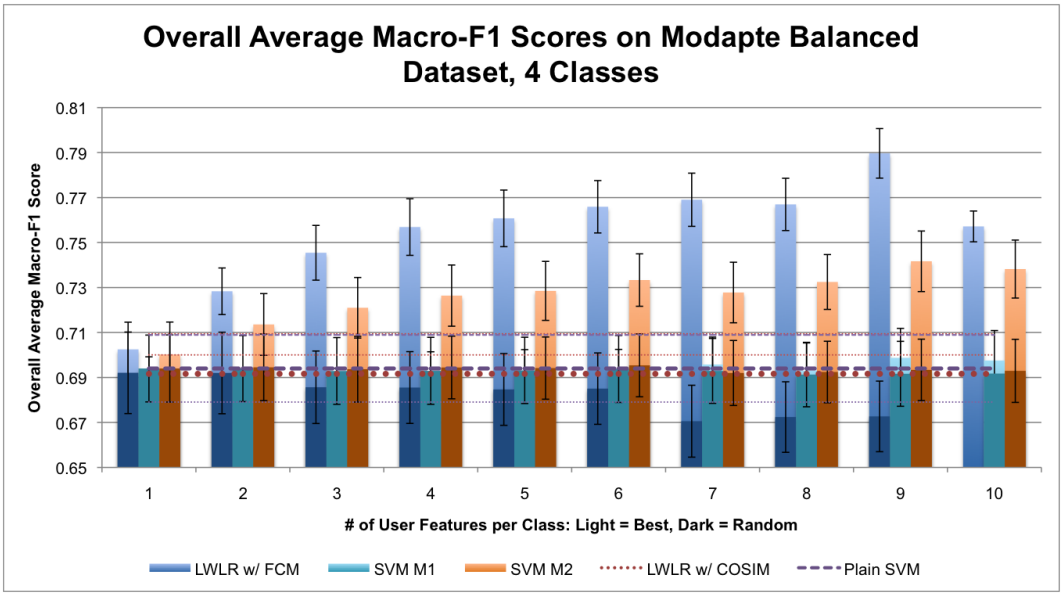

Figure 6.8: The overall average macro-F1 scores on the Modapte Balanced/Random dataset, 4 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

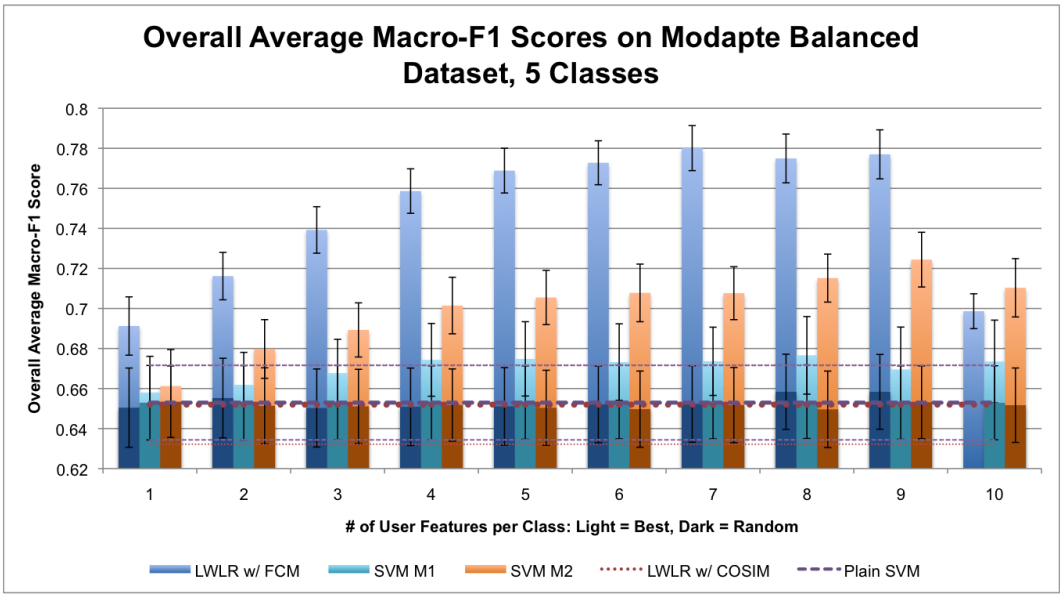

Figure 6.9: The overall average macro-F1 scores on the Modapte Balanced/Random dataset, 5 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

6.3.2 Uncertainty Datasets

We now look at look at the performance of the classifiers on the 20 Newsgroups Uncertainty dataset, shown in Figures 6.10 - 6.13. LWLR with FCM shows no significant improvement over method 2 in almost all cases, though it is significantly better than method 1 for all tests. Both FCM and method 2 show some response to the increasing number of user features, but it is much less dramatic than when using the Balanced/Random dataset. We also note that, in the 2 class test, both FCM and method 2 reach the highest average macro-F1 score out of all tests we performed, with method 2 achieving the highest score. As before, the use of random user features causes the classifiers to operate with almost no significant difference from their respective baselines.

Looking at the performance on the Modapte Uncertainty dataset, shown in Figures 6.14 - 6.17, we note that FCM is significantly worse than method 2 in the 2 - class tests, but shows no significant difference from method 2 in almost all the other test cases. Both methods show responsiveness to the increase in the number of best user features. Method 1 using the best user features fails to perform significantly better than both the baseline and method 1 using random user features in most cases except for the 5 class tests. As before, the use of random features causes the classifiers' performance to have no significant difference from their respective baselines in most cases.

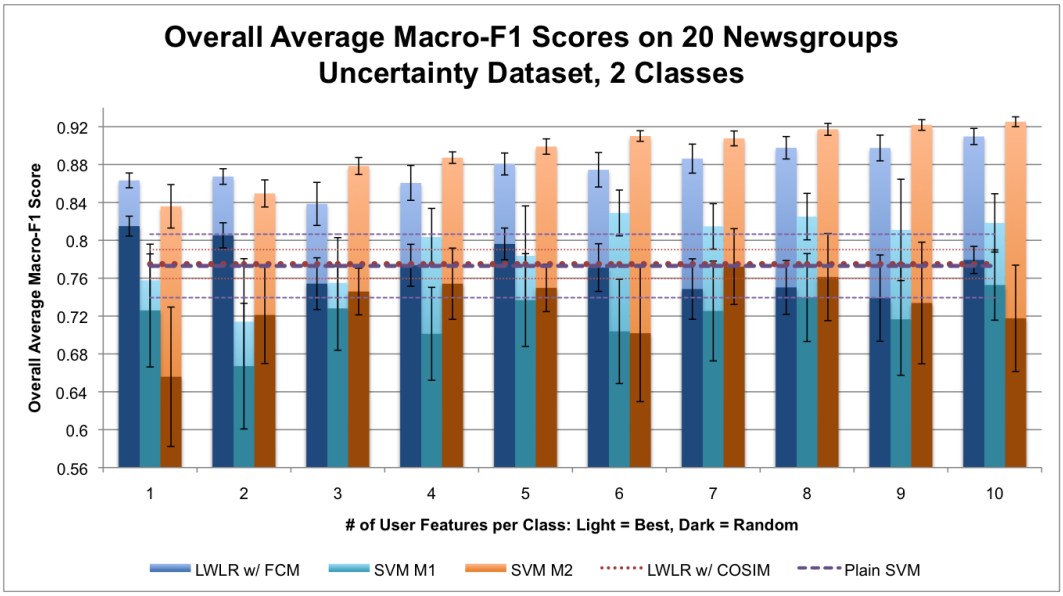

Figure 6.10: The overall average macro-F1 scores on the 20 Newsgroups Uncertainty dataset, 2 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

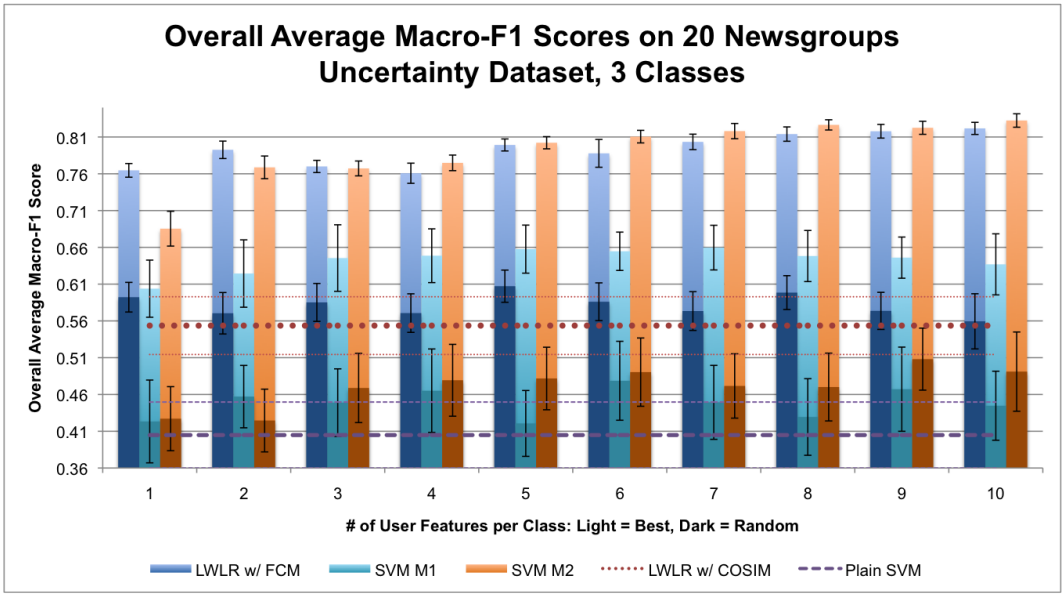

Figure 6.11: The overall average macro-F1 scores on the 20 Newsgroups Uncertainty dataset, 3 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

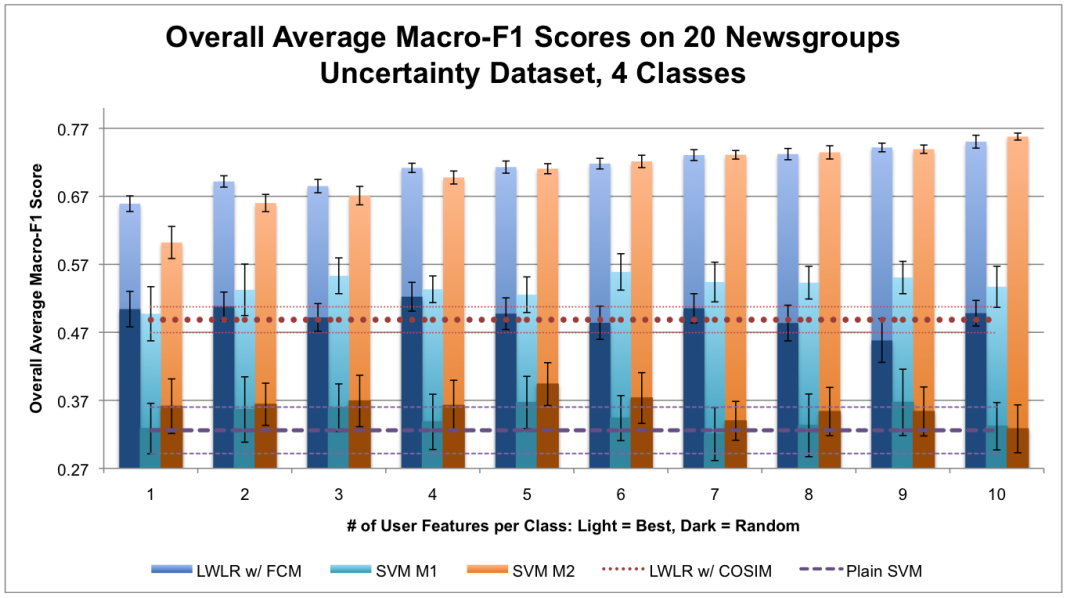

Figure 6.12: The overall average macro-F1 scores on the 20 Newsgroups Uncertainty dataset, 4 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

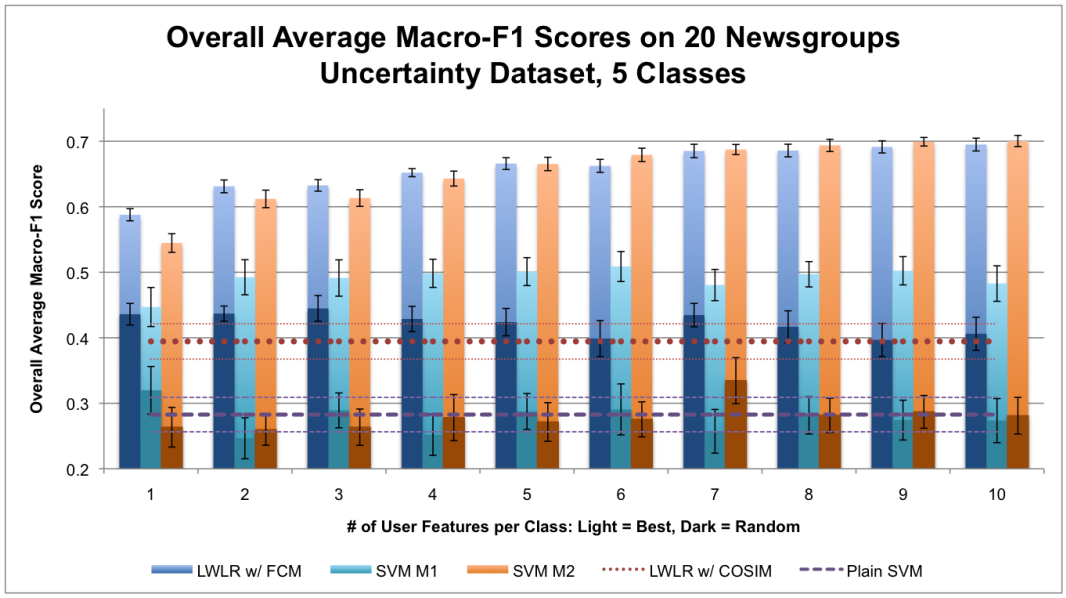

Figure 6.13: The overall average macro-F1 scores on the 20 Newsgroups Uncertainty dataset, 5 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

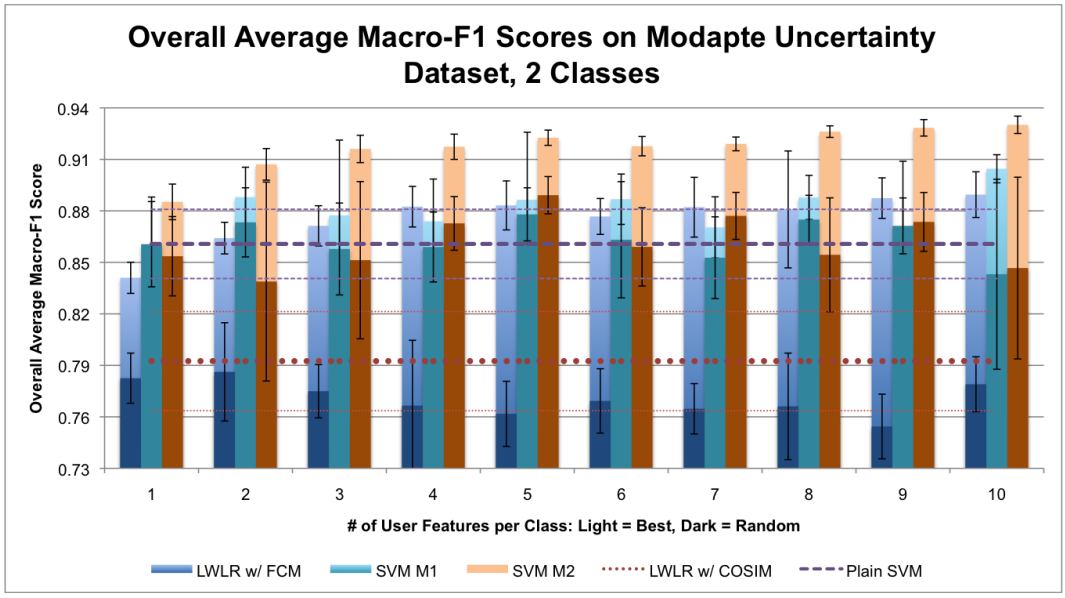

Figure 6.14: The overall average macro-F1 scores on the Modapte Uncertainty dataset, 2 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

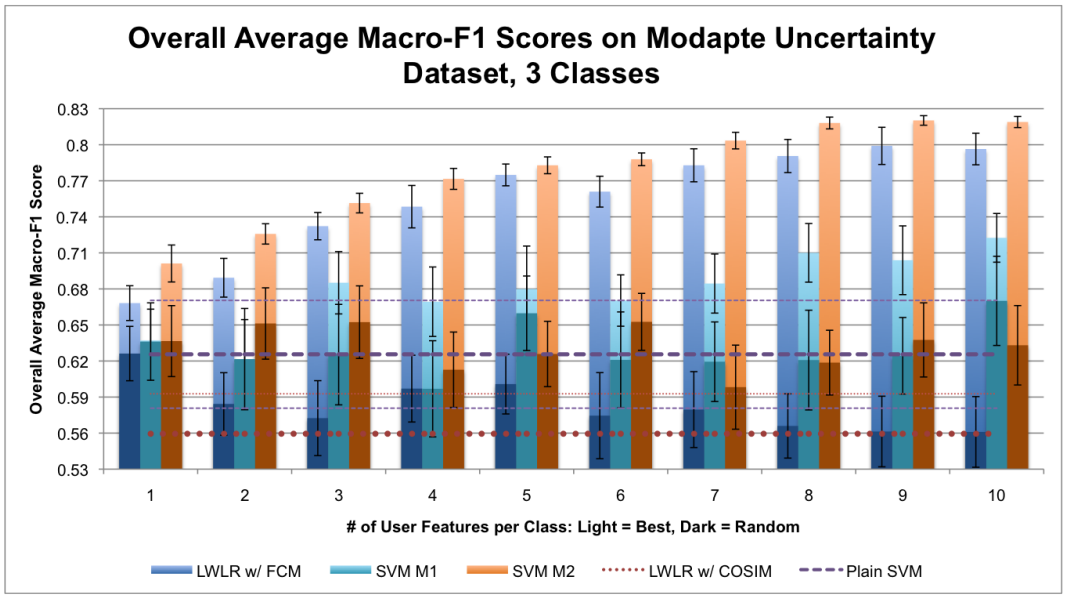

Figure 6.15: The overall average macro-F1 scores on the Modapte Uncertainty dataset, 3 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

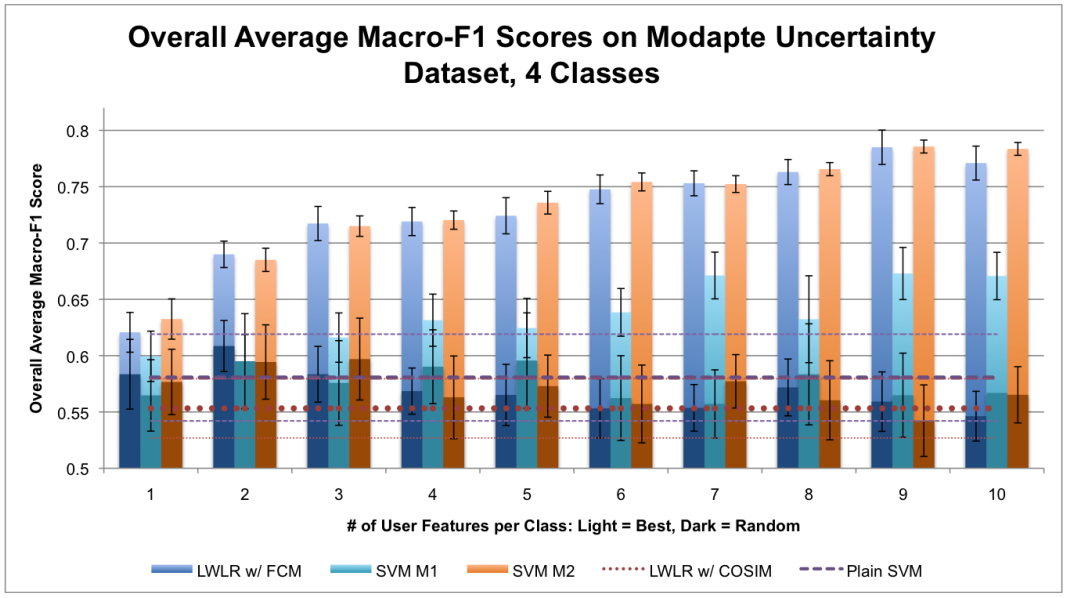

Figure 6.16: The overall average macro-F1 scores on the Modapte Uncertainty dataset, 4 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

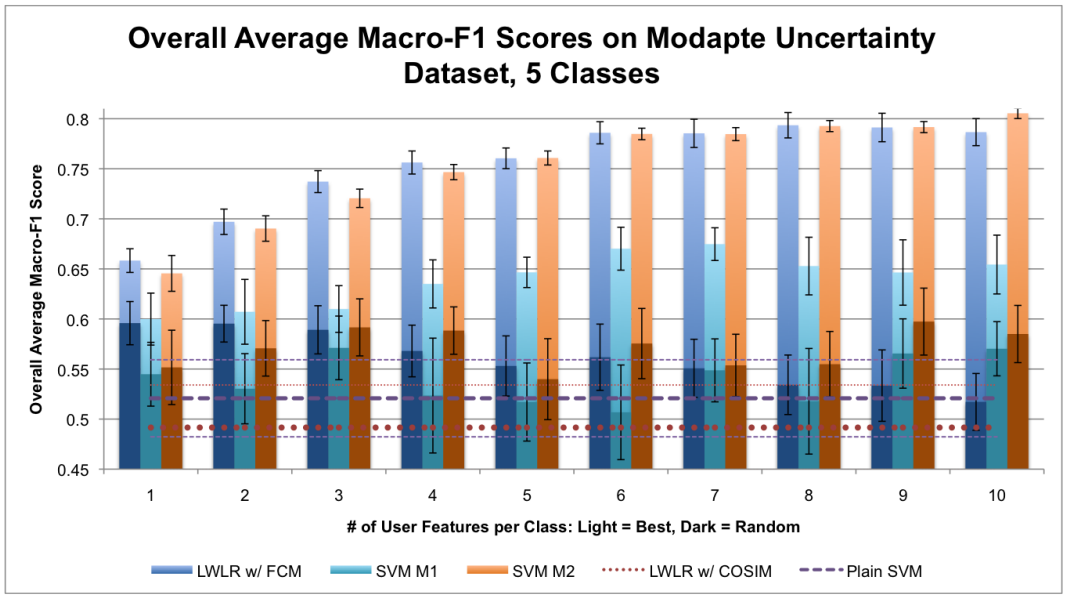

Figure 6.17: The overall average macro-F1 scores on the Modapte Uncertainty dataset, 5 classes and best/random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

6.3.3 Both Datasets

Looking over the results from all of the tests, we note that both baseline classifiers perform better on the Balanced/Random dataset than the Uncertainty dataset, and in almost all cases this difference is significant. We also note that method 2 appears to perform better when using uncertainty sampling. In contrast, method 1 appears to benefit from a lack of uncertainty sampling in all cases except for tests using only two classes.

Method 2 showed significant improvements when tested using uncertainty sampling, and the performance of LWLR with FCM remained extremely stable across both datasets when using the top user features. On the 20 Newsgroups datasets, FCM showed no significant difference in performance between the Balanced/Random tests and their corresponding Uncertainty tests in all cases except for one. On the Modapte datasets, it showed no significant difference in half of the tests. On average, the difference of the average macro-F1 score between the Modapte Balanced/Random and the Modapte Uncertainty tests using LWLR with FCM was less than .017.

6.3.4 Sensitivity to the Initial Training Data

The change in the training data had a significant effect on every feature relevance method. Out of all the methods, our FCM method appears to be the most robust to the initial training data, having no significant difference in performance and variance between the two different sampling methods in most cases. In contrast,

all other methods showed significant differences in performance between the two sampling methods, and this was particularly true for method two, whose change in performance when switching from using the Balanced/Random dataset to the Uncertainty dataset was very large.

6.3.4.1 Effects of Uncertainty Sampling

A quick look at the results of the Balanced/Random tests versus the Uncertainty tests shows a huge difference in the performance of the methods we tested. As we noted earlier, the performance of the baseline SVM was almost always significantly better when run on the Balanced/Random datasets as opposed to the Uncertainty datasets.

Raghavan & Allan use an uncertainty sampling method very similar to that used in previous studies [13, 1, 11]. Tong & Koller used uncertainty sampling in a 1-vs-All setup on both the Reuters 21578 and 20 Newsgroups corpora. They found the performance of uncertainty sampling can be variable and performed quite poorly on occasion [13]. Despite its poor performance, it was still able to perform reasonably well with fewer training instances than SVMs trained by random sampling, even balanced random sampling. Tong & Koller noted that uncertainty sampling tended to produce training sets where roughly half of the data points were for the positive class [13]. In contrast, random sampling on corpora with a large number of classes, such as 20 Newsgroups or Reuters 21578, in a 1-vs-All setting is unlikely to produce a similar number of positive training

examples without sampling a much larger number of instances. This is clearly seen in Tong & Koller's results, where their Random-Balanced sampling produced significantly better performance than random sampling.

Previous work has shown that uncertainty sampling usually performs significantly better in binary classification problems with fewer training instances than random sampling [13, 1, 11]. For both of our datasets, the baseline SVMs trained using uncertainty sampling performed significantly worse than SVMs trained using Balanced/Random sampling when the number of classes was greater than two. Despite the performance of the baselines being lower, uncertainty sampling did result in method 2 frequently producing the absolute highest average macro-F1 score in most tests on both datasets. In contrast, it typically degraded the performance of method 1. This suggests that uncertainty sampling may not be beneficial to classifiers in a multi-class setting, but can favorably affect certain types of feature relevance feedback methods.

A brief analysis of the average feature values for the best user features per dataset shows relatively little difference between the Balanced/Random data and the Uncertainty data. For the 20 Newsgroups data, the Balanced/Random training data typically contained feature values that were slightly higher on average than those in the Uncertainty training data. Using the Modpate data, both the Balanced/Random and Uncertainty training data typically had nearly the same average feature values for the best user features.

Looking at the average sum of the information gain for the user features in each instance in the datasets reveals that the Uncertainty dataset typically con-

tained instances that had a higher sum of information gain than compared to the Balanced/Random dataset. Often the average sum of the information in the Uncertainty dataset was two or more times greater than the Balanced/Random dataset. This suggests that uncertainty sampling tends to select instances that emphasize features that are more useful in discriminating between classes.

6.3.5 Sensitivity to the Quality of User Features

In the multi-class experiment, we see that the quality of features had a significant effect on the performance of the feature relevance methods. The use of top user features produced an increase in performance, and adding more top features continued to increase performance. In contrast, random features produced lower, but relatively stable, performance across all classifiers, regardless of the number of features included per class.

6.4 Conclusion

Our multi-class experiment extends Raghavan & Allan's original work and eliminates many of the issues that were present in the 1-vs-All experiment setup, which allowed us to more accurately evaluate not only the performance of the feature relevance methods we test, but also answer questions regarding the effect of the initial training data and feature quality. We demonstrated that different sampling methods can significantly affect the performance of feature relevance methods. In

addition, we found that feature quality has a large effect on the performance of feature relevance methods. Finally, we demonstrated that FCM often outperforms Raghavan & Allan's methods while being more robust to the initial training data.

Chapter 7 – Feature Relevance on User Study Data: Autocoder Experiment

7.1 Introduction

In Section 6.3.5 we noted that the feature relevance methods appeared to be sensitive to the quality of the user features. Using an oracle allowed us to select only high-quality features, which tested the performance of the feature relevance methods in an ideal setting. In order to evaluate the performance of the classifiers on more realistic data, we constructed an experiment that utilized user feedback from our earlier Autocoder experiment [8].

The Autocoder experiment presented users with the task of coding transcripts, a familiar exercise in the fields of psychology, social science, and HCI. With an application to aid them, users were given two transcripts to code, which consisted of verbal statements from participants in the experiment broken into segments. They were allowed to add features, remove features, and assign labels to segments. While performing this task, the application provided feedback and suggested labels to the user based on the feedback they had given so far. Users were given the ability to perform feature engineering, meaning that they could introduce new features that consisted of n-grams or complex regular expressions. These new features were incorporated into the feature relevance framework by adding them to the datasets

as features and indicating that they were relevant to a particular class.

The Autocoder dataset is particularly challenging as a machine learning problem. No gold standard for the data exists, which means that performance is evaluated based on what the users decided was the correct code for each segment. If users were inconsistent, that meant that the distribution of the data was not consistent. Additionally, coding is very time intensive, meaning that large training sets are not available. Since the Autocoder experiment was based around verbal transcripts, the data is also sequential, and the users were allowed to engineer features that spanned adjacent segments of the transcript. Finally, users were also able to introduce class imbalance through their labeling of segments in the transcripts.

7.2 Experiment

We constructed two sets of training and testing data. The first consisted of the autocoder data without any sequential information. In the original study, users were given a segment, but had access to the previous segment for creating features that spanned multiple segments. Using a recurrent sliding window, the classifier was given access to each instance and its predecessor. We create a non-sequential version of the data in order to perform validation to tune our parameters. We call this first dataset the Non-Sequential Autocoder (NSAC) set. The second dataset used a recurrent sliding window to re-create the same data that was given to the classifier in the study. We call this data set the Sequential Autocoder (SAC) set.

Each dataset consisted of two transcripts per user. The first transcript was used as the training set, and its associated features the user features. The second transcript was used as the test set. Using the NSAC set, we created thirty random splits of half of the training data to use as validation sets. Using these validation sets, we tuned our classifiers' parameters. These parameters, shown in Table 7.1, were used for tests on both the NSAC and SAC datasets, as it was not possible to sample good quality validation sets from the sequential data.

Table 7.1: Parameters obtained from the SAC validation sets.

	Parameters
COSIM	$k^2 = .001$
FCM	$k^2 = .001$
Method 1	$a = 15, b = 1$
Method 2	$r = 1$

Using the training and testing sets, we computed the overall macro-F1 and micro-F1 scores for each classifier, including a logistic regression classifier. Our analysis focuses on the macro-F1 scores, but the micro-F1 scores are available in Appendix A. All classifiers except the baseline classifiers, LWLR with COSIM and the plain SVM, and logistic regression, were given the user features.

7.3 Results

Figure 7.1 shows the results of tests using the NSAC dataset. There is no significant difference in performance between the LWLR methods, which both perform

significantly better than all SVM methods, as well as logistic regression. Using the sequential data, shown in Figure 7.2, causes a decrease in the performance of method 1, but all other classifiers remain relatively unchanged.

LWLR with COSIM performs slightly better than LWLR with FCM on both datasets. In the Autocoder study, we noted that users sometimes added non-predictive features that could reduce the performance of classifiers. Looking at the individual results for the LWLR classifiers, we noted that they performed identically except in a handful of cases where LWLR with FCM performed either slightly better or slightly worse than LWLR with COSIM. It is possible that non-predictive features added by certain users caused a decrease in the performance of LWLR with FCM. This is consistent with our results from Section 6.3, where the use of random feature sets caused LWLR with FCM to perform about the same as LWLR with COSIM, though slightly lower.

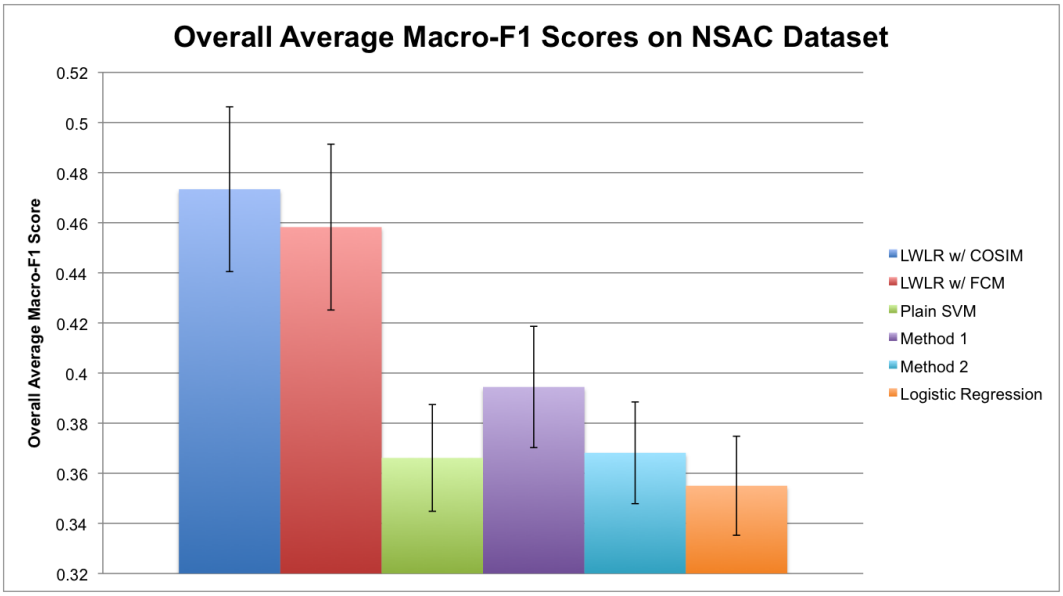

Figure 7.1: The overall average macro-F1 scores on the NSAC Autocoder dataset. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

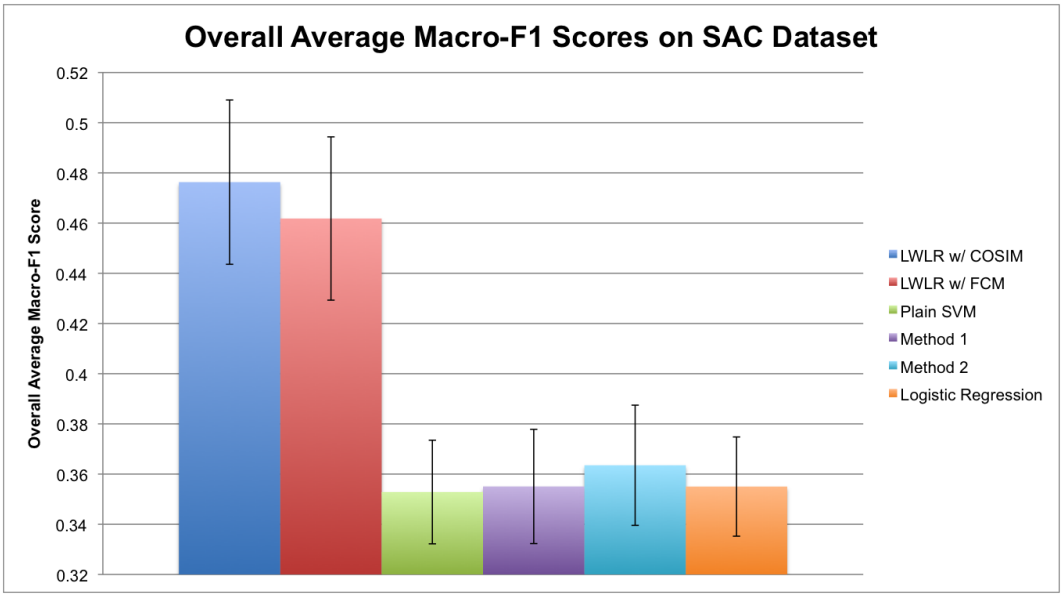

Figure 7.2: The overall average macro-F1 scores on the SAC Autocoder dataset. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

Chapter 8 – Conclusion

8.1 Introduction

At the beginning of this paper we asked three questions with respect to feature relevance feedback that we considered to be unanswered by Raghavan & Allan's previous experiment. We introduced a new experiment in a multi-class setting to help us answer these questions. In the following sections we summarize our findings and discuss future work.

8.2 Answering our Research Questions

The first question we asked was "**Does feature relevance work well outside of binary classification and, if so, how well does it scale with the number of classes?**" Using our new method, the *Feature Contrast Method*, we demonstrated that feature relevance does work well in a multi-class setting. Additionally, our method appeared to scale well as the number of classes was increased when compared to Raghavan & Allan's methods.

The second question was "**How does the quality of the initial labeled training points affect performance?**" We found that the initial training data can have a large effect on the performance of feature relevance methods. Some methods varied dramatically, while others, specifically FCM, maintained very sta-

ble performance.

The final question we asked was "**Does the quality of features given as feedback affect the performance of feature relevance?**" Our experiments demonstrated that feature quality can have a dramatic effect on the performance of feature relevance methods. The use of poor quality feedback significantly reduced the performance of the tested feature relevance methods in all cases.

8.3 The Feature Contrast Method

Our method, the Feature Contrast Method, demonstrated consistently high performance and responsiveness to feedback across different sampling methods. However, it suffers from several drawbacks. Training sets with few user features, many low quality user features, or severe class imbalance can reduce its performance. It is also sensitive to the kernel value chosen. We chose the kernel value using a validation set, which may not be available in a more realistic setting.

8.4 Future Work

Showing that more realistic problems can be addressed by feature relevance methods motivates further exploration of how these methods interact with users when addressing real-world problems. In particular, our findings with respect to the importance feature quality pave the way for future work addressing incorporating real user-feedback and methods for aiding users in providing quality features.

Active learning addresses this by selecting features for users to label, but this method may not be suitable in all cases. Alternative methods that are able to identify and filter potentially poor feedback could aid users in providing consistently high-quality feedback.

In addition to feature quality, our work has shown that incorporating feature relevance feedback in a supervised setting can improve the performance of classifiers significantly. This can be extended to semi-supervised methods where feature relevance could be used to contribute to performance through the use of unlabeled data. Previous work has often done this through the use of feature feedback to soft-label unlabeled data to be used as additional training data. Further work in this area could result in novel methods that use feature relevance more effectively than for only soft-labeling.

Bibliography

[1] Colin Campbell, Nello Cristianini, and Alex J. Smola. Query learning with large margin classifiers. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 111–118, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[2] Kan Deng. *Omega: on-line memory-based general purpose system classifier*. PhD thesis, Pittsburgh, PA, USA, 1999. Chair-Moore, Andrew William.

[3] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602, New York, NY, USA, 2008. ACM.

[4] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 320–327, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[5] Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. Text classification by labeling words. In *AAAI'04: Proceedings of the 19th national conference on Artifical intelligence*, pages 425–430. AAAI Press / The MIT Press, 2004.

[6] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989.

[7] Andrew McCallum, Gideon Mann, and Gregory Duck. Generalized expectation criteria. Amherst Technical Report 2007-60, University of Massachusetts, 2007.

[8] Ian Oberst, Travis Moore, Weng-Keen Wong, Todd Kulesza, Simone Stumpf, Yann Riche, and Margaret Burnett. End-user feature engineering in the presence of class imbalance. Technical report, Oregon State University, http://hdl.handle.net/1957/13225, November 2009.

[9] Hema Raghavan and James Allan. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 79–86, New York, NY, USA, 2007. ACM.

[10] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

[11] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 839–846, San Francisco, CA, USA, 2000. Morgan Kaufmann.

[12] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *Int. J. Hum.-Comput. Stud.*, 67(8):639–662, 2009.

[13] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.

[14] Xiaoyun Wu and Rohini Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–333, New York, NY, USA, 2004. ACM.

APPENDICES

Appendix A – Micro-F1 Results

A.1 Multi-Class Tests

Table A.1: Overall average micro-F1 scores on the 20 Newsgroups dataset for the baseline classifiers. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

Baseline Classifiers, Overall Average Micro-F1 Scores on the 20 Newsgroup Dataset

# of Classes → *Classifier ↓*	2	3
COSIM	.7942 +/- .0189	.6789 +/- .0127
SVM	.7888 +/- .0222	.6689 +/- .0188
	4	5
COSIM	.5786 +/- .0155	.4917 +/- 0.0135
SVM	.5706 +/- .017	.4857 +/- .0141

Table A.2: Overall average micro-F1 scores on the Modapte dataset for the baseline classifiers. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

Baseline Classifiers, Overall Average Micro-F1 Scores on the Modapte Dataset

# of Classes → *Classifier ↓*	2	3
COSIM	.8535 +/- .0203	.7261 +/- .0172
SVM	.8725 +/- .0145	.7246 +/- .0165
	4	5
COSIM	.6986 +/- .0152	.6571 +/- .018
SVM	.6909 +/- .0153	.6546 +/- .0167

Table A.3: Overall average micro-F1 scores on the 20 Newsgroups BR dataset for LWLR using FCM, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on 20 Newsgroups BR, Using Top Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.842 +/- 0.0123	0.8583 +/- 0.0104	0.8602 +/- 0.0088	0.8693 +/- 0.0084
FCM/3 Classes	0.7621 +/- 0.0141	0.7955 +/- 0.0116	0.7814 +/- 0.0089	0.7943 +/- 0.0086
FCM/4 Classes	0.6616 +/- 0.0091	0.6953 +/- 0.0078	0.7008 +/- 0.0075	0.7104 +/- 0.0078
FCM/5 Classes	0.5978 +/- 0.0082	0.6379 +/- 0.0074	0.6463 +/- 0.0068	0.6589 +/- 0.0069
	5	6	7	8
FCM/2 Classes	0.8777 +/- 0.0085	0.8821 +/- 0.0086	0.8891 +/- 0.0078	0.8958 +/- 0.0086
FCM/3 Classes	0.8093 +/- 0.0089	0.8124 +/- 0.0089	0.8221 +/- 0.0086	0.8202 +/- 0.008
FCM/4 Classes	0.7204 +/- 0.0064	0.731 +/- 0.0071	0.7404 +/- 0.0073	0.7506 +/- 0.0074
FCM/5 Classes	0.6676 +/- 0.0082	0.6716 +/- 0.0087	0.6844 +/- 0.0088	0.6895 +/- 0.009
	9	10		
FCM/2 Classes	0.90204 +/- 0.0079	0.9038 +/- 0.0079		
FCM/3 Classes	0.8251 +/- 0.0079	0.8325 +/- 0.0077		
FCM/4 Classes	0.7527 +/- 0.0071	0.7589 +/- 0.0072		
FCM/5 Classes	0.6963 +/- 0.0086	0.6999 +/- 0.0087		

Table A.4: Overall average micro-F1 scores on the 20 Newsgroups dataset for SVM Method 1 & 2, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on 20 Newsgroups BR, Using Top Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.8017 +/- 0.0207	0.8071 +/- 0.0219	0.8231 +/- 0.0164	0.8216 +/- 0.0168
SVM Method 1/3 Classes	0.7119 +/- 0.0168	0.7159 +/- 0.0165	0.7113 +/- 0.017	0.7189 +/- 0.0163
SVM Method 1/4 Classes	0.6082 +/- 0.0179	0.6135 +/- 0.0191	0.6062 +/- 0.0215	0.6158 +/- 0.019++
SVM Method 1/5 Classes	0.5323 +/- 0.0173	0.542 +/- 0.0181	0.536 +/- 0.01702	0.5302 +/- 0.0183
SVM Method 2/2 Classes	0.7946 +/- 0.0227	0.7971 +/- 0.0235	0.7985 +/- 0.0237	0.7993 +/- 0.0237
SVM Method 2/3 Classes	0.6802 +/- 0.0174	0.6853 +/- 0.0171	0.6933 +/- 0.016	0.6913 +/- 0.0178
SVM Method 2/4 Classes	0.5849 +/- 0.0168	0.59502 +/- 0.0125	0.5972 +/- 0.0135	0.6056 +/- 0.0144
SVM Method 2/5 Classes	0.4976 +/- 0.0128	0.5081 +/- 0.0131	0.5102 +/- 0.0129	0.5201 +/- 0.0114
	5	6	7	8
SVM Method 1/2 Classes	0.8304 +/- 0.0166	0.8325 +/- 0.0168	0.8299 +/- 0.0203	0.82903 +/- 0.0161
SVM Method 1/3 Classes	0.7143 +/- 0.0178	0.7105 +/- 0.0192	0.7165 +/- 0.0156	0.7118 +/- 0.0175
SVM Method 1/4 Classes	0.6128 +/- 0.0186	0.6063 +/- 0.0197	0.6094 +/- 0.0198	0.6076 +/- 0.0188
SVM Method 1/5 Classes	0.5285 +/- 0.0177	0.5195 +/- 0.0196	0.5299 +/- 0.0167	0.5308 +/- 0.0165
SVM Method 2/2 Classes	0.7947 +/- 0.0246	0.79304 +/- 0.0264	0.7909 +/- 0.0255	0.7998 +/- 0.0254
SVM Method 2/3 Classes	0.6942 +/- 0.0186	0.6925 +/- 0.01804	0.6949 +/- 0.0183	0.69504 +/- 0.0169
SVM Method 2/4 Classes	0.614 +/- 0.014	0.6158 +/- 0.0136	0.6225 +/- 0.014	0.6206 +/- 0.0145
SVM Method 2/5 Classes	0.5287 +/- 0.0114	0.5305 +/- 0.0117	0.528 +/- 0.0134	0.5298 +/- 0.0108
	9	10		
SVM Method 1/2 Classes	0.8279 +/- 0.0172	0.8268 +/- 0.0195		
SVM Method 1/3 Classes	0.7138 +/- 0.0194	0.7168 +/- 0.0175		
SVM Method 1/4 Classes	0.5954 +/- 0.0197	0.6068 +/- 0.0212		
SVM Method 1/5 Classes	0.5268 +/- 0.0181	0.5312 +/- 0.0197		
SVM Method 2/2 Classes	0.7926 +/- 0.0252	0.7934 +/- 0.0249		
SVM Method 2/3 Classes	0.6936 +/- 0.0178	0.69102 +/- 0.0181		
SVM Method 2/4 Classes	0.619 +/- 0.0142	0.6276 +/- 0.0166		
SVM Method 2/5 Classes	0.5293 +/- 0.0125	0.5239 +/- 0.0142		

Table A.5: Overall average micro-F1 scores on the 20 Newsgroups BR dataset for LWLR using FCM, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on 20 Newsgroups BR, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.8022 +/- 0.0169	0.8032 +/- 0.0169	0.8025 +/- 0.0168	0.8035 +/- 0.0166
FCM/3 Classes	0.6729 +/- 0.0159	0.6732 +/- 0.0159	0.6735 +/- 0.0158	0.6739 +/- 0.0154
FCM/4 Classes	0.5696 +/- 0.0164	0.5712 +/- 0.0164	0.5727 +/- 0.0163	0.5741 +/- 0.0163
FCM/5 Classes	0.4882 +/- 0.0123	0.4886 +/- 0.0121	0.4892 +/- 0.012	0.4902 +/- 0.01204
	5	6	7	8
FCM/2 Classes	0.8041 +/- 0.0165	0.8042 +/- 0.0165	0.8042 +/- 0.0165	0.8043 +/- 0.0165
FCM/3 Classes	0.6717 +/- 0.0151	0.6726 +/- 0.0149	0.674 +/- 0.0148	0.6742 +/- 0.0148
FCM/4 Classes	0.5766 +/- 0.0161	0.57601 +/- 0.0162	0.5768 +/- 0.0162	0.5772 +/- 0.01604
FCM/5 Classes	0.4905 +/- 0.0121	0.4922 +/- 0.0119	0.4926 +/- 0.0118	0.4921 +/- 0.0116
	9	10		
FCM/2 Classes	0.8046 +/- 0.0165	0.8048 +/- 0.0165		
FCM/3 Classes	0.6745 +/- 0.0148	0.6746 +/- 0.0147		
FCM/4 Classes	0.5746 +/- 0.0158	0.5755 +/- 0.0158		
FCM/5 Classes	0.4938 +/- 0.0116	0.4943 +/- 0.0116		

Table A.6: Overall average micro-F1 scores on the 20 Newsgroups BR dataset for SVM Method 1 & 2, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on 20 Newsgroups BR, Using Random Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.7888 +/- 0.0222	0.7887 +/- 0.0222	0.7884 +/- 0.0222	0.7882 +/- 0.0222
SVM Method 1/3 Classes	0.6688 +/- 0.0188	0.6689 +/- 0.0188	0.6689 +/- 0.0189	0.669 +/- 0.0188
SVM Method 1/4 Classes	0.5704 +/- 0.0171	0.5695 +/- 0.0169	0.5693 +/- 0.0169	0.5695 +/- 0.017
SVM Method 1/5 Classes	0.4856 +/- 0.0141	0.4854 +/- 0.0141	0.4857 +/- 0.0141	0.4856 +/- 0.0142
SVM Method 2/2 Classes	0.7888 +/- 0.0223	0.78901 +/- 0.0223	0.7871 +/- 0.0223	0.7887 +/- 0.0222
SVM Method 2/3 Classes	0.6687 +/- 0.0188	0.6688 +/- 0.0188	0.6687 +/- 0.0188	0.6693 +/- 0.0186
SVM Method 2/4 Classes	0.5703 +/- 0.01703	0.5696 +/- 0.017	0.5699 +/- 0.0171	0.57002 +/- 0.0169
SVM Method 2/5 Classes	0.4843 +/- 0.0141	0.4848 +/- 0.0138	0.4843 +/- 0.0136	0.4845 +/- 0.01404
	5	6	7	8
SVM Method 1/2 Classes	0.7883 +/- 0.0221	0.7883 +/- 0.0221	0.7886 +/- 0.0221	0.7882 +/- 0.0222
SVM Method 1/3 Classes	0.6693 +/- 0.0186	0.6685 +/- 0.01903	0.6684 +/- 0.01904	0.6682 +/- 0.0191
SVM Method 1/4 Classes	0.5688 +/- 0.0169	0.5682 +/- 0.0169	0.5691 +/- 0.0168	0.5705 +/- 0.0169
SVM Method 1/5 Classes	0.4856 +/- 0.0141	0.4854 +/- 0.0141	0.4855 +/- 0.0141	0.4852 +/- 0.0141
SVM Method 2/2 Classes	0.7876 +/- 0.0224	0.7877 +/- 0.0221	0.787 +/- 0.0221	0.788 +/- 0.0223
SVM Method 2/3 Classes	0.666 +/- 0.0187	0.66302 +/- 0.0206	0.6627 +/- 0.0204	0.66403 +/- 0.0204
SVM Method 2/4 Classes	0.5699 +/- 0.0173	0.57002 +/- 0.0172	0.5713 +/- 0.0172	0.5716 +/- 0.0172
SVM Method 2/5 Classes	0.4844 +/- 0.0139	0.4827 +/- 0.0138	0.483 +/- 0.0138	0.4833 +/- 0.0138
	9	10		
SVM Method 1/2 Classes	0.7884 +/- 0.0221	0.7882 +/- 0.0222		
SVM Method 1/3 Classes	0.6683 +/- 0.0191	0.6684 +/- 0.0191		
SVM Method 1/4 Classes	0.5701 +/- 0.016	0.5695 +/- 0.0164		
SVM Method 1/5 Classes	0.4853 +/- 0.0139	0.4851 +/- 0.014		
SVM Method 2/2 Classes	0.7881 +/- 0.0222	0.7876 +/- 0.0223		
SVM Method 2/3 Classes	0.66302 +/- 0.0202	0.6643 +/- 0.0198		
SVM Method 2/4 Classes	0.5727 +/- 0.0168	0.572 +/- 0.0176		
SVM Method 2/5 Classes	0.4828 +/- 0.0135	0.4819 +/- 0.0137		

Table A.7: Overall average micro-F1 scores on the Modapte BR dataset for LWLR using FCM, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on Modapte BR, Using Top Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.8723 +/- 0.0132	0.8846 +/- 0.0126	0.8915 +/- 0.0117	0.8979 +/- 0.0121
FCM/3 Classes	0.7152 +/- 0.0099	0.7371 +/- 0.0096	0.768 +/- 0.0068	0.7831 +/- 0.0065
FCM/4 Classes	0.7043 +/- 0.0122	0.7266 +/- 0.0098	0.74603 +/- 0.0112	0.7577 +/- 0.0123
FCM/5 Classes	0.6928 +/- 0.0137	0.7145 +/- 0.0115	0.7387 +/- 0.0115	0.7591 +/- 0.0112
	5	6	7	8
FCM/2 Classes	0.9006 +/- 0.0116	0.8946 +/- 0.0104	0.8996 +/- 0.0094	0.9008 +/- 0.0093
FCM/3 Classes	0.79001 +/- 0.0061	0.7936 +/- 0.006	0.8038 +/- 0.00701	0.8156 +/- 0.0076
FCM/4 Classes	0.7625 +/- 0.0122	0.7683 +/- 0.0112	0.7714 +/- 0.0113	0.7705 +/- 0.0107
FCM/5 Classes	0.7697 +/- 0.0112	0.7741 +/- 0.011	0.7802 +/- 0.0111	0.7752 +/- 0.0117
	9	10		
FCM/2 Classes	0.9016 +/- 0.00902	0.9038 +/- 0.0079		
FCM/3 Classes	0.8181 +/- 0.0076	0.8325 +/- 0.0077		
FCM/4 Classes	0.7914 +/- 0.0101	0.7589 +/- 0.0072		
FCM/5 Classes	0.7775 +/- 0.0117	0.6999 +/- 0.0087		

Table A.8: Overall average micro-F1 scores on the Modapte BR dataset for SVM Method 1 & 2, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on Modapte BR, Using Top Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.8779 +/- 0.0117	0.8855 +/- 0.0116	0.8884 +/- 0.0135	0.8876 +/- 0.0149
SVM Method 1/3 Classes	0.7029 +/- 0.0159	0.708 +/- 0.0154	0.7142 +/- 0.0156	0.7181 +/- 0.0152
SVM Method 1/4 Classes	0.6082 +/- 0.0179	0.6135 +/- 0.0191	0.6062 +/- 0.0215	0.6158 +/- 0.019++
SVM Method 1/5 Classes	0.6604 +/- 0.0163	0.6604 +/- 0.015	0.66602 +/- 0.0155	0.6731 +/- 0.0167
SVM Method 2/2 Classes	0.8784 +/- 0.01304	0.8846 +/- 0.0116	0.8878 +/- 0.0113	0.8935 +/- 0.0111
SVM Method 2/3 Classes	0.7341 +/- 0.0147	0.7433 +/- 0.0129	0.7501 +/- 0.0131	0.7561 +/- 0.0134
SVM Method 2/4 Classes	0.6962 +/- 0.0148	0.7098 +/- 0.0142	0.7178 +/- 0.014	0.7237 +/- 0.0141
SVM Method 2/5 Classes	0.6598 +/- 0.0174	0.6771 +/- 0.0141	0.6863 +/- 0.0135	0.6988 +/- 0.0138
	5	6	7	8
SVM Method 1/2 Classes	0.8881 +/- 0.0146	0.88602 +/- 0.0138	0.8895 +/- 0.0126	0.8879 +/- 0.0129
SVM Method 1/3 Classes	0.7196 +/- 0.0141	0.7207 +/- 0.0145	0.7196 +/- 0.015	0.7263 +/- 0.0145
SVM Method 1/4 Classes	0.6862 +/- 0.0123	0.6864 +/- 0.0117	0.6928 +/- 0.0122	0.6894 +/- 0.0127
SVM Method 1/5 Classes	0.6731 +/- 0.0176	0.6722 +/- 0.0176	0.6724 +/- 0.0158	0.6759 +/- 0.0179
SVM Method 2/2 Classes	0.8877 +/- 0.0127	0.8954 +/- 0.0114	0.8949 +/- 0.0107	0.8894 +/- 0.0142
SVM Method 2/3 Classes	0.757 +/- 0.0127	0.7613 +/- 0.0125	0.76103 +/- 0.0128	0.7703 +/- 0.0121
SVM Method 2/4 Classes	0.7259 +/- 0.0137	0.7316 +/- 0.0119	0.7257 +/- 0.0139	0.732 +/- 0.0124
SVM Method 2/5 Classes	0.7026 +/- 0.0133	0.7053 +/- 0.0141	0.7069 +/- 0.0128	0.7138 +/- 0.0115
	9	10		
SVM Method 1/2 Classes	0.8868 +/- 0.0138	0.8831 +/- 0.0152		
SVM Method 1/3 Classes	0.7278 +/- 0.0164	0.7256 +/- 0.0157		
SVM Method 1/4 Classes	0.6955 +/- 0.0135	0.6943 +/- 0.0132		
SVM Method 1/5 Classes	0.6694 +/- 0.02003	0.6731 +/- 0.0194		
SVM Method 2/2 Classes	0.8957 +/- 0.0105	0.8923 +/- 0.0108		
SVM Method 2/3 Classes	0.7756 +/- 0.0111	0.7745 +/- 0.0103		
SVM Method 2/4 Classes	0.7394 +/- 0.0137	0.7371 +/- 0.0129		
SVM Method 2/5 Classes	0.7224 +/- 0.0136	0.7095 +/- 0.014		

Table A.9: Overall average micro-F1 scores on the Modapte BR dataset for LWLR using FCM, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on Modapte BR, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.8508 +/- 0.0208	0.85104 +/- 0.0207	0.85104 +/- 0.0207	0.8513 +/- 0.0206
FCM/3 Classes	0.7268 +/- 0.0173	0.7265 +/- 0.0172	0.7265 +/- 0.0172	0.7256 +/- 0.0167
FCM/4 Classes	0.6995 +/- 0.0155	0.6995 +/- 0.0155	0.6922 +/- 0.0141	0.6923 +/- 0.0139
FCM/5 Classes	0.6557 +/- 0.018	0.6607 +/- 0.018	0.6548 +/- 0.0177	0.6553 +/- 0.0176
	5	6	7	8
FCM/2 Classes	0.8514 +/- 0.0207	0.8515 +/- 0.0207	0.8514 +/- 0.0207	0.8513 +/- 0.0206
FCM/3 Classes	0.7247 +/- 0.0169	0.725 +/- 0.0169	0.725 +/- 0.0169	0.7252 +/- 0.0169
FCM/4 Classes	0.6913 +/- 0.0139	0.6916 +/- 0.0138	0.6786 +/- 0.0139	0.6811 +/- 0.0136
FCM/5 Classes	0.6555 +/- 0.0176	0.6564 +/- 0.0175	0.6566 +/- 0.0174	0.6621 +/- 0.0173
	9	10		
FCM/2 Classes	0.8511 +/- 0.0207	0.8048 +/- 0.0165		
FCM/3 Classes	0.7254 +/- 0.0169	0.6746 +/- 0.0147		
FCM/4 Classes	0.6814 +/- 0.0136	0.5755 +/- 0.0158		
FCM/5 Classes	0.6621 +/- 0.0172	0.4943 +/- 0.0116		

Table A.10: Overall average micro-F1 scores on the Modapte BR dataset for SVM Method 1 & 2, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on Modapte BR, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.8726 +/- 0.0145	0.8725 +/- 0.0145	0.8726 +/- 0.0145	0.8725 +/- 0.0145
SVM Method 1/3 Classes	0.7247 +/- 0.0166	0.7247 +/- 0.0166	0.7247 +/- 0.0165	0.7245 +/- 0.0167
SVM Method 1/4 Classes	0.6909 +/- 0.0153	0.6909 +/- 0.0152	0.6899 +/- 0.015	0.6899 +/- 0.0152
SVM Method 1/5 Classes	0.6546 +/- 0.0167	0.6545 +/- 0.0167	0.6545 +/- 0.0165	0.6544 +/- 0.0165
SVM Method 2/2 Classes	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145
SVM Method 2/3 Classes	0.7246 +/- 0.0165	0.7245 +/- 0.0167	0.724 +/- 0.0166	0.7246 +/- 0.0166
SVM Method 2/4 Classes	0.6909 +/- 0.0153	0.6914 +/- 0.0152	0.6913 +/- 0.0149	0.692 +/- 0.0142
SVM Method 2/5 Classes	0.6548 +/- 0.0163	0.6534 +/- 0.0169	0.652 +/- 0.0166	0.6531 +/- 0.0164
	5	6	7	8
SVM Method 1/2 Classes	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145
SVM Method 1/3 Classes	0.7243 +/- 0.0167	0.72402 +/- 0.0167	0.7241 +/- 0.0167	0.7243 +/- 0.0167
SVM Method 1/4 Classes	0.6901 +/- 0.0153	0.6907 +/- 0.0152	0.6899 +/- 0.0152	0.6882 +/- 0.0148
SVM Method 1/5 Classes	0.6544 +/- 0.0164	0.6548 +/- 0.0165	0.6548 +/- 0.0165	0.6548 +/- 0.0165
SVM Method 2/2 Classes	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8723 +/- 0.0146
SVM Method 2/3 Classes	0.7273 +/- 0.0163	0.7262 +/- 0.0167	0.7251 +/- 0.0169	0.7238 +/- 0.0171
SVM Method 2/4 Classes	0.6917 +/- 0.0141	0.6931 +/- 0.0143	0.6896 +/- 0.0147	0.6903 +/- 0.0139
SVM Method 2/5 Classes	0.6517 +/- 0.0167	0.6506 +/- 0.0172	0.6526 +/- 0.017	0.6508 +/- 0.0173
	9	10		
SVM Method 1/2 Classes	0.8726 +/- 0.0145	0.8709 +/- 0.0151		
SVM Method 1/3 Classes	0.7242 +/- 0.0167	0.7241 +/- 0.0167		
SVM Method 1/4 Classes	0.6885 +/- 0.0148	0.6886 +/- 0.0148		
SVM Method 1/5 Classes	0.6544 +/- 0.0164	0.6543 +/- 0.0165		
SVM Method 2/2 Classes	0.8733 +/- 0.0142	0.8724 +/- 0.0146		
SVM Method 2/3 Classes	0.7256 +/- 0.0165	0.7254 +/- 0.0166		
SVM Method 2/4 Classes	0.6915 +/- 0.0139	0.691 +/- 0.0142		
SVM Method 2/5 Classes	0.6541 +/- 0.0167	0.6519 +/- 0.0171		

Table A.11: Overall average micro-F1 scores on the 20 Newsgroups UN dataset for LWLR using FCM, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on 20 Newsgroups UN, Using Top Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.8536 +/- 0.009	0.8614 +/- 0.009++	0.8414 +/- 0.0155	0.8628 +/- 0.0134
FCM/3 Classes	0.7675 +/- 0.0086	0.7949 +/- 0.0111	0.7725 +/- 0.0082	0.7652 +/- 0.0134
FCM/4 Classes	0.6652 +/- 0.0111	0.6975 +/- 0.0078	0.6925 +/- 0.0082	0.713 +/- 0.0064
FCM/5 Classes	0.5977 +/- 0.0089	0.6371 +/- 0.00801	0.6368 +/- 0.0073	0.6539 +/- 0.0058
	5	6	7	8
FCM/2 Classes	0.8787 +/- 0.0101	0.8786 +/- 0.0137	0.8859 +/- 0.0128	0.8958 +/- 0.0104
FCM/3 Classes	0.8009 +/- 0.0085	0.7918 +/- 0.0175	0.8066 +/- 0.0094	0.8152 +/- 0.0097
FCM/4 Classes	0.7162 +/- 0.009	0.72103 +/- 0.0078	0.7331 +/- 0.0079	0.736 +/- 0.0077
FCM/5 Classes	0.66901 +/- 0.0087	0.6659 +/- 0.0095	0.6869 +/- 0.0102	0.6897 +/- 0.00903
	9	10		
FCM/2 Classes	0.8972 +/- 0.0113	0.9068 +/- 0.0089		
FCM/3 Classes	0.8202 +/- 0.0091	0.8232 +/- 0.0084		
FCM/4 Classes	0.7452 +/- 0.00601	0.7519 +/- 0.0094		
FCM/5 Classes	0.6943 +/- 0.0092	0.6987 +/- 0.0094		

Table A.12: Overall average micro-F1 scores on the 20 Newsgroups UN dataset for SVM Method 1 & 2, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on 20 Newsgroups UN, Using Top Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.7703 +/- 0.0341	0.7334 +/- 0.0373	0.7681 +/- 0.0331	0.787 +/- 0.0327
SVM Method 1/3 Classes	0.6164 +/- 0.0316	0.6491 +/- 0.0353	0.6584 +/- 0.0374	0.6584 +/- 0.0314
SVM Method 1/4 Classes	0.5168 +/- 0.0346	0.5453 +/- 0.0304	0.5475 +/- 0.029	0.5451 +/- 0.0188
SVM Method 1/5 Classes	0.4631 +/- 0.0237	0.4861 +/- 0.023	0.4982 +/- 0.0232	0.4974 +/- 0.0203
SVM Method 2/2 Classes	0.841 +/- 0.0179	0.8506 +/- 0.0147	0.8721 +/- 0.0127	0.88403 +/- 0.0081
SVM Method 2/3 Classes	0.691 +/- 0.021	0.7704 +/- 0.0144	0.77304 +/- 0.00803	0.7798 +/- 0.0093
SVM Method 2/4 Classes	0.6079 +/- 0.0223	0.6634 +/- 0.0134	0.6749 +/- 0.0126	0.6938 +/- 0.011
SVM Method 2/5 Classes	0.5499 +/- 0.0144	0.612 +/- 0.0126	0.6108 +/- 0.0124	0.6379 +/- 0.0116
	5	6	7	8
SVM Method 1/2 Classes	0.7962 +/- 0.0334	0.8052 +/- 0.0342	0.8042 +/- 0.0284	0.807 +/- 0.0327
SVM Method 1/3 Classes	0.6672 +/- 0.0325	0.6693 +/- 0.0232	0.669 +/- 0.0275	0.6577 +/- 0.0319
SVM Method 1/4 Classes	0.5457 +/- 0.0207	0.5699 +/- 0.0244	0.5567 +/- 0.0278	0.5599 +/- 0.022
SVM Method 1/5 Classes	0.5122 +/- 0.0218	0.5172 +/- 0.0208	0.4955 +/- 0.0218	0.5094 +/- 0.0174
SVM Method 2/2 Classes	0.8937 +/- 0.0113	0.9076 +/- 0.0074	0.9031 +/- 0.0101	0.9131 +/- 0.0081
SVM Method 2/3 Classes	0.8049 +/- 0.0074	0.814 +/- 0.0074	0.8212 +/- 0.0092	0.8284 +/- 0.0062
SVM Method 2/4 Classes	0.7151 +/- 0.0067	0.7257 +/- 0.0083	0.7356 +/- 0.0058	0.7394 +/- 0.009
SVM Method 2/5 Classes	0.6685 +/- 0.0102	0.6826 +/- 0.0076	0.6914 +/- 0.0089	0.6965 +/- 0.0063
	9	10		
SVM Method 1/2 Classes	0.8142 +/- 0.0286	0.8126 +/- 0.0285		
SVM Method 1/3 Classes	0.6622 +/- 0.0234	0.6532 +/- 0.0347		
SVM Method 1/4 Classes	0.5651 +/- 0.0217	0.556 +/- 0.0263		
SVM Method 1/5 Classes	0.5107 +/- 0.02001	0.4968 +/- 0.0248		
SVM Method 2/2 Classes	0.9191 +/- 0.007	0.9224 +/- 0.0063		
SVM Method 2/3 Classes	0.8259 +/- 0.0079	0.8353 +/- 0.0081		
SVM Method 2/4 Classes	0.7427 +/- 0.0055	0.7612 +/- 0.0046		
SVM Method 2/5 Classes	0.7018 +/- 0.0083	0.7032 +/- 0.0073		

Table A.13: Overall average micro-F1 scores on the 20 Newsgroups UN dataset for LWLR using FCM, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on 20 Newsgroups UN, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.794 +/- 0.0142	0.7898 +/- 0.0176	0.7586 +/- 0.0213	0.7766 +/- 0.0178
FCM/3 Classes	0.6128 +/- 0.015	0.5979 +/- 0.0223	0.6064 +/- 0.0213	0.5942 +/- 0.022
FCM/4 Classes	0.5284 +/- 0.0207	0.5374 +/- 0.0178	0.5195 +/- 0.01704	0.5433 +/- 0.0182
FCM/5 Classes	0.4572 +/- 0.0141	0.46204 +/- 0.0104	0.4605 +/- 0.0174	0.4512 +/- 0.0167
	5	6	7	8
FCM/2 Classes	0.791 +/- 0.0164	0.7816 +/- 0.0193	0.7542 +/- 0.0199	0.7503 +/- 0.0221
FCM/3 Classes	0.6261 +/- 0.018	0.6116 +/- 0.0207	0.5983 +/- 0.0222	0.6188 +/- 0.0182
FCM/4 Classes	0.5266 +/- 0.0197	0.5081 +/- 0.0216	0.5307 +/- 0.0173	0.5121 +/- 0.0207
FCM/5 Classes	0.449++ +/- 0.018	0.4307 +/- 0.0226	0.4603 +/- 0.0149	0.4431 +/- 0.0207
	9	10		
FCM/2 Classes	0.7546 +/- 0.0217	0.7718 +/- 0.0137		
FCM/3 Classes	0.5992 +/- 0.0212	0.5885 +/- 0.0292		
FCM/4 Classes	0.493 +/- 0.02602	0.5267 +/- 0.0149		
FCM/5 Classes	0.4275 +/- 0.0213	0.4357 +/- 0.0223		

Table A.14: Overall average micro-F1 scores on the 20 Newsgroups UN dataset for SVM Method 1 & 2, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on 20 Newsgroups UN, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.7698 +/- 0.0315	0.7209 +/- 0.0363	0.7648 +/- 0.0268	0.7132 +/- 0.0364
SVM Method 1/3 Classes	0.4908 +/- 0.0403	0.5126 +/- 0.0312	0.5081 +/- 0.0307	0.5052 +/- 0.0456
SVM Method 1/4 Classes	0.3828 +/- 0.0272	0.4033 +/- 0.0375	0.391 +/- 0.0273	0.3993 +/- 0.0352
SVM Method 1/5 Classes	0.3539 +/- 0.0257	0.3047 +/- 0.0227	0.3222 +/- 0.0259	0.3149 +/- 0.0272
SVM Method 2/2 Classes	0.6584 +/- 0.0421	0.7092 +/- 0.0435	0.6941 +/- 0.0427	0.7512 +/- 0.0344
SVM Method 2/3 Classes	0.4919 +/- 0.0293	0.4988 +/- 0.0317	0.5274 +/- 0.0343	0.541 +/- 0.0358
SVM Method 2/4 Classes	0.4153 +/- 0.0289	0.4137 +/- 0.0255	0.4206 +/- 0.0287	0.4183 +/- 0.0277
SVM Method 2/5 Classes	0.3295 +/- 0.026	0.3224 +/- 0.0209	0.3297 +/- 0.0206	0.3357 +/- 0.0252
	5	6	7	8
SVM Method 1/2 Classes	0.7477 +/- 0.0358	0.7447 +/- 0.0331	0.7339 +/- 0.0377	0.7318 +/- 0.0369
SVM Method 1/3 Classes	0.4803 +/- 0.0341	0.5293 +/- 0.0388	0.5026 +/- 0.0383	0.4896 +/- 0.0415
SVM Method 1/4 Classes	0.4114 +/- 0.03704	0.408 +/- 0.0289	0.3844 +/- 0.0297	0.385 +/- 0.0319
SVM Method 1/5 Classes	0.3207 +/- 0.0222	0.3263 +/- 0.0244	0.3154 +/- 0.0277	0.3347 +/- 0.0241
SVM Method 2/2 Classes	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8726 +/- 0.0145	0.8723 +/- 0.0146
SVM Method 2/3 Classes	0.5465 +/- 0.0317	0.5428 +/- 0.0356	0.5272 +/- 0.0312	0.5336 +/- 0.0363
SVM Method 2/4 Classes	0.4376 +/- 0.0217	0.4316 +/- 0.02903	0.4066 +/- 0.0205	0.4219 +/- 0.0279
SVM Method 2/5 Classes	0.3434 +/- 0.0253	0.342 +/- 0.0189	0.3734 +/- 0.023	0.3376 +/- 0.0215
	9	10		
SVM Method 1/2 Classes	0.7352 +/- 0.0358	0.7709 +/- 0.0256		
SVM Method 1/3 Classes	0.5291 +/- 0.0432	0.4901 +/- 0.0333		
SVM Method 1/4 Classes	0.4212 +/- 0.0323	0.3899 +/- 0.0232		
SVM Method 1/5 Classes	0.3174 +/- 0.0242	0.324 +/- 0.0257		
SVM Method 2/2 Classes	0.7654 +/- 0.0344	0.7634 +/- 0.0286		
SVM Method 2/3 Classes	0.5574 +/- 0.029	0.5484 +/- 0.0402		
SVM Method 2/4 Classes	0.4099 +/- 0.0261	0.4037 +/- 0.0272		
SVM Method 2/5 Classes	0.3456 +/- 0.0211	0.3419 +/- 0.019++		

Table A.15: Overall average micro-F1 scores on the Modapte UN dataset for LWLR using FCM, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on Modapte UN, Using Top Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.8294 +/- 0.0129	0.8619 +/- 0.0122	0.8707 +/- 0.0147	0.8823 +/- 0.0147
FCM/3 Classes	0.6767 +/- 0.0095	0.6919 +/- 0.0144	0.7332 +/- 0.009++	0.7503 +/- 0.0159
FCM/4 Classes	0.6283 +/- 0.0156	0.6878 +/- 0.0116	0.7149 +/- 0.0151	0.7207 +/- 0.0125
FCM/5 Classes	0.6615 +/- 0.0107	0.6918 +/- 0.0125	0.7337 +/- 0.0107	0.7562 +/- 0.011
	5	6	7	8
FCM/2 Classes	0.8832 +/- 0.0153	0.8775 +/- 0.0145	0.8855 +/- 0.0175	0.8919 +/- 0.0211
FCM/3 Classes	0.7746 +/- 0.0082	0.7627 +/- 0.0113	0.7847 +/- 0.0128	0.7917 +/- 0.0125
FCM/4 Classes	0.7274 +/- 0.0152	0.7499 +/- 0.0124	0.7535 +/- 0.0114	0.7661 +/- 0.0113
FCM/5 Classes	0.7601 +/- 0.0105	0.7855 +/- 0.0113	0.7867 +/- 0.0136	0.7943 +/- 0.0123
	9	10		
FCM/2 Classes	0.8874 +/- 0.0144	0.8904 +/- 0.0169		
FCM/3 Classes	0.7994 +/- 0.0144	0.797 +/- 0.0122		
FCM/4 Classes	0.7878 +/- 0.0138	0.7724 +/- 0.0142		
FCM/5 Classes	0.7943 +/- 0.0128	0.7893 +/- 0.0131		

Table A.16: Overall average micro-F1 scores on the Modapte UN dataset for SVM Method 1 & 2, using top user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on Modapte UN, Using Top Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.8333 +/- 0.0413	0.8866 +/- 0.0206	0.888 +/- 0.026	0.8631 +/- 0.0348
SVM Method 1/3 Classes	0.6564 +/- 0.0157	0.6421 +/- 0.0257	0.6772 +/- 0.0254	0.6838 +/- 0.0215
SVM Method 1/4 Classes	0.61703 +/- 0.0185	0.6079 +/- 0.0308	0.6219 +/- 0.0195	0.6453 +/- 0.0209
SVM Method 1/5 Classes	0.6075 +/- 0.0215	0.6029 +/- 0.0281	0.6213 +/- 0.0198	0.6463 +/- 0.0223
SVM Method 2/2 Classes	0.8849 +/- 0.0144	0.908 +/- 0.0116	0.9162 +/- 0.0098	0.9182 +/- 0.008
SVM Method 2/3 Classes	0.7032 +/- 0.0119	0.7241 +/- 0.0079	0.7499 +/- 0.008	0.7703 +/- 0.0083
SVM Method 2/4 Classes	0.6362 +/- 0.0158	0.6789 +/- 0.0105	0.7075 +/- 0.0098	0.7172 +/- 0.0087
SVM Method 2/5 Classes	0.6457 +/- 0.0176	0.6819 +/- 0.0132	0.7108 +/- 0.009++	0.7405 +/- 0.0081
	5	6	7	8
SVM Method 1/2 Classes	0.8972 +/- 0.0254	0.8857 +/- 0.0171	0.8614 +/- 0.0239	0.8891 +/- 0.0143
SVM Method 1/3 Classes	0.6898 +/- 0.0313	0.6831 +/- 0.0169	0.6881 +/- 0.0216	0.7156 +/- 0.0203
SVM Method 1/4 Classes	0.6351 +/- 0.0213	0.6453 +/- 0.0208	0.6754 +/- 0.0186	0.6563 +/- 0.0312
SVM Method 1/5 Classes	0.6551 +/- 0.0155	0.6694 +/- 0.0222	0.6846 +/- 0.01601	0.6604 +/- 0.025
SVM Method 2/2 Classes	0.9242 +/- 0.00404	0.9205 +/- 0.0054	0.9211 +/- 0.00404	0.9286 +/- 0.003
SVM Method 2/3 Classes	0.7806 +/- 0.0069	0.7868 +/- 0.0055	0.8021 +/- 0.0071	0.8162 +/- 0.005
SVM Method 2/4 Classes	0.7306 +/- 0.011	0.7504 +/- 0.0084	0.749 +/- 0.008	0.764 +/- 0.0063
SVM Method 2/5 Classes	0.7558 +/- 0.0074	0.7805 +/- 0.0058	0.7798 +/- 0.0068	0.7894 +/- 0.0059
	9	10		
SVM Method 1/2 Classes	0.8751 +/- 0.0269	0.9058 +/- 0.0096		
SVM Method 1/3 Classes	0.7095 +/- 0.0263	0.7243 +/- 0.0186		
SVM Method 1/4 Classes	0.6828 +/- 0.02003	0.6701 +/- 0.0205		
SVM Method 1/5 Classes	0.6561 +/- 0.0274	0.6667 +/- 0.0254		
SVM Method 2/2 Classes	0.9296 +/- 0.0051	0.9323 +/- 0.0045		
SVM Method 2/3 Classes	0.8187 +/- 0.0037	0.8176 +/- 0.0048		
SVM Method 2/4 Classes	0.7833 +/- 0.0057	0.7815 +/- 0.0064		
SVM Method 2/5 Classes	0.7887 +/- 0.0057	0.8034 +/- 0.0057		

Table A.17: Overall average micro-F1 scores on the Modapte UN dataset for LWLR using FCM, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

LWLR with FCM, Overall Average Micro-F1 Scores on Modapte UN, Using Random Features

# of Features per Class → / *Classifier/# of Classes ↓*	1	2	3	4
FCM/2 Classes	0.7366 +/- 0.0206	0.7492 +/- 0.02703	0.72503 +/- 0.0226	0.7267 +/- 0.0297
FCM/3 Classes	0.6459 +/- 0.0194	0.6115 +/- 0.0218	0.6039 +/- 0.0274	0.6192 +/- 0.0254
FCM/4 Classes	0.6066 +/- 0.0279	0.6294 +/- 0.0193	0.6064 +/- 0.0223	0.5891 +/- 0.0195
FCM/5 Classes	0.6096 +/- 0.0195	0.6072 +/- 0.0182	0.6043 +/- 0.0215	0.586 +/- 0.0235
	5	6	7	8
FCM/2 Classes	0.7068 +/- 0.0254	0.7083 +/- 0.028	0.7069 +/- 0.0216	0.7233 +/- 0.0226
FCM/3 Classes	0.6249 +/- 0.0208	0.6074 +/- 0.0287	0.6063 +/- 0.0258	0.5947 +/- 0.0213
FCM/4 Classes	0.5869 +/- 0.025	0.5774 +/- 0.0245	0.5746 +/- 0.0182	0.5919 +/- 0.0219
FCM/5 Classes	0.5712 +/- 0.028	0.5849 +/- 0.0303	0.5669 +/- 0.0277	0.5486 +/- 0.0288
	9	10		
FCM/2 Classes	0.6961 +/- 0.0249	0.7222 +/- 0.0271		
FCM/3 Classes	0.5923 +/- 0.0247	0.591 +/- 0.02403		
FCM/4 Classes	0.5783 +/- 0.0238	0.5679 +/- 0.0206		
FCM/5 Classes	0.54901 +/- 0.0329	0.5367 +/- 0.0279		

Table A.18: Overall average micro-F1 scores on the Modapte UN dataset for SVM Method 1 & 2, using random user features. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

SVM Method 1 & 2, Overall Average Micro-F1 Scores on Modapte UN, Using Random Features

# of Features per Class → *Classifier/# of Classes ↓*	1	2	3	4
SVM Method 1/2 Classes	0.8546 +/- 0.0321	0.8713 +/- 0.0265	0.8593 +/- 0.0261	0.8584 +/- 0.0244
SVM Method 1/3 Classes	0.6537 +/- 0.0232	0.6497 +/- 0.0285	0.647 +/- 0.0308	0.6325 +/- 0.0283
SVM Method 1/4 Classes	0.588 +/- 0.0285	0.6204 +/- 0.0263	0.6068 +/- 0.0356	0.6077 +/- 0.0288
SVM Method 1/5 Classes	0.5517 +/- 0.0344	0.5497 +/- 0.0315	0.5789 +/- 0.0275	0.5357 +/- 0.0474
SVM Method 2/2 Classes	0.8536 +/- 0.0248	0.8643 +/- 0.0307	0.8579 +/- 0.0341	0.8772 +/- 0.0155
SVM Method 2/3 Classes	0.6563 +/- 0.0212	0.6657 +/- 0.02601	0.6671 +/- 0.0212	0.6386 +/- 0.0212
SVM Method 2/4 Classes	0.5985 +/- 0.0259	0.6233 +/- 0.0262	0.6223 +/- 0.0306	0.59701 +/- 0.0295
SVM Method 2/5 Classes	0.5784 +/- 0.0318	0.5884 +/- 0.0252	0.5993 +/- 0.0233	0.5989 +/- 0.0195
	5	6	7	8
SVM Method 1/2 Classes	0.8754 +/- 0.0226	0.8724 +/- 0.0229	0.8536 +/- 0.0306	0.8759 +/- 0.0164
SVM Method 1/3 Classes	0.6751 +/- 0.0266	0.6475 +/- 0.0297	0.6457 +/- 0.0224	0.6522 +/- 0.0295
SVM Method 1/4 Classes	0.588 +/- 0.0285	0.6204 +/- 0.0263	0.6068 +/- 0.0356	0.6077 +/- 0.0288
SVM Method 1/5 Classes	0.5335 +/- 0.0349	0.5181 +/- 0.0431	0.5611 +/- 0.0236	0.5268 +/- 0.0496
SVM Method 2/2 Classes	0.8945 +/- 0.0099	0.8681 +/- 0.0196	0.884 +/- 0.01201	0.8577 +/- 0.0273
SVM Method 2/3 Classes	0.6549 +/- 0.0171	0.6681 +/- 0.0167	0.6316 +/- 0.0244	0.6391 +/- 0.0189
SVM Method 2/4 Classes	0.6037 +/- 0.0245	0.5897 +/- 0.0266	0.6074 +/- 0.0179	0.5956 +/- 0.0286
SVM Method 2/5 Classes	0.5552 +/- 0.0321	0.5831 +/- 0.0294	0.5675 +/- 0.0282	0.5666 +/- 0.0301
	9	10		
SVM Method 1/2 Classes	0.8728 +/- 0.0176	0.862 +/- 0.0292		
SVM Method 1/3 Classes	0.642 +/- 0.0214	0.6877 +/- 0.0277		
SVM Method 1/4 Classes	0.5908 +/- 0.0354	0.5914 +/- 0.0368		
SVM Method 1/5 Classes	0.5844 +/- 0.0265	0.5848 +/- 0.0244		
SVM Method 2/2 Classes	0.87503 +/- 0.0214	0.8702 +/- 0.0267		
SVM Method 2/3 Classes	0.6564 +/- 0.0215	0.6577 +/- 0.0236		
SVM Method 2/4 Classes	0.5809 +/- 0.0249	0.6021 +/- 0.0216		
SVM Method 2/5 Classes	0.6114 +/- 0.0276	0.5865 +/- 0.0277		

A.2 Autocoder Tests

Table A.19: Overall average micro-F1 scores on the Autocoder dataset. The confidence intervals were computed over the average macro-F1 scores used in the overall average.

Overall Average Micro-F1 Scores on Autocoder Dataset

	Non-Sequential	*Sequential*
Logistic Regression	.36 +/- .0224	.36 +/- .0224
COSIM	.3242 +/- .0281	.324 +/- .0281
FCM	.3379 +/- .0276	.3362 +/- .0268
SVM	.3614 +/- .0208	.3387 +/- .0197
Method 1	.3969 +/- .0226	.3405 +/- .0216
Method 2	.3663 +/- .0202	.3506 +/- .0224