GroupTime: Probabilistic Scheduling

Kendra Carattini and Mike Brzozowski

Introduction

Perhaps one of computer-supported cooperative work (CSCW)'s greatest successes of the past decade has been group scheduling. But virtually all major groupware systems available today present a binary view of group calendaring: users are either free or they are busy. This presents the false appearance that all of a user's free time is equally free. In reality, not all "available" times are equally free—people often prefer to keep large blocks of free time open for work, or try to avoid scheduling nonessential meetings the night before a large assignment is due, for instance.

User Group

A prime example of a user segment whose calendars are difficult to predict is college students. Without a clearly defined Monday to Friday, nine-to-five work week, many students find they need to be available to work at almost any time—day or night, weekday or weekend. At the same time a plethora of commitments compete for students' time, from classes and studying to jobs, interviews, meetings, and rehearsals to parties and dates. Many students must coordinate their schedules with others on a regular basis to arrange team meetings, whether for class or for outside activities. And yet each user has their own set of priorities when it comes to resolving the inevitable scheduling conflicts that arise.

Related Work

Probabilistic models have been successfully employed before to predict user availability. Horvitz, Koch, Kadie, and Jacobs used a Bayesian network to forecast a user's presence and availability, based on calendar information, sensors at his/her desk, and the location of various mobile devices [1]. This operated more on the short term, specializing in predicting, for instance, how soon a colleague who just stepped out of the office would return. And Joe Tullio used Bayesian networks to estimate the likelihood of a user attending a given meeting based on empirical evidence about which meetings he/she attended in the past, hoping to improve group calendar accuracy [2]. However, both projects were specific to relatively more constrained office environments; as far as we know no one has yet applied probabilistic methods to more erratic student schedules.

In our earlier study (Brzozowski and Carattini [3]), a system was built to model what times would most likely be convenient for students to schedule their commitments. The model was then used to predict the optimum meeting times for a user based on implicit user preferences for how he/she likes to schedule free time. Over time, such a system would adapt to its user's unique scheduling priorities and preferences.

Model Ontology and Training Data

Following the model ontology in our earlier study (see selection from Brzozowski and Carattini, "Probabilistic Group Scheduling for Chaotic People" in Appendix A for full details of the model ontology), we asked a group of students to provide us with the details of their commitments over a period of one week. We then proposed various meeting times and types over the course of the week, and asked the users to classify their availability for the given appointment into one of four categories: "Can't make it",

"rather not", "is ok", and "works great". We extracted features from this data that corresponded to things like the category of the meeting to be scheduled (e.g. was it a lecture or a dance rehearsal?), the day of week and time of day of the meeting, the events that the meeting conflicted with, and the amount of time before and after the meeting to the next scheduled event. We then trained our model use softmax regression for the 4 possible labels.

Methodology and Discussion

The goal of the early study was to model and build a system for a very small number of people that supplied large data sets so that the system could be fine-tuned for an individual. We found, however, that although we obtained mild success using cross validation on a single user's data set, when we learned and tested on different users, the error rate was very high. The goal of our present study is to now use a much larger set of users (46 users in total with 200 data points each) and determine the best method of transfer learning for determining a prediction on a user with a sparse data set. The utility of this is obvious: when a user begins using the system, there will be little or no data to reflect the individual's preferences. We can, however, still make useful predictions for a person for which we have no training data, as some principles hold true across many users. For example, most of the users in our sample did not wish to schedule anything between 3am to 6am on any day. Simply training on all user data sets may not be the best way to represent these trends, as an initial 70:30 cross validation over all 46 users gave an error rate of 44.5%.

The first method of transfer learning attempted was to first learn the weights for each individual user, and then cluster the users according to their weights. The idea behind this method is that people who like to schedule things in a similar way will have similar weights, and therefore cluster together. The cross validation technique was then repeated on each of the individual clusters. The initial clustering was hierarchical (see Fig. 1), to obtain an overall view of the data and determine a good value for the number of clusters to use.

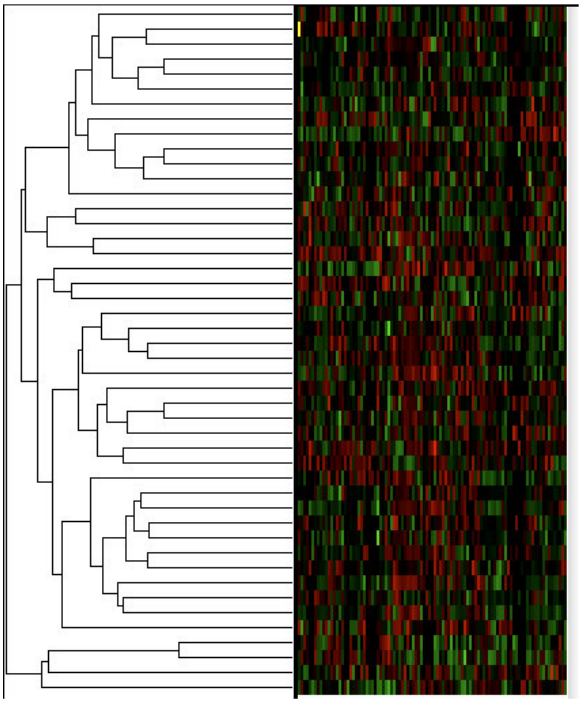

Fig. 1 Initial hierarchical clustering of 46 user study. Each data set is a row and each feature weight is a column. Red corresponds to negative weights and green to positive weights.

We determined the best number of clusters for the training set to be six, and proceeded to cluster the data into six clusters using k-means. The clusters can be seen in Fig. 2.

Cluster #	Number of data sets	Cross Validation Error Rate	Off By 1	Off By 2	Off By 3
1	9	0.50626	0.37209	0.094812	0.039356
2	1	0.68852	0.39344	0.16393	0.13115
3	11	0.46356	0.35131	0.086006	0.026239
4	6	0.36856	0.22764	0.086721	0.054201
5	10	0.43065	0.24839	0.12258	0.059677
6	9	0.41301	0.26714	0.10896	0.036907
All Data sets	46	0.44507	0.28647	0.12222	0.036383

Table 1. Error rates for clusters given by K-means algorithm. "Off by 1" indicates the predicted class was one class away from the correct class (e.g. "rather not" is one class away from "can't make it").

We then repeated the 70:30 cross validation on each of the individual clusters, and obtained the error rates show in Table 1.

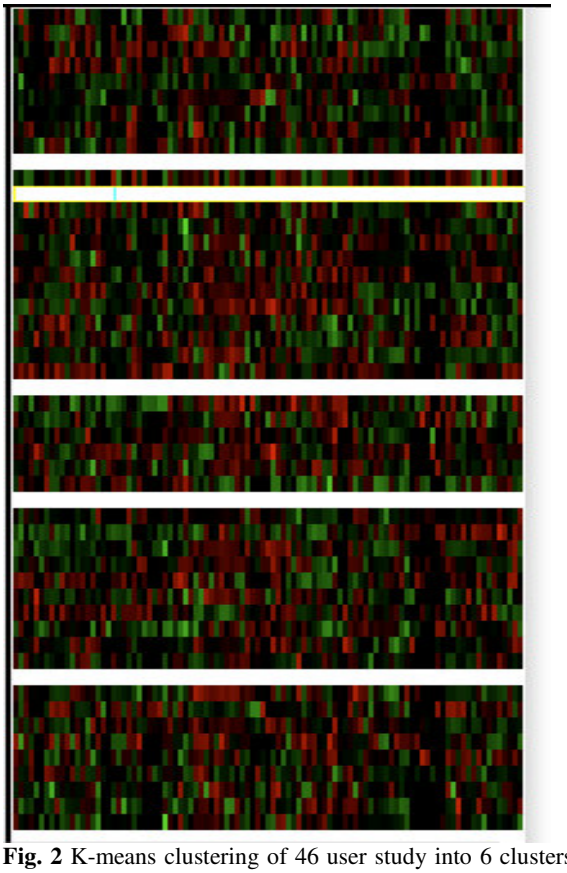

Fig. 2 K-means clustering of 46 user study into 6 clusters. Each data set is a row and each feature weight is a column. Red corresponds to negative weights and green to positive weights.

The results are mixed. Three of the clusters showed an increase in error rate over the entire user set-trained system, whilst the other three showed a decrease. One cluster in particular (cluster 4) showed a very large decrease in error rate. Although these error rates are high, we can see that most of the errors are "off by one", i.e. the classification predicted by our algorithm was one class away from the actual class (e.g. "Works great" is one class away from "Is ok"). The error rate from Cluster 2, which only had one user's data set, was extremely high. This result shows the need for a good transfer learning algorithm, as learning over a single user's data provides the algorithm with insufficient data to accurately predict a scheduling preference.

The results from this initial transfer learning method were disappointing, in that there was on average no decrease in the cross validation error rate over the baseline using all data points. The next step was to investigate alternative methods of transfer learning in the hopes of attaining decreased cross-validation error rates.

An initial attempt at simulating online learning using the perceptron algorithm yielded the results shown in Table 2. The class labels were modified so that "is ok"

and "works great" were combined into one class, and "rather not" and "can't make it" were combined into another.

Data Set	# Training Points Used in Online Portion of Learning	Error Rate
KTPH	200	0.495
W15P	200	0.555
All	9200	0.41937
All	200	0.505

Table 2. Error rates for data sets using the perceptron algorithm for online learning.

Only one result showed a minor improvement over the baseline softmax cross-validation error. The trial first trained on 9000 data points using the perceptron algorithm, and then proceeded to use the online learning update for the next 200 data points (which were all from the same user). The rest of the results showed little to know improvement over random guessing (0.5 error rate), and so this method of learning was not pursued further.

The third method of transfer learning investigated was weighted regression. Four different kinds of weighting schemes were investigated on the six clusters:

1) The weights were calculated according to the Euclidean distance between them using the formula:

$$w^{(i)} = \exp\left(-\frac{\sum_j (x_j^{(i)} - x_j)^2}{2\tau^2}\right) \qquad (1)$$

2) The weights were calculated using the time of day and day of week features only. Times of day and days of week closer to the point in question were weighted more heavily. Since the feature vectors contained mostly Boolean values and had very high dimension, the Euclidean distance between many of the vectors was nearly identical. For this reason, we decided to use a distance metric based on only a few key features that were not Boolean values.

3) The third weighting scheme gave a training point a weight of 1.0 if the query point was targeted for the same user as the training point came from. If the user was different, the training point was given a weight of 0.5.

4) The fourth weighting scheme was a combination of the first and third schemes.

The weighted regression was run on each of the smaller clusters using both 70/30 and 90/10 cross-validation. The results are shown in Table 3.

Cluster	Weighting Scheme	70/30 CV	90/10 CV	Error Rate	Off By 1	Off By 2	Off By 3
1	3	x		0.44361	0.31391	0.097744	0.031955
1	1		x	0.51053	0.31579	0.15263	0.042105
1	2		x	0.42781	0.30481	0.090909	0.032086
1	3		x	0.42246	0.28877	0.1123	0.02139
1	4		x	0.41711	0.28342	0.10695	0.026738
2	1	x		0.54545	0.34545	0.10909	0.090909
2	2	x		0.58182	0.34545	0.16364	0.072727
4	1	x		0.33523	0.2017	0.079545	0.053977
4	2	x		0.32955	0.18182	0.085227	0.0625
4	4	x		0.31534	0.17898	0.079545	0.056818
4	2		x	0.33858	0.17323	0.14961	0.015748
4	3		x	0.35878	0.21374	0.1145	0.030534
4	4		x	0.27559	0.15748	0.10236	0.015748
6	4	x		0.34586	0.22556	0.10338	0.016917

Table 3. Error rates for clusters using 4 different weighted regression schemes. Scheme 1 weighted according to Euclidean distance, 2 according to distance in the time of day and day of week dimensions only, 3 according to the user, and 4 was a combination of 1 and 3.

We can see that, with one exception, the error rate decreased from the baseline for each cluster using the un-weighted regression. The combination weight scheme (#4) consistently gave the lowest error rates of all the weighting schemes, but there does not appear to be a straightforwardly discernable hierarchy of weighted regression schemes. It is not surprising that the combined weight scheme gave the lowest error rate, since this scheme takes into account both how closely related two data points are (distance) and how their sources are linked (same or different users).

These results indicate that weighted regression within a cluster would be a useful way to learn weights for an individual that would make use of other users training data as well, and thereby require fewer training points for accurate predictions from the individual in question. The online learning method did not prove very promising for this task, nor did the clustering on its own. Future work in this area should therefore concentrate on finding the best weighted regression scheme using a minimal number of data points from an individual user, as this will be the actual task faced in a groupware scheduling application. Other areas of interest that will arise in such a setting include learning with an incomplete set of features (e.g. if the user does not provide a full calendar), and taking into account the person who initiates the meeting invitation (e.g. a boss vs. a friend).

References

1. Horvitz, E., Koch, P., Kadie, C. M., and Jacobs, A. Coordinate: Probabilistic Forecasting of Presence and Availability. Proceedings of the Eighteenth Conference on Uncertainty and Artificial Intelligence, Edmonton, Alberta. Morgan Kaufman (2002), 224-233.

2. Tullio, J. Intelligent Groupware to Support Communication and Persona Management. ACM Symposium on User Interface Software and Technology (Doctoral Consortium). (2003)

3. Brzozowski, M., and Carattini, K. Probabilistic Group Scheduling for Chaotic People. CS 229, Stanford University, Fall (2004).

Appendix A: selection from:

Probabilistic Group Scheduling For Chaotic People

Mike Brzozowski **Kendra Carattini**

MODEL ONTOLOGY

We believe that the probability that a user will want to schedule meeting M at time T, where T is a span of time on one or two specific days, is dependent on:

- The category C of M, that is, what type of meeting it is. Table 1 shows the categories we chose for this domain, which capture a range of pressures (social, academic, professional, economic, etc.) to attend various types of events.
- The time of week T, encapsulating both a day and time.
- The presence of other events E_O overlapping with T and their corresponding categories.
- The length of time dT_- between the end of preceding event E_- and the start of T.
- The length of time dT_+ between the end of T and the start of the following event E_+.

Category	Description	Example from user data
Interest	An optional event the user might like to attend out of interest, not because friends will be present	Psychology professor's talk on politics of fear and Iraq prison abuses
Interview	A job interview.	Phone interview with Yahoo!
Lecture	A lecture for a class the user is taking, attendance at which is not strictly required.	Ling 238 lecture
Practice	Practice for a sports team or group.	Tae kwon do practice
Project	A meeting to work on a team class project	CS 221 class project
Rehearsal	A rehearsal for a performance group or show.	Viennese Ball Opening Waltz rehearsal
Section	An optional discussion section for a class.	CS 229 section
Seminar	A small seminar with required attendance.	CS 376 seminar
Sleep	Time the user plans to be asleep.[1]	Six hours preceding morning rehearsal
Social-private	A social event with a small group of friends; peer pressure encourages the user to attend.	Thanksgiving dinner at a friend's apartment
Social-public	A social event with a large group of people, most of whom will not notice if the user is absent.	Ragtime Ball
Study	Time the user plans to be studying.	Study for CS 221 midterm

[1] We discovered that not all users view sleep as an inflexible time commitment but believe our model should reflect that some users don't mind scheduling meetings far into their "sleep" hours.

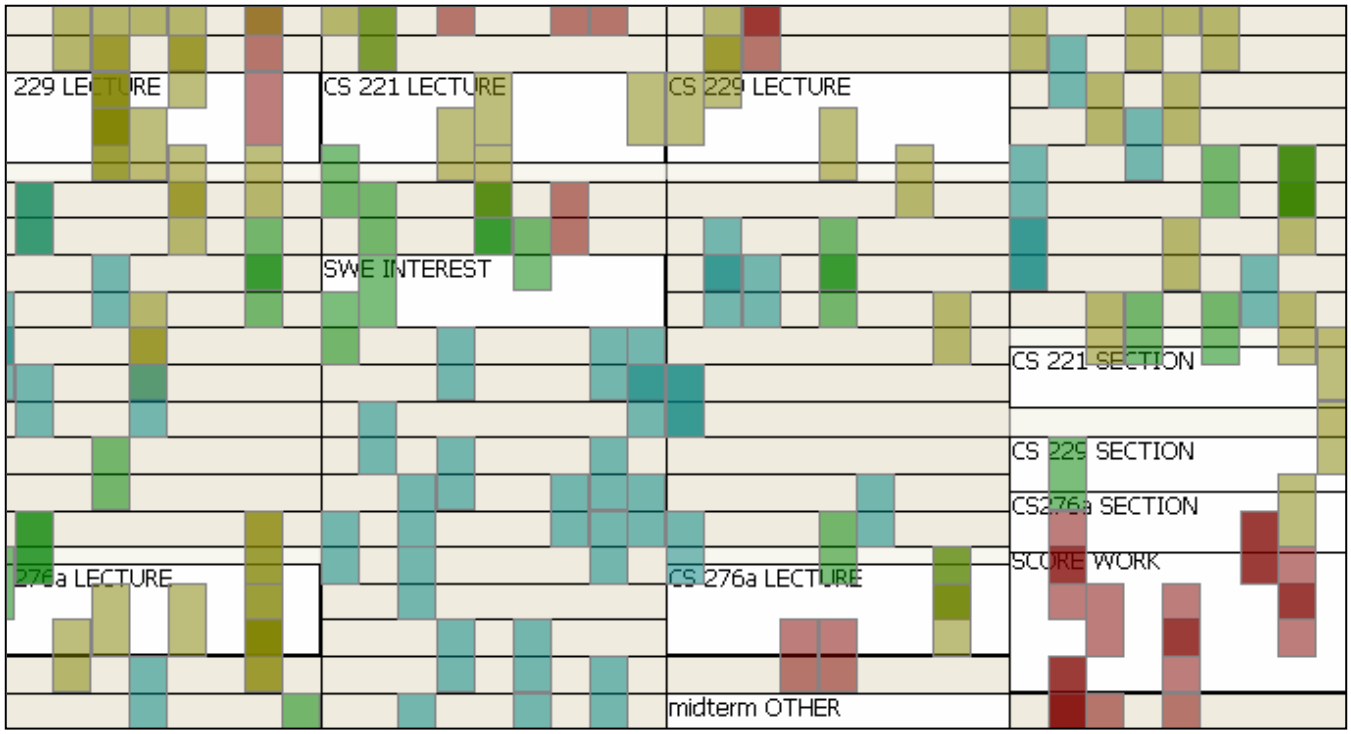

Figure 1 A sample of data collectd from a test subject. White boxes are prior scheduled commitments; red boxes are times labeled *Can't Make It*; yellow ones are *Rather Not*; green ones are labeled *Is OK*; blue ones are *Works Great*. The four large columns represent four days of the week; the columns within each day correspond to different hypothetical commitment categories. Notice the subject is more willing to miss events marked INTEREST and has no flexibility to get out of WORK.

Study-group	A meeting with a group not obligated to work together, to study.	CS 229 problem set 2 study session
Work	Work as part of a paying job.	Course advisor office hours

Table 1 Possible values for category *C* in data collected

TRAINING DATA

We gathered training data by soliciting subjects' complete schedules for the following week and encoding a series of events according to our ontology. We then proceeded to ask subjects to consider a series of hypothetical meetings randomly selected from categories the subject is likely to encounter (e.g. users who do not participate in sports or performing arts are not asked about extra practices or rehearsals; users are not invited to pick a time to meet to sleep). Subjects were given a graphical representation of the schedule they supplied us with the suggested times highlighted. They were asked to assume that they wanted to schedule *M* at some point over the week and to label each prospective hour-long slot *T* with one of four options:

- *Can't Make It:* The subject cannot attend *M* at *T* under any circumstances.
- *Rather Not:* The subject would rather not attend at *T* but could if necessary.
- *Is OK:* The subject could meet at *T* but there are other times that work better.
- *Works Great: T* is one of the best times for the subject to meet.

Currently we have 1600 data points obtained from nine subjects, representing a variety of preferences and some variations in priorities. The subjects are undergraduate and graduate students in both engineering and humanities disciplines.

A sample of the data collected from one subject is shown in Figure 1.

FEATURE CALCULATIONS

After our early trials with logistic regression, it became apparent that the time of day, day of week, and time until the events before and after the hypothetical meeting were the most highly

weighted in our hypothesis function. However, this achieved poor results with high error over our training data. We hypothesized this may be due to the fact that there is not a linear relationship between some features and the likely classifications. We therefore considered alternate possibilities for these highly weighted features that would more accurately represent the user's preferences.

Day and Time

Initially, the time of day feature was simply a value between 0 and 47, representing the half-hour slot when the hypothetical meeting would begin. But the most favorable times are not necessarily early or late in the day. So we calculated the expected value of the convenience labeling over each time slot over the entire week. Similarly, we modified the day of week feature, which had initially been a value between zero and six to return the expected value of convenience calculated for each day over every time slot.

Due to the sparseness of the training data, we implemented a smoothing algorithm to avoid zero probabilities and ensure that slots with few examples were not biased. To do this, instead of taking the exact expectation over each time slot (and day of week), we calculated a locally weighted average of the expectation over all time slots (and days of week). These adjustments increased the accuracy of our learned hypothesis on the training data slightly; a sample of the final results can be seen in Figure 2.

Times Between Events

The initial features storing the times in between events (dT_- and dT_+) were simply the raw values of dT_- and dT_+ in minutes. This formulation seemed reasonable, as it favors a positive response when the hypothetical event occurs far away from other events in the schedule. We soon realized, however, that this scheduling preference did not generalize from the training data, but rather from our own pre-suppositions about how people schedule their time. After observing more training data, we noticed that a significant proportion of our test subjects actually preferred to schedule their events close together, thereby leaving large blocks of free time for other purposes. In order to take this into account, we adopted a similar approach as the time of day and day of week features. However, since the dT's are continuously valued, it is implausible to directly calculate the expected value over every possible dT. Instead, we fit the data to 2nd through 6th-order polynomial regressions (using a 70:30 cross validation split), and averaged the mean-squared error over ten trials for each order polynomial.

We chose cubic regression as our feature function because it had one of the lowest generalization errors, and did not appear to overfit the training data. As an additional feature, we decided to experiment with a combined regression over the joint distribution of dT_- and dT_+. This cubic regression proved to have an even lower generalization error, and was included in the final feature set. This seems intuitive if we consider people trying to schedule their meetings in the "cracks" in their schedule, i.e. places where both dT_- and dT_+ are low, but not where one of them is low and the other high. Thus, it makes sense that these features are not necessarily independent. After

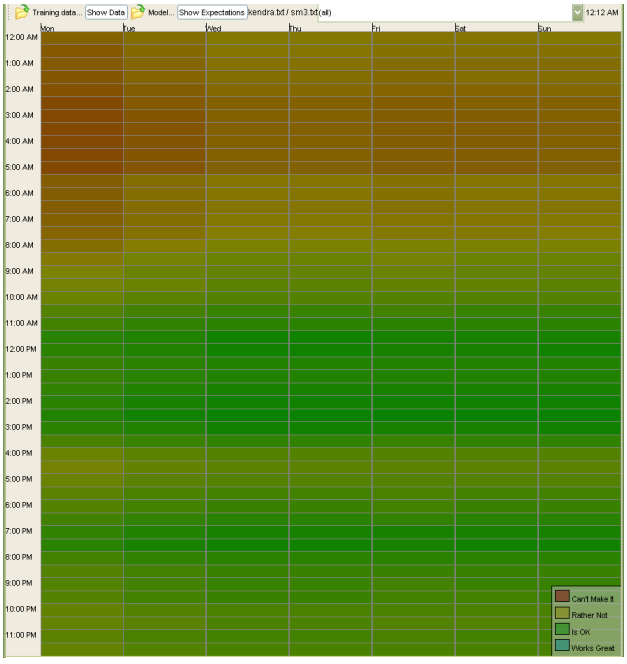

Figure 2 Distribution of E[Label | Day, Time] for a subject (see Figure 1 for color coding). For this subject, the best times to meet tend to be between 11 AM and 3 PM.

altering the dT feature functions to reflect the user responses, we were able to obtain 86% accuracy on our training data using logistic regression.

Conflicts

The conflict category feature(s) posed an interesting dilemma: Choosing a number to reflect the category of a conflict would incorrectly suppose a continuous relationship between conflict categories, when actually the categories are either independent or their preferential ordering is indeterminable *a priori*. To overcome this, we implemented a separate feature for each potential conflict category. For each query, a Boolean array was built to indicate which categories the query conflicted with (note this number can be greater than one, since we can conflict with multiple events for a given query). The feature for each category was then '1' if the query conflicted with an event in this category, and '0' otherwise. This formulation seemed unsatisfactory, since it blew up the size of our feature set so that a different weight had to be trained on each conflict category. In order to improve this, we tried to use the same expectation trick we described earlier.

SOFTMAX REGRESSION

We still felt that this did not adequately capture the midrange of possible labels (*Rather Not*/*Is OK*), the chief area of interest, since we seek to provide nonbinary classification. Our next step was to implement softmax regression. Using softmax, we were able to classify the responses into each of the four classes, rather than arbitrarily splitting the response space in half. This yielded encouraging results, and errors typically took the form of misclassifying into neighboring labels (e.g. *Is OK* into *Works Great* instead of *Can't Make It*).

Conflicts Revisited

We attempted to implement a "conflict category" feature that would return the expected value of the response, given training data with that conflict. Categories that did not conflict with any of the training data returned a smoothed expected response over all conflict categories, and the "no-conflict" category returned the smoothed expected value over all non-conflicting queries in the training data. The features were smoothed by taking a fraction (lambda) of the expected value over the conflict category and summing with (1 – lambda)*Expected value over all conflict categories. However, adding this feature actually decreased the accuracy of our hypothesis function. This was a rather surprising result, since the similar formulations over the other features had all increased accuracy.
One possible reason for this may be that the weights being learned for each of the label classifiers themselves don't bear a linear relationship with, say, P(*Rather Not*). Thus, if the classifier were to attach a positive weight to the expectation it would attach an even greater weight to an expectation closer to *Works Great*. Thus, our estimate of the expectation does not necessarily contribute to the correct P(*Y*)s we'd expect.