ble and hence disliked the stumble. Again, we studied the user timespent pattern and found that retained users indeed spend more time across k sessions as compared to the non-retained users. Figure 4 shows the cumulative timespent across three sessions for the retained and non-retained set. It should be noted that for brevity the x-axis and y-axis are log-scaled. As can be seen, there are more non-retained users spending less than or equal to $0.5x$ units of time compared to the retained users. Similarly, a larger number of retained users spend more than $0.5x$ units of time as compared to non-retained users. In essence, given a fixed number of sessions, retained users tend to spend more time compared to the non-retained users.

4.3 Timespent on Engaged Stumbles

For a user to like the stumble, the user first needs to process the stumble. One way to determine whether the user processed the URL is to find if the user spent sufficient time required to process that URL. We start by choosing a fixed timespent threshold for all the stumbles. In this fixed threshold case, an URL is considered as processed, if the user spends more time than threshold τ. We experimented with different values of threshold ranging from 0 secs to l secs. We found that the threshold of $\tau = p$ [1] seconds gave us the best split for the retained and non-retained users. The best split was found by training a decision tree on a part of data and testing it on a hold out test set.

Larger the number of engaged stumbles in k sessions, higher is the chance that the user liked the stumbles and hence higher is the chance of the user being retained. The number of engaged stumbles in a session could be one of the predictor for user retention. The predictors discussed above are used as baseline predictors (metrics) at StumbleUpon currently. In the succeeding sections, we discuss new predictors and compare them to the baseline predictors.

4.4 Issues with the earlier predictors

Although the above measures provide a fair indication of user retention based on the timespent data, they fail on various accounts. For instance, user's may skip (move to next stumble, without processing) a lot of stumbles. So, higher number of stumbles may not necessarily be a good predictor of user satisfaction and in turn retention. Further, although the cumulative timespent takes into account the user's total timespent in the session, it does not capture whether the user was engaged in processing the individual stumbles/URLs. The engaged stumble metric tries to overcome this issue by using only the time spent by the user when he/she was engaged in processing the stumble. However, the engaged stumble metric discussed in section 4.3 is very simplistic; one that uses the same timespent threshold irrespective of the content type. For instance, if a stumble is just an photo URL, it is likely to take a lot less than p secs to process that image by the user. On the other hand, if a stumble is a 10 minutes long video, and the user just spends 50 seconds on it, probably means that the user did not process the stumble.

5. FINDING ATTENDED STUMBLES

As mentioned in the last section, the engaged stumbles measure assumes a fixed threshold for all the stumbles. We can devise a better metric that accounts for content type, rather than keeping a fixed threshold for all the content types. We label a stumble *attended* if the user's timespent on the URL is more than or equal to the typical time required to process the URL. We use the term *attended* in place of *engaged*, just to differentiate these methods from the earlier fixed threshold version. The motivation behind this method is that more the number of attended stumble a user has, higher are his chances of being retained. In order to find the *typical time* required to process an URL, we leverage the ratings data. For each URL, we compute the timespent distribution of user's who liked the URL. This distribution can be used to get a good approximation of the typical time (I_u) required to process an URL. For an URL u, the typical time I_u can be determined as follows.

5.1 Single Threshold Model

In this model, the typical time I_u for an URL u is the mean of the timespent distribution computed from the instances when the URL is thumbed-up. So for an URL u, if there are p thumb-up events by different users, then the typical time can be computed as the mean of all the timespent values.

$$I_u = \frac{\sum_{i=0}^{p} t_{iu}}{p} \quad (1)$$

Where t_{iu} is the timespent on the i^{th} thumb-up on the URL u. To have enough confidence on the calculated mean threshold, we only take URLs having at least n likes. Alternatively, as the timespent distribution is not necessarily normal, we also experiment with the median model. In the median model, the I_u value represents the median of all the thumb-up timespent values.

There are different ways the single threshold model could be used to predict the user retention. Following are the different ways we use the model to generate features.

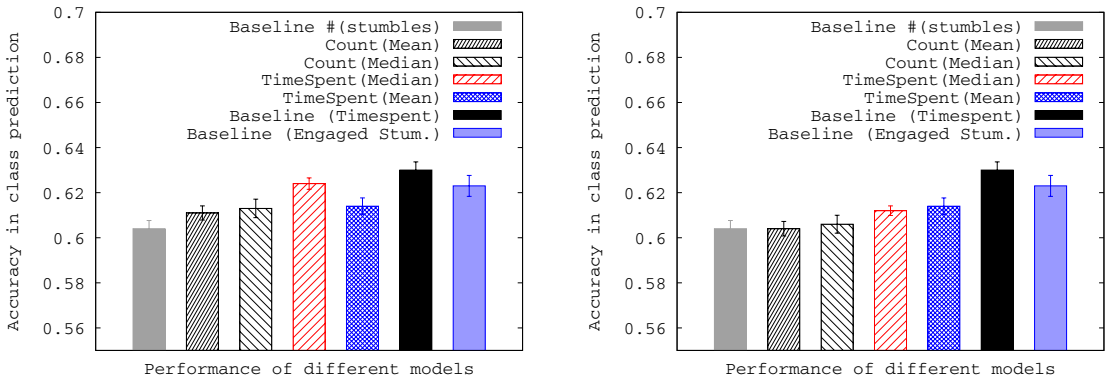
- The count of attended stumbles across k sessions. We call these as *count* features.
- The average number of attended stumbles in k sessions. It is the number of attended stumbles divided by the total number of stumbles in k sessions. We call these as the *normalized count* features.
- The cumulative timespent on these attended stumbles. This is referred to as *timespent* feature.
- The average timespent on these attended stumbles in k sessions is computed as the timespent on all attended stumbles divided by the number of stumbles in k sessions. We refer to this feature as *normalized timespent* features.

As these features are correlated, (for example, the count features and the timespent features are correlated), we study the performance of different features individually. Later in the paper, we also study the performance of the combination of these features.

5.2 Range Threshold Model

It can be argued that a single threshold model does not agree with the intuition that different users can have different speed of processing the URL. Perhaps a range of time

[1]Due to confidentiality reasons we scale the original values



(a) Single Threshold Model

(b) Range Threshold -1δ to +1δ

Figure 5: Performance of Threshold Models (Error bars 1 standard deviation)

is more suitable to find if the user spent the required time in a given time range to have processed the URL. Based on the single threshold mean and median models, we have two range models as follows.

For the first model, we use the standard deviation (δ) of the thumbed-up timespent distribution. For an URL u, the time range would be from $-(x * \delta)$ to $+(x * \delta)$. We try two ranges for $x = 1, 2$ respectively. We refer to this standard deviation range as *mean range* model. If the timespent by a user falls within this mean range for an URL u being stumbled by the user, we say that the user attended that stumble, otherwise we consider that the stumble was skipped.

As with the mean, we experiment with a range model for median as well. We use median absolute deviation to define our range. Median absolute deviation $\hat{\delta}$ is the median of the absolute deviations of the values from their median. Mathematically,

$$\hat{\delta} = median(|t_i - median_u|) \quad (2)$$

Where, $median_u$ is the median of the timespent distribution generated from all the instances when the URL u was thumbed-up. As before, we try the range from $-(x * \hat{\delta})$ to $+(x * \hat{\delta})$ for $x = 1, 2$ with the median absolute deviation $\hat{\delta}$.

As with the single threshold model, we have four kinds of features for each range of the range threshold model.

5.3 Performance of Threshold Models

Figure 5(a) and (b) present the performance of the single threshold and range threshold models. The x-axis lists the different models whereas the y-axis is the accuracy in predicting the retention class, hence higher the accuracy value, better is the model performance. As mentioned previously, we can have four variants for each threshold model. In both the figures, the model labels prefixed by '*Count*' use the count of selected stumbles (as directed by the model) as described above, while systems prefixed by '*TimeSpent*' use the cumulative timespent of selected stumbles (as directed by the model) as a feature. So for instance, if the model label reads *TimeSpent(Median)*, it means that the model uses the cumulative timespent of all the stumbles j of a user, where the user spent more time than the median time spent on the URL u. The '*Count*' models count the number of stumbles that are attended as defined by the model. As the goal of this work is to find the discriminative model/feature for user retention, we start by studying the performance of individual features. Incremental performance can then be achieved by combining multiple features into a single model. We study the combination of top performing features later in Section 7.

The normalized versions did not perform better than any of the baselines, hence we do not present the results for the normalized versions. Figure 5(a) shows the performance of the single threshold models. Amongst the single threshold models, the timespent mean and the median models outperform the number of stumbles (baseline) and just edged above the engaged stumbles baseline model. However, surprisingly the mean and median models could not outperform the baseline cumulative timespent model. Contrary to our initial hypothesis, these normalizations did not improve the prediction accuracy of retention. The range threshold model (Figure 5(b)) showed a marginal degradation in performance. Also, keeping an URL specific threshold did not help either and could only perform as good as the fixed threshold (engaged stumble model).

5.4 Analysis of Threshold Models

Even though the threshold models could not do better than all the three baselines, there is some important information that can be learned from the plots in Figure 5. Timespent models always perform better than the count models. For both the single threshold and range threshold models, the timespent mean and median models perform better than the count mean and median models. Timespent model is an interaction of a user and the content, as different users spend different amount of time on the same URL. However, the count models just counts the number of attended stumbles as the user timespent is integrated out. With baselines, a similar trend was observed. The timespent based baselines (cumulative timespent and the engaged stumbles) perform better than the baseline model for count of the number of stumbles.

6. ACCOUNTING FOR USER STUMBLING SPEED

Although the threshold models described above have a separate threshold for each URL, the threshold is fixed for all the users. Diane et. al [8] showed that the dwell time (speed of processing) for a web page varies for different users.

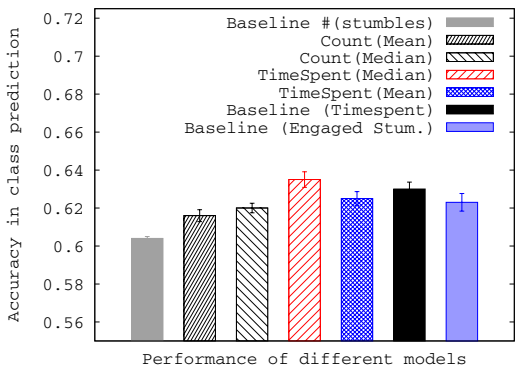


Figure 6: User modeling performance (Error bars 1 standard deviation)

Some users displayed consistently longer dwell times compared to others. In order to account for this effect, for the next model, we incorporate the variance amongst different user's stumbling speed as an additional normalization to timespent. The idea is to change the typical time spent threshold I_u for an URL based on the user stumbling speed. One of the primary motivation for analyzing the user related features is to check if they can contribute substantially towards the prediction of user retention.

For single threshold model, let I_u be the threshold for an URL u (the time required to process u). Now, if a user on average processes an URL faster than others, we decrease the threshold (move the threshold to its left) for this user by some amount, say z. Whereas if a user is slower than others, we push the threshold to its right. We compute the z-score to shift the threshold for a given user as follows. Let V be the set of all the URLs stumbled by the user s and let t_{sv} be the timespent by user on a URL v. Let the mean of the timespent distribution (generated by using only the instances when the URL was thumbed-up by all users) of an URL v be denoted by I_v. Also, let δ_v be the standard deviation of this distribution. So the average z-score $\hat{z_s}$ of user s over all URLs v can be computed as follows.

$$\hat{z_s} = \frac{\sum_{v \in V} \frac{t_{sv} - I_v}{\delta_v}}{||V||} \tag{3}$$

Now for each user s, we will have a $z-score$ for each of the URLs the user has stumbled. We find the average $z-score$ of the user for all these URLs, denoted by $\hat{z_s}$. This average z-score $\hat{z_s}$ is used to shift the threshold for an URL u based on the user s's stumbling speed. The amount by which the threshold is shifted is computed by multiplying the standard deviation (δ_u) of the timespent distribution (only thumb-up instances) of URL u being recommended and the average z-score, $\hat{z_s}$, for the user s. The sign of z determines the direction in which the threshold is shifted. Negative value indicates that the threshold is shifted to left, and positive value means it will be shifted to its right.

Mathematically,

$$z = \delta_u * \hat{z_s} \tag{4}$$

$$I_u^{new} = I_u + z \tag{5}$$

I_u^{new} is the new threshold for URL u. For the range threshold model, we move the range in a similar manner based on the user stumbling speed.

6.1 Performance of user features

Figure 6 presents the performance of incorporating the user stumbling speed into the single threshold model. This sophisticated model of changing the threshold for the stumble based upon the speed at which the user processes the stumbles promised better performance than what it could deliver. One reason for this is that the user z-score timespent values are spread out and perhaps mean is not the representative of the z-scores of the user. Though we also tried the median model on the z-scores, it did not perform any better either. As with the earlier models, timespent models performed better than the count models.

7. FINDING INFORMATIVE URLS

The metrics discussed above make an assumption that each and every stumble has an equal influence on the user's decision to be retained or not. It may be the case that some URLs leave a greater positive impact on the user while other URLs may push the user away from stumbling further. The impact of an URL on user retention can be estimated by measuring the odds of URL being stumbled when a user is retained versus not retained. Thus URLs that have a higher odds of being stumbled during a retained user's session, we refer to it as a *retaining-urls*. Whereas URLs that have a higher odds of being stumbled when a user is not retained we call it a *non-retaining-urls*. The name should *only* be seen as an indicator for retaining-ness. Thus, models discussed next will account for the number of *retaining-urls* and *non-retaining-urls* in a user's session and use this information to decide whether the user will be retained or not. We hypothesize that if the user spends more time on *retaining-urls*, they have more chance of being retained and vice-versa. On the other hand, if the user spends more time on *non-retaining-urls* then they have higher chance of not being retained. We make a simplifying assumption, that each URL has an independent effect on the user's retention or non-retention outcome.

We compute the retaining-ness *score* for each URL from all the retained and not retained user's past sessions. For each (URL) u, we compute the probability of the URL being retaining and non-retaining. The extent to which an URL is retaining is calculated based on that URLs participation in the retaining and non-retaining user sessions. Let c_{ru} be the count of URL u's participation in retained user sessions and c_{nu} be the count of URL u's participation in non-retaining user sessions. The total count of URL u with all the users is $c_u = c_{ru} + c_{nu}$. The probability of an URL being retaining is computed as follows.

$$P(RET|u) = \frac{c_{ru}}{c_u} \tag{6}$$

While, the probability of an URL being non-retaining is computed as follows.

$$P(NR|u) = \frac{c_{nu}}{c_u} \tag{7}$$

The retaining and non-retaining probabilities can be combined in a score that can indicate if the URL u is retaining

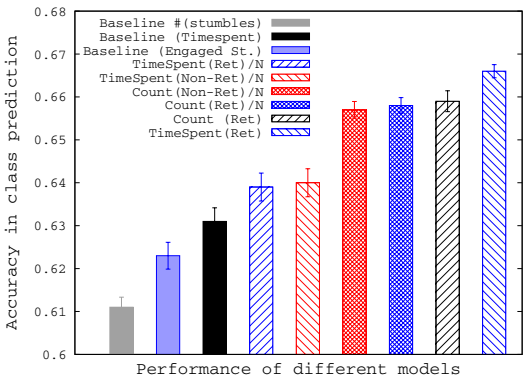


Figure 7: Performance of models based on informative URLs (Error bars - 1 standard deviation)

or not. The score for each URL is calculated according to the following equation.

$$score(u) = P(RET|u) * log\frac{P(RET|u)}{P(NR|u)} \quad (8)$$

The score value can in fact be used to rank all the URLs. We use a threshold on $score(u)$ to separate urls as retaining or non-retaining. We start with a threshold of 0 for $score(u)$. URLs with a $score(u) >= 0$ are annotated as *retaining-urls*, indicating that the URL u has a higher chance of being retaining than non-retaining. And URLs with $score(u) < 0$ are annotated as *non-retaining-urls*. We experiment with a range of thresholds in the paper.

7.1 Models on Informative URLs

Once the informative URLs are identified, the timespent, count and normalization models can be generated as before. The models using the *timespent* are prefixed by timespent and models using the count of the stumbles are prefixed by count. For instance, the model *TimeSpent(Ret)* is the timespent by the user on all the retaining stumbles, while *count(Non-ret)* indicates the count of all the non-retaining stumbles. As with the retaining stumbles, the non-retaining stumbles can give a fair indication of the user stumbling session. For example, a user who spends less time on non-retaining stumbles, has an increased chance of being retained. For both the count and timespent models, we have a normalizing model that normalizes the feature with the number of stumbles in k session. For instance, *Timespent(Non-ret)/N* represents timespent on non-retained stumbles normalized by the total number of stumbles in k sessions.

7.1.1 Bias removal

As evident from the plots in Figure 2, retained users tend to have more stumbles than non retained users. In addition, during recommendations, URLs are ordered based on the quality of the URL. Hence it is likely that retained users see URLs that non retained users never see. Thus $score(u)$ for certain URLs can be biased simply because retained user saw the URL and not because of the retaining-ness quality of the URL. This would violate the identically and independently distributed assumption of the prior stumble distribution of a given URL for the two classes; retained and not retained. In order to remove this bias, we consider equal number of stumbles for both the retained and non-retained sessions. The number of stumbles to be considered from each session is taken as the average number of stumbles in a non-retained users session. This ensures that we take enough number of non-retained sessions into our computation while removing the bias.

7.1.2 Data set generation

We estimate the retaining quality, $score(u)$ of URLs from user retention data (on training user data) and in turn use this retaining $score(u)$ of the URLs to predict user retention (on test data). In order to keep the experiment free from biases, we separate the training and testing users from each other. We randomly select 75% of user for training and test on the remaining 25% percent of users so that there is no overlap between the training and testing set users. Although the users in training and testing set are completely different but there is overlap in the set of URLs that were recommended to the two sets of users. This is still a fair comparison, since we test our accuracy on predicting user retention, not on predicting whether the URL is *retaining-url* or *non-retaining-urls*. In a later section, we also perform experiments by keeping a non-overlapping training and test set in terms of both URLs and users.

7.2 Model Performance

Figure 7 presents the results of the URL information model. We follow the same naming scheme to label the different models as before. The histograms involving solid color scheme show the performance of the three baselines. We only show the models that perform better than the three baselines. The best performing model was the timespent on the retaining URLs. As shown, the improvement by the *TimeSpent(Ret)* over the three baselines is statistically significant. This model is followed by the count of retaining stumbles model. The top three models are generated from the *retaining-url*. The *TimeSpent(Ret)* model gives a percentage accuracy gain of 5.54% and the next best which is the *count(Ret)* gave a gain of 4.44%. It should be noted that, the problem being tackled is a very hard classification problem due to the dynamic user behavior and even modest performance gains, if significant, are very important in StumbleUpon's setting. Considering this fact, performance gains of 5.54% and 4.44% can lead to significant improvements in user targeting in form of email, notifications at StumbleUpon. Also as before, the timespent model does better than the count model and the baselines barring few exceptions. The results in Figure 7 are scaled to justify this fact.

7.3 Varying the Number of Sessions

The results discussed above were based on the analysis of the first 3 sessions for each user. In order to confirm that similar results are observed irrespective of the k, we vary the number of sessions from 1 to 5 and analyze the performance of the system. We do this exercise only for the informative URL models. As shown in Figure 8, the *TimeSpent(Ret)* model performance out-performs both the baselines for all the values of k. The relative performance of all the models stayed approximately same across different values of k. We only report the performance of the top model – *TimeSpent(Ret)* and the top two baselines.

One important point that can be observed from the plot is that as we increase the value of k, the accuracy of all the

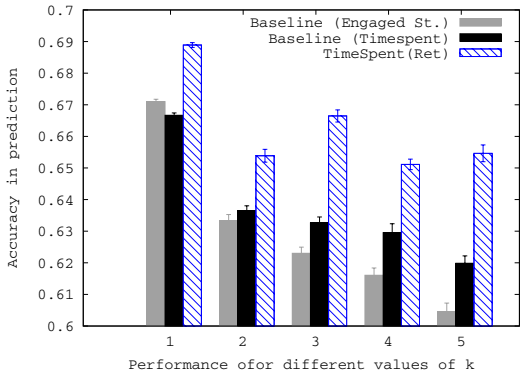


Figure 8: Performance of Timespent model for different k values

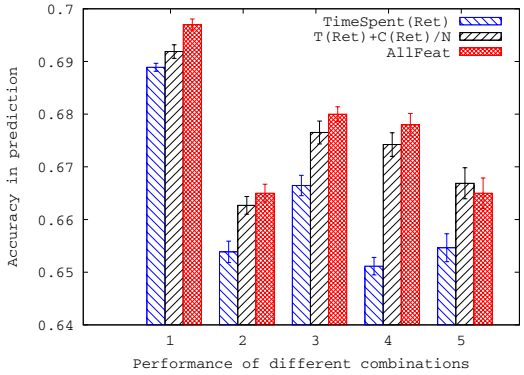


Figure 9: Performance of combination of models for different k values

models and the baselines decreases. This indicates that features measured in earlier sessions are more discriminative of user retention than later sessions. This means the boundary between the classes will usually be more crisp for the first few sessions. As we increase the number of sessions, the difference between the retained and the non-retained users blurs, making it difficult for the classifier to classify.

The accuracy gain for *TimeSpent(Ret)* over the timespent baseline was found to be 0.5%, 1.34%, 1.51%, 3.54% and 1.87% for $k = 1$ to 5 respectively. The improvements for different values of k were found to be statistically significant. It can be noted that the performance gain is higher for increasing value of k.

7.4 Combining Models

Given that the timespent and count models separately explain the retention accuracy, we wondered if the combination would provide even better prediction accuracy ? We tried combining other features with the top performing *TimeSpent(Ret)* feature. The maximum gain was achieved when *TimeSpent(Ret)* and *Count(Ret)/N* were combined. Figure 9 presents the accuracy gain for the combination of two top features. Compared to the *TimeSpent(Ret)*, this combined model improved the accuracy gain by 0.5%, 1.4%, 1.51%, 3.57%, 1.94% for session 1 to 5 respectively. We incrementally added additional features to the combination model, but this did not produce any noticeable improvements over

Rank	Feature	Chi-square	Accuracy
1	No. of times stumbled	0.218	0.710
2	Count of thumb-ups	0.209	0.703
3	Count of thumb-downs	0.203	0.698
4	Count of shares	0.180	0.676
5	Count of comments	0.170	0.662
6	Category	0.098	0.579
7	Mime-type	0.051	0.534

Table 2: Characteristics of Retained URLs

the model that consists of the top two features. This is expected since some degree of correlation exits between features. For instance, if the number of retained stumbles increases, this will result in a larger timespent on retained stumbles.

7.5 Improvisations

The classification of URL as *retaining-url* or *non-retaining-url* is dependent on few free parameters. Firstly, a threshold of 0 was applied on the $score(u)$ for determining *retaining-url* or *non-retaining-urls*. We experimented with increasing the threshold on the positive side. This threshold is a very aggressive selection of *retaining-url*. Higher the threshold, larger is the confidence on the retaining quality of the URL (and lower confidence on non-retained URLs). There was no change in the performance of the top feature as we gradually increased the threshold. After a certain point the threshold was too aggressive and performance of the model started declining.

We also experimented with the attended stumble model (section 5), timespent model, count models and the user stumbling speed model as discussed in the earlier sections, but this time applying them only on the informative URLs. Most of the these improvisations lead to small improvements, hence we do not report them.

7.6 Retained URL characteristics

What are the unique properties of the *retaining-urls*? At StumbleUpon, URL quality is assessed by a number of factors like, number of times stumbled, number of thumb-ups, number of thumb-downs, count of shares, count of comments, mime-type, topic category etc. Category and mime-type are categorical features while rest all are numeric features. The motivation for including the mime-type and topic category was to determine if there are any specific categories and / or mime types that forms a large share of the retaining URLs set which can be used for improved recommendations.

There are many ways to find the importance of a feature for a class, like chi-squared ranking, information gain, oneR, KL-divergence between the posteriors $p(f|RET)$ and $p(f|NR)$. We assess the feature effectiveness based on chi-squared test and information gain. As decision trees use information gain to find the best feature, we instead use decision trees with single features and report their accuracies on the decision tree classification for retained and non-retained URLs. Table 2 shows the ranking based on chi-square measure and the decision tree accuracy on individual features. As shown, both the rankings agree with each other. In general, number of stumbles is a strong indicator of URL quality, followed by the thumb-up and thumb-down features. As per the decision tree, mime types like plain texts, text/css, shockwave mime type, videos fell in the non-retained cate-

Rank	Feature
1	Domain average timespent
2	Domain greats ratio
3	Domain nonadult count
4	No. of Search results for the URL
5	Link word score
6	Average URL size
7	Common link score

Table 3: Characteristics of Retained URLs (for newly discovered URLs)

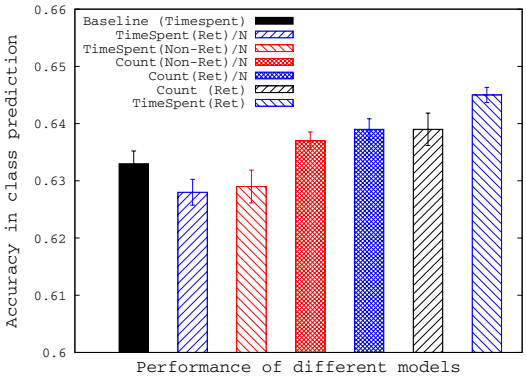


Figure 10: Performance of models based on retaining URL prediction (Error bars - 1 std. deviation)

gory. If the length of the video is long, there is a possibility that the user might not process the complete video. While applications like shockwave etc. may have issues with plugin/driver etc.

8. DEALING WITH NEW URLS

Daily, thousands of news URLs are discovered by users on StumbleUpon. For these newly discovered URLs that have received very few stumbles, computing the retention quality score, $score(u)$ will be unstable. In such cases, the problem is to first predict if the URL is retaining or not based on the webpage content of the URL.

We study this problem under the *worst* case situation where if all the stumbles recommended to the user are new URLs, and the classifier has to predict if the user will be retained or not. This experiment involves two levels of classification – The first classifier (referred as *URL classifier*) classifies if the URL is *retaining-url* or *non-retaining-url*, and the second classifier (referred as *user classifier*) determines if the user is going to be retained or not. This problem shares some similarity to Sponsored search where the click-through rate (CTR) for new ads needs to be predicted and based on the prediction, new ads are ranked with already established ads based on the CTR values [7].

For this double classification task, we had to generate the training and testing dataset so as to avoid any user/URL overlap in the two data sets. For creation of the datasets, we divide our data into two sets, one before time M and the other with stumbles after time $M+1$. For training data, we collect all the user stumbles from the sessions prior to time M. The unique URLs from the user stumbles makes up our training data for the URL classifier. Next, in order to generate the testing dataset, we find all the user sessions which do not include any URLs from the ones collected before time M. This exercise is done to ensure that no URL is present in both the training and testing set of the URL classifier. All the unique URLs from these non-overlapping user sessions (after time $M+1$) form our testing set for the URL classifier. The testing set for the user classifier is the non-overlapping sessions generated after time $M+1$. By splitting the data on the basis of time, we have a sufficiently large number of non-overlapping sessions. We treat all the URLs in the test set as if they were new URLs and only use the URL specific features. Also, unlike the previous experiments we do not balance the retaining and non-retaining URLs for the URL classifier, as the decision for all the URLs will be used by the user classifier to classify the user as retained or non-retained.

For each URL, we use three different set of features. *Domain features:* This set contains statistics about the domain of the new URL, for example, the count of stumbles, average timespent by users on the domain, count of greats, bads, shares and comments from the domain. *URL specific feature:* These features include page rank of the URL, number of search results for the URL as a query, number of capitalized words in the title, common link score (number of words in common between the anchor words), number of images in the webpage content of the URL, number of spelling errors, number of ads etc. *Discoverer features:* These features pertain to the user who discovered the URL. It includes the average number of thumb ups/downs, shares, comments on all the discoveries by the user, and binary features like whether the user who discovered the URL was a spammer etc. In total, the URL classifier uses 41 features.

Given the set of features, we used a decision tree for the URL classification. The URL classifier gave an overall accuracy of *67.35%*. Table 3 shows the top seven features used by the URL classifier. Majority of the features belong to the domain of the URL. Additionally, some of the URL specific features like link to word ratio (number of words appearing as a link divided by the total words in the URL content), number of results fetched for the URL as a query, common link score are also present in the list of top features.

Once the classifier predicts whether the URL is *retaining-url* or not, we run the URL information models on the *retaining-urls* to test the user classifier. The results for the user classifier for different models (described in Section 7) is as shown in Figure 10. For this model, the best performing baseline was again the timespent model. Hence, we only report the performance against the best performing baseline. The overall accuracy of all the models based on the informative URL prediction reduced by a small amount. However the relative performance amongst the informative URL model was maintained. As a result of this, *Timespent(Ret)* gave the best accuracy and still achieved an accuracy gain of around 1.9% over the baseline. The *Timespent(Ret)/N* and *Timespent(Non-Ret)/N* could not outperform the baseline, while the count based retained and non-retained models managed to give a small accuracy gain over the baseline.

9. RELATED WORK

Despite its numerous applications, the problem of predicting user retention based on the timespent appears to be under-explored. Some of the earlier work that addresses similar problems of user churn prediction include Dror et. al [5], Borbora et. al [2] and Burke et. al [3].

Dror et. al [5] worked on the problem of predicting if the

new users will be retained for Yahoo! answers. As in our case, Dror et. al attempt to predict the retention of new users who are into their initial phase of using the product. The authors attempt to predict churn based on a variety of features ranging from activity in the first week, personal information, frequency of activity, and interaction with the contacts etc. Their results indicate that the number of answers given by the user and the extent to which the user was recognized on those answers (e.g. thumbs up on the answers)were the best indicator for retention.

In the social network domain, Burke et. al [3] study the experience of newcomers on Facebook during their first few weeks and predict their sharing behavior based on the learnings in the initial period. They analyze the sharing behavior on different signals; whether the user was inclined to share early on, whether the user's share was recognized by the friends, whether the user was singled out during their initial phase. They found that users who are inclined to share or are motivated early on, share increasingly later. On the other hand, users who are singled out share less later on.

Work by Borbora et. al [2] study the user churn problem in a social gaming environment. They study the time period just before the user stops using the social gaming product and compared it with the activity of a regular user. Based on this discriminative analysis the authors identified several features related to signals like engagement, persistence and enthusiasm. The discriminative features were used to create a distance metric that gives the distance between two sets of users (ones who are likely to leave versus normal users).

Richter et. al [9] attempt to predict the customer churn in the telecommunication market. The authors exploit customer interactions to detect the groups of subscribers who are prone to churn. The interactions within different groups are analyzed to predict churn based on different information theoretic measures.

Additionally, work by Gunduz et al. [6] and Srivastava et. al [11] addresses the problems related to usage patterns mining. Srivastava et. al introduce different user mining pattern algorithms, use cases and applications. Gunduz et. al propose to cluster similar sessions. A tree is generated from all the clustered user sessions. For every new session, the session will be assigned to some cluster and the recommendation is made based on the tree. Deepak et. al [1] propose a fast online method for a time-sensitive recommender problem of recommending various web pages based on user modeling. They learn item specific features in addition to the user factors and use a regression based method for recommendation. User modeling has also been explored in web search for personalization. Shen et. al [10] infer user modeling features based on a client side that performs implicit user feedback. Next, the authors use the inferred implicit user model for personalized search.

10. DISCUSSION & FUTURE WORK

Content recommendation products on the web often rely on the small fraction of feedback ratings that their users provide. In such scenarios, using timespent to infer user satisfaction would be beneficial. For entertainment platforms like Youtube and Netflix that host videos of varying length and types (TV shows, movies etc), inferring whether a user processed/liked the video simply based on timespent would be advantageous. As long as a sparse set of user ratings and timespent on videos can be collected, one can use the methods for content type normalization and user normalization discussed in this work to infer user satisfaction and churn rate.

The informative URL model demonstrates that the timespent on a subset of URLs carries more information about user retention as opposed to the timespent on all the stumbles. Also, as evident from Figure 5(a) and (b) and 7 the timespent based models (user + URL) perform better than the count models (URL). This result demonstrates that the content+user interaction models perform better than just the content models.

The informative URLs model assumes that each stumble affects user retention independent of the other stumbles. In practice it is possible that a pair of specific URLs (or more) appearing together in a session may influence the retention outcome. Presently, due to data sparsity issue, estimating the joint influence, *P(RET | url1, url2)* of two or more URLs on user retention is not possible. In reality if the user retention outcome is an effect of a sequence of two or more stumbles, modeling the dependence of the URLs can help improve the performance. Dealing with the dependency and the resulting sparsity is a part of our future work.

In addition to using the timespent, we could factor in other types of indicators to improve the prediction. For instance, we could account for the topic the user is stumbling and further analyze if different topics exhibit different characteristics. We could do the same analysis by segmenting the users in different age-groups, factoring users biases towards some topics. Further, it would be nice to work out a way to combine the ratings with the informative URL models to improve the prediction of user retention. Also it would be nice to figure out a way to impute the sparse rating data and make it more usable.

11. CONCLUSION

The work described in this paper can be summarized as follows: 1. Timespent on Informative URLs is more indicative about user retention than total timespent on all URLs or just the URL features alone. 2. Modeling user's speed of stumbling alone contributes little to the overall performance gain and when combined with timespent modeling for the URL, gives only a marginal improvement in prediction accuracy. 3. The best performing informative URL model (*TimeSpent(Ret)*) uses the timespent by the user on the retaining URLs. This model is an interaction of URL and user properties. So overall, features that involve interaction of user and URL (timespent based models) perform better than the models involving only the URL (count based models). Further, normalizations of the timespent (user centric) and the URL models do not help much in improving prediction of user retention. 4. The informative URL model is robust even for predicting the retaining-ness quality of newly ingested URLs, which in turn, helps in predicting user retention.

12. ACKNOWLEDGEMENTS

We would like to thank Ashish Agrawal, Rohan Monga and Ulas Bardak for providing us the domain and URL features for building our classifiers. In addition, a special thanks to Geoff Smith and Christina Marcet for their valuable comments on the manuscript.

13. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 703–712, New York, NY, USA, 2010. ACM.

[2] Z. H. Borbora and J. Srivastava. User behavior modelling approach for churn prediction in online games. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, SOCIALCOM-PASSAT '12, pages 51–60, Washington, DC, USA, 2012. IEEE.

[3] M. Burke, C. Marlow, and T. Lento. Feed me: motivating newcomer contribution in social network sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 945–954, New York, NY, USA, 2009. ACM.

[4] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces*, IUI '01, pages 33–40, New York, NY, USA, 2001. ACM.

[5] G. Dror, D. Pelleg, O. Rokhlenko, and I. Szpektor. Churn prediction in new users of yahoo! answers. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 829–834, New York, NY, USA, 2012. ACM.

[6] c. Gündüz and M. T. Özsu. A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 535–540, New York, NY, USA, 2003. ACM.

[7] D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, and C. Leggetter. Improving ad relevance in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 361–370, New York, NY, USA, 2010. ACM.

[8] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 377–384, New York, NY, USA, 2004. ACM.

[9] Y. Richter, E. Yom-Tov, and N. Slonim. Predicting customer churn in mobile networks through analysis of social groups. In *SDM*, pages 732–741, 2010.

[10] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 824–831, New York, NY, USA, 2005. ACM.

[11] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, Jan. 2000.