Mining User Development Signals for Online Community Churner Detection

MATTHEW ROWE, School of Computing and Communications, Lancaster University, UK

Churners are users who stop using a given service after previously signing up. In the domain of telecommunications and video games, churners represent a loss of revenue as a user leaving indicates that they will no longer pay for the service. In the context of online community platforms (e.g. community message boards, social networking sites, question-answering systems, etc.) the churning of a user can represent different kinds of loss: of social capital, of expertise, or of a vibrant individual who is a mediator for interaction and communication. Detecting which users are likely to churn from online communities therefore enables community managers to offer incentives to entice those users back; as retention is less expensive than re-signing users up. In this paper we tackle the task of detecting churners on four online community platforms by mining user development signals. These signals explain how users have evolved along different dimensions (i.e. social and lexical) relative to their prior behaviour and the community in which they have interacted. We present a linear model, based upon elastic-net regularisation, that uses extracted features from the signals to detect churners. Our evaluation of this model against several state of the art baselines, including our own prior work, empirically demonstrates the superior performance that this approach achieves for several experimental settings. This paper presents a novel approach to churn prediction that takes a different route from existing approaches that are based on measuring static social network properties of users (e.g. centrality, in-degree, etc.).

Categories and Subject Descriptors: H.1.8 [**Information Systems Applications**]: Data Mining; H.1.2 [**Information Systems Applications**]: Collaborative and social computing systems and tools—*Social networking sites*; J.3.4 [**Human-Centred Computing**]: Collaborative and social computing—*Empirical studies in collaborative and social computing*

General Terms: Experimentation

Additional Key Words and Phrases: Churn prediction, lifecycle mining, online communities, social dynamics, lexical terms

ACM Reference Format:
Matthew Rowe. 2015. Mining User Development Signals for Online Community Churner Detection. *ACM Trans. Knowl. Discov. Data.* V, N, Article 1 (January 2015), 29 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The problem of users *churning* (i.e. leaving a service) can have a significant impact upon service adoption and thus return on investment. Therefore, churn detection, the process of identifying which users will drop out of using a given service, is an important challenge in a number of domains including: (i) telecommunications, where operators wish to know who is likely to leave the service so that incentives may be given to those individuals to remain as customers; (ii) online games, where users may cancel their subscription to the service and thus result in a financial loss to the service provider;

Author's address: M. Rowe, School of Computing and Communications, Lancaster University, UK. email: m.rowe@lancaster.ac.uk

DOI: http://dx.doi.org/10.1145/0000000.0000000

(iii) social networks, where brand managers and social media marketers wish to maintain large subscriber networks to increase the number of individuals who see their published information, and finally; (iv) online community management, where the churning of certain users, in particular in communities which revolve around knowledge sharing, can have a significant impact upon the utility of the community and its ability to provide solutions to support requests.

To date, the majority of work within the area of churn detection, and from the disparate domains cited above, has focussed on building a detection model using information about a user's social network position, and thus the extent to which he is interacting with other users, and/or the activity of a given user up until a given point in time. As a result, few pieces of work have considered how a user has developed throughout his *lifecycle* (i.e. the period of time between a user joining a service to the present) when using the service, and how this information can be used to detect which users will churn. This leads to two questions: *Are there salient differences in how churners and non-churners develop?* And; *how can such development information be incorporated into an approach to detect churners?*

In this paper we present the results from examining both of these questions with an analysis of the development signals of churners and non-churners mined from four online community platforms, and a detection model that leverages features of these signals to identify who will churn and who will remain. In our prior work [Rowe 2014] we presented a detection model based upon Gaussian Sequences, in this paper we advance over this work by explaining how a linear model with elastic net regularisation offers significantly better detection accuracy (e.g. yielding ROC value of 0.706 in one instance) in certain cases, and examining the impact of smoothing on detection accuracy of the Gaussian Sequence model [Rowe 2014]. We build on our other prior work [Rowe 2013] which presented an approach to model the lifecycles of users in online communities by examining different lifecycle fidelities, rather than the 20 stages inspected previously, and without the need to induce development trajectory functions. This paper also presents a more thorough, fine-grained, and deeper analysis of how churners and non-churners differ in their development; thereby expanding over our prior work [Rowe 2014; 2013], and leading to the proposition of a *theory* for churner development. We hope that the framework and approach outlined in this paper, and the theory of churner development, will be tested in future work within the community on differing datasets and platforms. We make the following contributions in this paper:

— An approach to model the development signals of web users across different lifecycle fidelities (i.e. 5, 10, 20 stages) rather than one fidelity setting [Rowe 2013].
— Inspection of churners' and non-churners' development signals across four online community platforms, and the proposal of a theory for churners' development from development observations.
— A method to mine static and rate features from users' development signals.
— A detection model based upon linear models with elastic net regularisation trained using a stochastic coordinate descent learning approach - rather than average coordinate descent.
— Evaluation of the detection model across the four platforms' datasets showing that we significantly outperform existing models including baselines from current work [Karnstedt et al. 2011; Rowe 2014].

We have structured the paper as follows: Section 2 presents related work from the related fields of social network evolution, user lifecycles, and churn prediction. Section 3 details the datasets that we used for our experiments and analyses within this paper. Section 4 explains how we model users' lifecycles, elaborating on model design

decisions that we made previously [Rowe 2013]. Section 5 describes how we decide who is a churner and who is not a churner in the context of online community platforms, and examines how churners and non-churners develop differently across the examined platforms. Section 6 defines the process of engineering features from users' development signals, and section 7 presents the various detection models that harness those features and the evaluation of those models. Section 8 discusses the implications of this work and plans for future work, within both churn prediction and social computing, and section 9 finishes the paper with the conclusions that we drew from this work.

2. RELATED WORK

2.1. Social Network Evolution

Social network evolution concentrates on examining social network development across disparate systems. For instance, work by Kossinets and Watts [Kossinets and Watts 2006] examined university social networks derived from email headers passed between students and university staff, finding that network properties reached a global equilibrium in terms of the proportion of mutual acquaintances between individuals in the network and shared study classes. Mislove et al. [Mislove et al. 2007] compared the social network properties (link symmetry, power law distributions of edges and nodes, and local clustering of users) of Flickr, LiveJournal, Orkut and YouTube, and found high degrees of local clustering on the different platforms which contained densely populated subgroups of similar users. Similar cross-platform inspections were performed by Leskovec et al. [Leskovec et al. 2008] using Flickr, Delicious, Answers and LinkedIn, by deriving network processes such as node arrival and edge creation rates. The authors showed similar effects in terms of the social processes at work throughout the social platforms. Zheleva et al. [Zheleva et al. 2009] examined the evolution of users' social networks and affiliation networks on Flickr and used a generative model to replicate the processes of users creating edges and joining affiliate networks.

Panzarasa et al. [Panzarasa et al. 2009] used an online community platform provided for students of the University of California, Irvine, to assess how the social network structure of the community platform evolved over time. The authors found that certain users acted consistently as hubs through which communication was mediated. Similar work by Backstrom et al. [Backstrom et al. 2006] explored the effects that govern group formation and joining behaviour on LiveJournal, and found that using the proportion of a user's friends already within a group had a key effect on identifying group joiners. Kairam et al. [Kairam et al. 2012] assessed the dynamics of group formations within the social networking platform Ning. Like Backstrom et al., the authors found that the probability of a user joining a group was linked to the number of prior members within whom he has a relationship. Recent work by Gong et al. [Gong et al. 2012] inspected the evolution of social networks on Google+ as the platform was growing in memberships, in particular they focused on *social-attribute networks* (i.e. bipartite graphs containing people and their attributes as nodes), finding that the platform exhibited unique growth and characteristics of the networks as more people joined Google+ (i.e. reduced reciprocity). Chung et al. [Chung et al. 2012] examined the assortativity (i.e. the degree to which similar degree nodes interact with one another) of social networks derived from an online community building web site over a ten-year period, and found assortativity to increase with time.

2.2. User Lifecycles

The above works from the field of social network evolution allow one to understand the latent processes governing social network formation and development. Although useful, they do not afford insights into how specific users develop over their lifetimes in social systems, or how user development differs between users. Several pieces of recent work have explored this gap by characterising users' development throughout their lifecycles within both social platforms and adopted services. For instance, Miritello et al. [Miritello et al. 2013] examined the social evolution of users over time in terms of: a) communication capacity of users (the limit of the number of social connections they can maintain), and b) the activity rate of users (the number of edges users created), using call records data. The authors found that as people aged throughout their lifecycle that their social circle reduced in size and that interaction occurs less towards later life periods. Similarly Danescu-Niculescu-Mizil et al. [Danescu-Niculescu-Mizil et al. 2013] assessed the lexical dynamics of online community members by modelling their lexical term distributions and how these changed relative to the community, finding that users began their lifecycle within the community by adapting their language to the community but then stopped doing so. McAuley and Leskovec [McAuley and Leskovec 2013] examined how users evolved in their expertise (assuming a monotonic progression) over time in the same beer rating communities as [Danescu-Niculescu-Mizil et al. 2013]. The authors defined users as evolving based on their own '*personal clock*' where the rate of progression is user-dependent, and used this notion to mine latent experience levels for each user and then use these for recommendations.

In this paper we build on the prior work of McAuley et al. [McAuley and Leskovec 2013] by using the same notion of a *personal clock* from which to derive *lifecycle periods* of users. We then combine the social dynamics, examined in prior work of Miritello et al. [Miritello et al. 2013], and the lexical dynamics, inspected in the prior work of Danescu et al. [Danescu-Niculescu-Mizil et al. 2013], to assess how users evolve throughout their lifecycle periods. Not only do we compare the user's development with the community in which they are interacting, as in [Danescu-Niculescu-Mizil et al. 2013], but we also assess how the user has developed relative to their prior states and behaviour. This analysis of user evolution is used to discern how churners and non-churners differ in their *development signals*, and how this information can inform our churn prediction model with features to use. To gather sufficient observations from which to base a theory of churner development upon, we examine these development signals across four different online community platforms; finding commonalities in the patterns of user evolution.

2.3. Churn Prediction

The prediction of *churners* has been studied across a variety of domains. Dasgupta et al. [Dasgupta et al. 2008] analysed Call Detail Records from a large (unnamed) telecommunications provider, using initial insights from inspecting the data to show that churn likelihood, for a given user, was dependent on the number of their friends who had previously churned. A spreading activation model was then implemented to predict churn, finding that network effects are better predictors than other features such as call usage and activity. For instance, Zhang et al. [Zhang et al. 2010] predicted churners in a Chinese telecommunications network by inducing a decision tree classifier from user activity features (e.g. call duration) and network properties (e.g. 2nd order ego-network clustering coefficient), finding that a combination of activity and network features performed best for predicting churners. McGowan et al. [McGowan et al. 2011] also predicted churners from a telecommunications provider by experimenting with different dimensionality reduction and boosting methods, finding that

reducing model complexity, via information gain ratio and χ^2-testing, improved predictive performance. Meanwhile earlier work from Stutzbach and Rejaie [Stutzbach and Rejaie 2006] assessed the churning (i.e. dropping-off) of users from peer-to-peer sharing networks (e.g. Gnutella, and BitTorrent), finding that the peers who join to share files and churn demonstrate a heavy-tailed distribution of *how long* they remain to share information - i.e. the majority of churners remain only for a short time.

Analogous to churn prediction in telecommunications is the prediction of churners from online games such as Massive Multiplayer Online Role Playing Games (MMORPGs). Borbora et al. [Borbora et al. 2011] presented an approach to predict churners from Sony's game Everquest II. The authors examined theories on user participation as to what motivates users to play the game (e.g. progression and power, story lines, collaboration with groups, etc.), before then engineering features that capture, in a tangible form, the realisation of such motivations (e.g. rate of quest participation, experience points gained, number of characters interacted with). By inducing a decision tree classifier, using different feature sets (achievement-oriented, socialisation, data-driven), Borbora et al. found that socialisation features provided the best predictive performance.

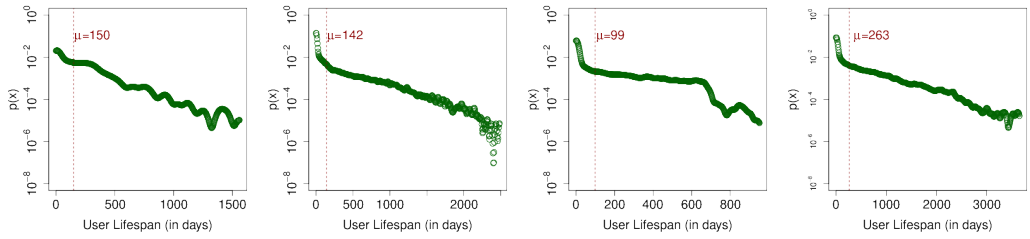
Several works have examined the prediction of churners from *social networks*: Lewis et al. [Lewis et al. 2012] examined Facebook networks of college students over a 4 year period and found an association between friendship maintenance and geographical proximity and shared tastes. Quercia et al. [Quercia et al. 2012] analysed Facebook friendships and users' personality traits, finding that churn was likely to happen if the ages of connected users differed and if one of the users was neurotic or introverted. Kwak et al. have examined churners from Twitter networks in [Kwak et al. 2011] and [Kwak et al. 2012]: in the former the authors analysed the differences between social network snapshots, separated by 6 weeks, of Korean Twitter users finding that users unfollowed other users when the users talked about uninteresting topics; while in the latter work [Kwak et al. 2012], the authors induced a logistic regression model to predict churners based on pairwise features (formed between the user and each of his subscribers), finding the more common tags between users, then the less likely they are to churn. Dror et al. [Dror et al. 2012] predicted the churning of 'new users' on Yahoo! answers by examining user features mined over a 7-day period. Features included information related to questions posed (e.g. number of questions asked), answers given (e.g. number of answers given), and gratification-received (e.g. number of best answers, number of up-votes); unsurprisingly key features for discriminating between churners and non-churners were found to be the number of answers given, the fraction of time between posting answers, and the number of answers the user submits before he/she wins the 'best answer' for a given question.

The detection of churners from the Irish online community message board Boards.ie was examined in the work of Karnstedt et al. [Karnstedt et al. 2010; Karnstedt et al. 2011] The authors found in [Karnstedt et al. 2010] that the probability of a user churning was related to the number of prior users with whom the individual has communicated having churned before. Based on such initial insights, the authors' follow-up work [Karnstedt et al. 2011] then sought to engineer a model to identify which users would churn. For this, the authors examined the social network properties of churners against non-churners (i.e. in-degree, out-degree, clustering coefficient, closeness centrality, etc.), and then used this information to learn a decision tree classifier. In order to provide a comparison between our proposed detection models, and that of prior work, we use the approach from [Karnstedt et al. 2011] as our baseline, along with a model from our own prior work based on a dual-gaussian sequence model [Rowe 2014]. We empirically demonstrate that our proposed linear detection model significantly outper-

forms these existing approaches, and therefore indicates the utility in using features mined from users' development signals to identify churners.

3. ONLINE COMMUNITY DATASETS

In order to examine the development of users across a multitude of online community platforms, and to see how our proposed churn detection models fare in disparate settings, we collected and employed four community platform datasets for our analyses and experiments: Facebook, the SAP Community Network (SAP), Server Fault, and Boards.ie. Figure 1 presents distribution plots of the user life periods (in days) and the posts per-user across the three platforms, and Table I provides summary statistics of the datasets.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 1. Distribution plots of user life periods and posts-per-user across the four platforms. The dashed red lines indicate the mean of each distribution.

3.1. Facebook

We obtained data from Facebook groups related to Open University degree course discussions. The groups allow users to discuss the problems and issues that they may be having with degree course material and potential avenues for solving those problems. Although Facebook provides the ability to collect social network data for users, we did not collect such data in this instance and instead used the reply-to graph within the groups to build social networks for individual users. In doing so we would constrain the social dynamics at play to those within the context of the groups.

3.2. SAP Community Network (SAP)

SAP provides a range of software solutions for enterprises, therefore it also offers a community question-answering platform in which users can post questions related to the issues they may be having and receive advice and solutions from the other community members; this platform is known as the SAP Community Network. Should users provide answers to questions then they receive points as rewards, thereby building up a reputation over time as being an expert in some SAP technology or software domain. We used the reply-to graph to form implicit connections between users, given that users cannot 'friend' other users on the platform - and we therefore have consistent social dynamics across the analysed platforms based upon communication patterns.

3.3. Server Fault

Server Fault is a platform that is part of the Stack Overflow question answering site collection.[1] The platform functions in a similar vein to SAP by providing users with the

[1] http://stackoverflow.com/

Table I. Statistics of the community platform datasets.

Platform	#Users	#Posts	Time Span
Facebook	4,745	118,432	[18-08-2007,24-01-2013]
SAP	32,926	427,221	[15-12-2003,20-07-2011]
Server Fault	33,285	234,790	[01-08-2008,31-03-2011]
Boards.ie	65,528	6,120,008	[01-01-2005,13-02-2008]

means to post questions pertaining to a variety of server-related issues, and allowing other community members to reply with potential answers. Similar to SAP, Server Fault also lacks explicit edge-creation features, therefore we use the reply-to graph (i.e. where a user has replied to another user's question) to form an implicit edge between the users.

3.4. Boards.ie

For the fourth dataset we were provided with data from the online community message board Boards.ie, the most popular message board in Ireland, where we had access to all posts from 2005 to 2008. On Boards.ie users post to dedicated forums where each forum contains discussions related to a specific topic (e.g. football, rugby union, chess, etc.). Users interact with one another through discussion *threads* where one user will post a message to initiate a thread and to which users subsequently reply. Unlike on SAP and Server Fault, on Boards.ie there is no notion of *expertise*, instead users build up a *reputation* through the total number of posts that they made. Given the nature of the platform, and in a similar vein to the above three platforms, we used the reply-to graph (formed from users replying to one another in threaded discussions) as implicit edges between users.

4. REPRESENTING USER LIFECYCLES

User lifecycles describe the lifetime period of a user in a given platform, or service. In the context of our work, a user's lifecycle is bound by the first and last time that they posted within the datasets that we have available. In this section we explain how such lifecycles are split up into periods, or stages, and how different user dimensions can then be inspected by analysing discrete lifecycle periods.

4.1. Deriving Lifecycle Periods

Existing recent work [Miritello et al. 2013; Danescu-Niculescu-Mizil et al. 2013; McAuley and Leskovec 2013] has demonstrated the extent to which users develop at their own pace and thus evolve according to their own '*personal clock*' [McAuley and Leskovec 2013]. We validated this finding in the context of our datasets by deriving each user's lifetime in the system into 20-equally time sliced windows and examining the proportion of the user's activity (posts) within each windowed interval. Let $birth(P)$ denote a utility function that returns the timestamp of the earliest post within a given set of posts P and $death(P)$ also be a utility function that returns the the timestamp of the final post of a user given his set of the posts. We define $interval(P)$ to return the interval width of a user's lifetime (between birth and death) when divided into 20 segments. We then derived the set of equally time-sliced life periods for a given user as a set of tuples representing closed time intervals ($[t_i, t_j], t_i < t_j$):

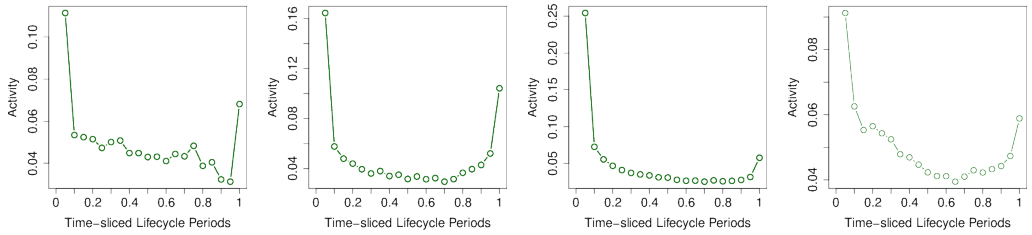
$$T_{time} = \{[t_i, t_j] : k \in \mathbb{Z}^+_{20}, t_i = birth(P_{u_i}) + k \times interval(P_{u_i}), \\ t_j = birth(P_{u_i}) + (k+1) \times interval(P_{u_i})\} \quad (1)$$

We were therefore provided with 20 time periods over which we could inspect the development of the user. We began by analysing the posting activity of each user within

our respective datasets within the different lifetime periods, therefore given a user $u_i \in U$ and a time interval $[t_i, t_j] \in T_{time}$ we derive the proportion of activity that took place within the given period as follows:

$$activity(u_i, [t_i, t_j]) = \frac{|\{p : p \in P_{u_i}, t_i \leq time(p) < t_j)\}|}{|P_{u_i}|} \quad (2)$$

where $time(p)$ denotes a function that returns the time and date at which the post was made, thus defining the set in the numerator as being those posts which fall within the given interval. Figure 2 presents the activity distribution across the time-sliced lifetime period averaged over all users in our datasets. We found that for each platform user activity peaks at the start of their lifetimes, before reducing and then increasing again towards the end, suggesting that users join the community and participate initially, before reducing their activity over time gradually. We also examined the distribution of initiations (thread starters) and replies (responses to previously initiated threads) across the platforms and found the same u-shaped curve as with the activity plots.

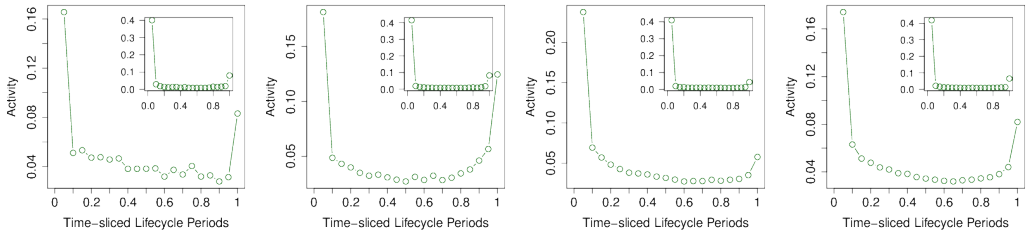


(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 2. Proportion of activity per lifecycle period (i.e. 0 = first period, 1 = final) across the online community platforms' users.

To further examine whether this behaviour is common across users who have only contributed a few posts to each platform, and those who had contributed a lot we plotted the same activity distribution for users divided into two groups: (i) users who had posted 10 or more times, and; (ii) users who had posted less than 10 times. Figure 3 shows these plots for each platform. We find here that regardless of the *type* of user (i.e. light or heavy poster) these activity distributions have the same *U*-shape: a user posts a lot initially, before tailing off, and then posting again.

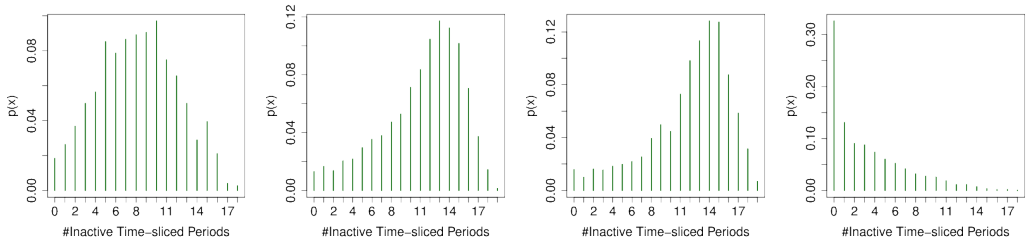
In the above plots it is clear that users follow a *U*-shape activity curve across their lifecycles irrespective of their total post frequency, however it is not clear to what extent users' lifecycle periods contain no activity at all (i.e. the user has effectively *temporarily* churned). Therefore, we assessed each user's activity throughout their life periods and recorded, for each user, the frequency of inactive periods (from the 20 periods under assessment). Figure 4 presents the distribution of the no activity stages per user across each of the platforms. We find that Facebook has a fairly symmetrical distribution about its mean of 10 inactive periods, while SAP and Server Fault have left-skewed distributions. One can imagine that the use of question-answering community platforms such as SAP and Server Fault include users who only participate when they have a given information need and thus need to find the solution to a problem, the remainder of the time they would be either *lurking* and not directly contributing or not visiting the site at all. Boards.ie meanwhile has a right-skew, as



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 3. Proportion of activity per lifecycle period with users who have posted 10 or more times (main plot) and users who have posted less than 10 times (inset plot)

over 30% of users have posted in every lifecycle period; however this is symptomatic of a community-message board where the needs of the community users to interact (i.e. for conversation and social sharing) differ from the information and issue-resolution needs of more question-answering oriented communities.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 4. Distribution of 0% activity stages throughout individual users' time-sliced lifeperiods.

These findings suggest that a time-sliced approach to deriving the lifetime periods of a user would be inappropriate as the lack of activity within certain periods would have a strong effect on the social and lexical dynamics that could be observed. For instance, a reduction in activity has a strong effect on the user socialising with other users, as he is not contributing any content he is not participating socially. Therefore for deriving the lifecycle periods of users within the platforms we adopted an activity-slicing approach that divides a user's lifetime into k discrete time intervals but with an equal proportion of activity within each period; this approach is analogous to those adopted in prior work [Miritello et al. 2013; Danescu-Niculescu-Mizil et al. 2013; McAuley and Leskovec 2013], and is an extension of our prior work in which we fixed $k = 20$ [Rowe 2013]. We defined this approach in Algorithm 1 which functions as follows: we derive the set of interval tuples ($\{[t_i, t_j]\} \in T$) by first deriving the chunk size (i.e. the number of posts in a single period) for each user, we then sort the posts in ascending date order, before deriving the start and end points of each interval in an incremental manner. This derives the set of time intervals T that are specific to a given user, these are then used to assess the evolution of users across disparate properties. In order to aid comprehension and set notation, we hereafter use an equivalence mapping between T and S where the latter is a set of integers ($S = \mathbb{Z}_k^+$) corresponding to each of the k lifecycle stages, thus we choose $s \in S$ for a given lifecycle stage that actually maps to a discrete time-interval $[t, t']$. We now define the modelling of users' social and lexical

ALGORITHM 1: Deriving the set of lifecycle periods (T) given fidelity k

Input: Collection of a user's posts P_u and lifecycle fidelity k.
Output: Set of lifycycle periods modelled as discrete closed time intervals T.

$chunkSize \leftarrow |P_{u_i}|/k$
$Q_{u_i} \leftarrow sort(P_{u_i})$
$i \leftarrow 0$
$T \leftarrow \emptyset$
while $i < k$ **do**
  $start \leftarrow i \times chunkSize$
  $end \leftarrow (i+1) \times chunkSize$
  if $end > |Q_{u_i}| - 1$ **then**
    $end = |Q_{u_i}| - 1$
  end
  $t_i \leftarrow time(Q_{u_i}[start])$
  $t_j \leftarrow time(Q_{u_i}[end])$
  $T \leftarrow T \cup \{[t_i, t_j]\}$
end
return T

dimensions in the form of discrete probability distributions - these are later used to mine development signals.

4.2. Social Dimensions

We begin by modelling the social dimensions of users using their in-degree and out-degree distributions. The former describes the number of edges that connect to a given user, while the latter describes the number of edges from the user. As we are dealing with conversation-based platforms for our experiments we can use the *reply-to* graph to construct these edges, where we define an edge connecting to a given user u_i if another user u_j has replied to him. Likewise, we also define an edge from a given user u_i to another user u_j if the former has replied to the latter.

Given our use of lifecycle periods we use the discrete time intervals that constitute $s \in S$ to derive the set of users who replied to u_i, defining this set as Γ_s^{IN}.[2]

$$\Gamma_s^{IN} = \{author(q) : p \in P_{u_i}, q \in P, time(q) \in s, q \rightarrow p\} \tag{3}$$

We also define the set of users that u_i has replied to within a given time interval as Γ_s^{OUT}:

$$\Gamma_s^{OUT} = \{author(q) : p \in P_{u_i}, q \in P, time(q) \in s, p \rightarrow q\} \tag{4}$$

From these definitions we can then form a discrete probability distribution that captures the distribution of repliers to user u_i, using Γ_s^{IN}, and user u_i responding to community users, and hence using Γ_s^{OUT}. For an arbitrary user ($u_j \in \Gamma_s^{IN}$) who has contacted user u_i within period s we define this probability of interaction as follows:

$$Pr(u_j \mid \Gamma_s^{IN}) = \frac{|\{q : p \in P_{u_i}, q \in P_{u_j}, time(q) \in s, q \rightarrow p\}|}{\sum_{u_k \in \Gamma_s^{IN}} |\{q : p \in P_{u_i}, q \in P_{u_k}, time(q) \in s, q \rightarrow p\}|} \tag{5}$$

And for an arbitrary user ($u_j \in \Gamma_{u_i t}^{out}$) who user u_i has contacted within period s we define the probability of interaction as follows:

$$Pr(u_j \mid \Gamma_s^{OUT}) = \frac{|\{p : p \in P_{u_i}, q \in P_{u_j}, time(p) \in s, p \rightarrow q\}|}{\sum_{u_k \in \Gamma_s^{OUT}} |\{p : p \in P_{u_i}, q \in P_{u_k}, time(p) \in s, p \rightarrow q\}|} \tag{6}$$

[2]We use $p \rightarrow q$ to denote message q replying to message p.

Given this formulation we now have time-dependent discrete probability distributions for a given user's in-degree and out-degree distribution, thereby allowing the *social* changes of users to be analysed in terms of the users communicating with a given user over time.

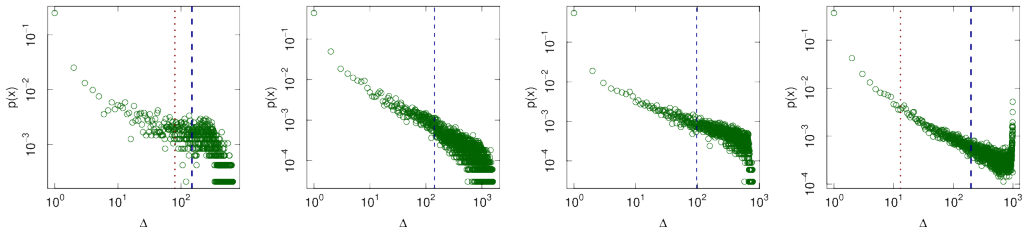
4.3. Lexical Dimensions

We model the *lexical dimensions* of users based on their term usage over time. To derive the set of terms we first retrieve all posts made by a given user within a lifecycle period and then remove stop words and filter out any punctuation. Having derived the set of cleaned posts, we then define the discrete probability distribution for a user u_i within period s based on the conditional probability of term x being used within the time interval. We define a multiset as containing the set of terms used by u_i in a given time period: $x \in C_s$ and a mapping function $\mu : C_s \rightarrow \mathbb{N}$ that returns the multiplicity of a given term's usage by the user at a given time period. Thus we define the conditional probability for term x being used by u_i during s as:

$$Pr(x \mid s) = \frac{\mu(x)}{\sum_{x' \in C_s} \mu(x')} \tag{7}$$

5. MINING DEVELOPMENT SIGNALS: CHURNERS VS. NON-CHURNERS

On subscription-based services, such telecommunications networks and gaming platforms, a *churner* is identifiable by the cancellation of the service (e.g. cancelling a mobile phone contract), however on online community platforms we do not have such information. Instead, we must examine what is *normal* in terms of users' activity and then decide on a suitable *inactivity* threshold where, should a user remain inactive for more than that period (i.e. x days) then we can say he has *churned*. To decide on this threshold, we first examined the gap distribution of the users: let Δ define the maximum number of days between posts across the platforms' datasets for each user, we are then interested in seeing how Δ is distributed for all users of the platforms.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 5. Gap distributions across users of the different platforms. The mean of each distribution, and its median, are shown using blue dashed and red dotted lines respectively.

Figure 5 shows the relative frequency distribution as a function of Δ across the platforms. We note that each platform is heavy-tailed: indicating that many users have a short *maximum* span between their posts, while only a few users actually have large windows of inactivity. We also measured two quantities of each distribution: its mean and median - these have been plotted on each distribution in blue dashed and red dotted lines respectively. As each distribution is characteristic of a left-skew we find that the median is smaller than the mean in each instance. We selected each distribution's

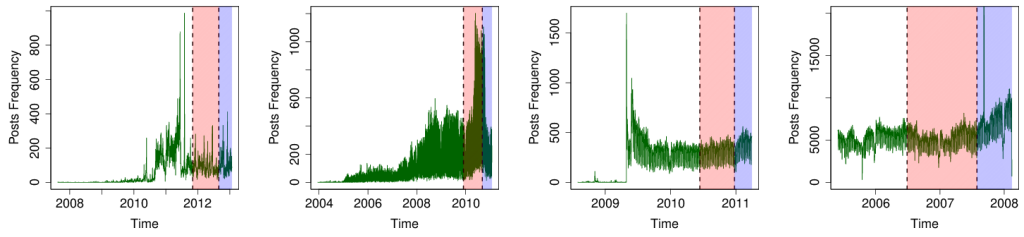
Table II. Splits of users within the datasets and the churn window duration

Platform	#Churners	#Non-churners	Churn Window
Facebook	1,033	1,199	[04-11-2011, 28-08-2012]
SAP	10,421	7,255	[29-11-2009,07-09-2010]
Server Fault	12,314	11,144	[13-06-2010,24-12-2010]
Boards.ie	18,778	30,335	[29-06-2006,30-07-2007]

mean as the *churn control window*, the usage of this window is then used a *safety* barrier when we decide who has churned and who has not: it is set to 149 days, 141 days, 97 days and 198 days for Facebook, SAP, ServerFault and Boards.ie respectively.

Having derived the *churn control window* for each of our respective platforms we then derived the churners as follows: we began by taking the final post date in a given dataset and went back n days, where n denotes the size of the churn control window, this date gives the *churn window end point* (t''). We then went back a further $2n$ days back to give the *churn window start point* (t'), thus the *churn window* is defined as a closed date interval $[t', t'']$. We then defined anyone who posted for the final time in $[t', t'']$ as a *churner* and anyone who posted after the end of the churn window as a *non-churner*, as a result we are only inspecting the behaviour of users who are active after the start of the churn window, ignoring anyone who churned before this point. The reason for this is that we not only want to predict who has churned, but also when they post for the last time - we will be exploring this in our future work - thereby mimicking a real-world community management scenario where we are provided with information up to the start of the churn window and nothing else.

Having defined the churn windows for each platform, we then derived the churners and non-churners in each of our datasets; the resultant statistics are shown in Table II. As our aim is to both detect churners and then predict the churn point of those users, we split each platform's users up into a *training* and *test* set using an 80:20% split respectively. We then used the former set to inspect how users develop and evolve along the above-mentioned dimensions, and engineer features to capture such evolution; and the latter set (test) to detect churners from non-churners. We now move on to examining how users evolve along social and lexical dimensions, and how we can derive signals that capture such evolution.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

Fig. 6. Posts per-day for the datasets with the *churn window* and *churn control window* highlighted with red and blue overlays respectively.

5.1. Period-Variation Signals

We are provided with different lifecycle-period-specific dimensions (i.e. in-degree) modelled in the form of discrete probability distributions. This therefore allows us to use

information theory measures to quantify the variance and change in such distributions over time. We begin by looking at the variation of each of such dimensions' distributions through entropy. To aid legibility we define a *measure*, or *measurement*, as constituting the combination of a user dimension (e.g. in-degree) assessed along a single development indicator (e.g. entropy), thus an example measure would be *in-degree period entropy*. Entropy describes the amount of variation within a random variable, and therefore provides a useful means to gauge how much a given user is varying: (i) the people with whom he is communicating, and (ii) the terms that he is using within his posts. We define the entropy of an arbitrary probability distribution P as:

$$H(P) = -\sum_{x} p(x) \log p(x) \tag{8}$$

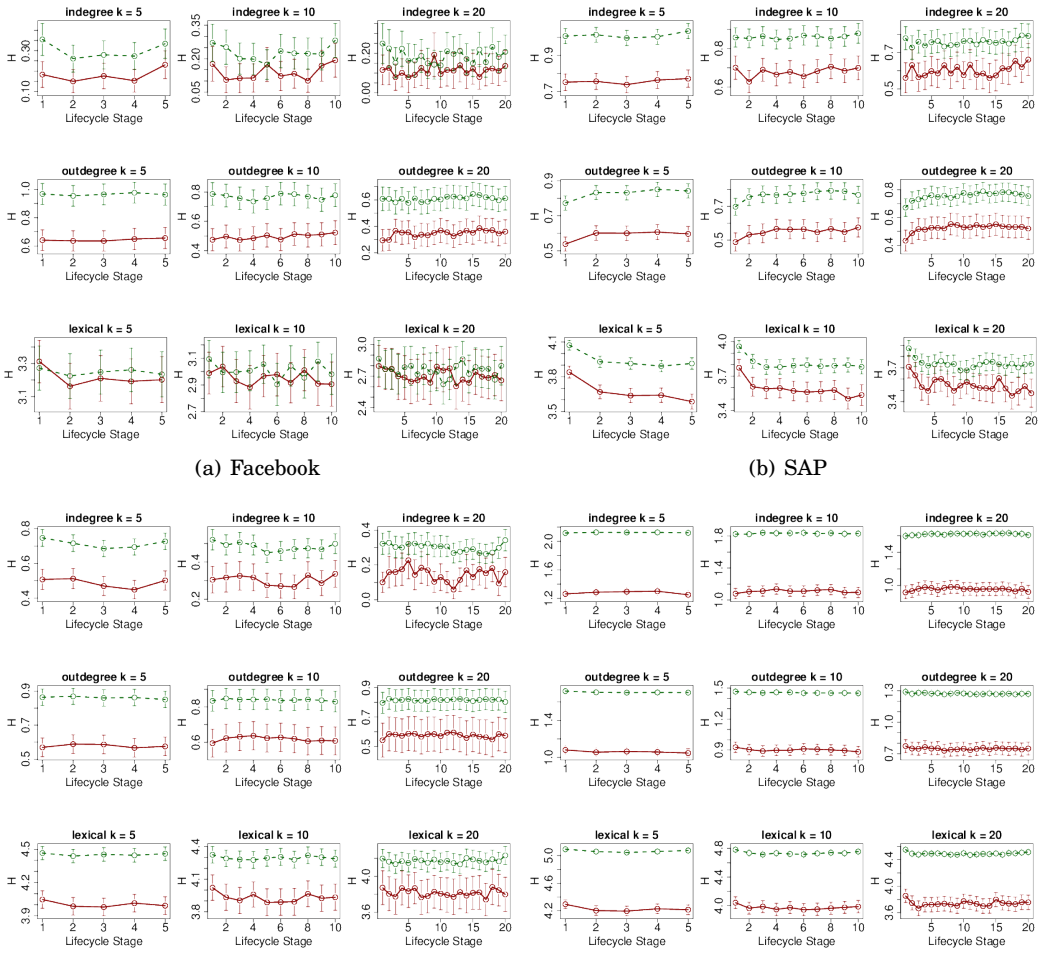
For each platform we derived the entropy of each user in each of his individual lifecycle periods based on the in-degree, out-degree and term distributions. We then recorded the mean of these entropy values over each lifecycle period, thereby providing an assessment of the general changes that users go through. By segmenting the users up into their respective classes (churner or non-churner), we were then able to plot the development signals of these different classes over time - showing the mean and 95% confidence intervals for the different lifecycle stages in Figure 7. One thing becomes clear from an initial inspection of these development signals: churners vary significantly less than non-churners, and that this is consistent across all platforms, all dimensions (i.e. in-degree, out-degree, lexical terms), and lifecycle fidelities (settings of k). It is particularly stark when one focuses on Boards.ie: in this case, there is markedly little variation in the users with whom churners interact, and the language they use. For other platforms, in particular for Facebook, as the lifecycle fidelity is increased the development signals become closer - in particular for the lexical terms used where there is little to distinguish between churners and non-churners in terms of their language.

5.2. Retrospective-Comparison Signals

We now move on to examining the changes that users go through relative to earlier lifecycle periods, this is quantified using the cross-entropy between one lifecycle period's distribution and an earlier lifecycle period that minimises the cross-entropy. To inform such cross-period assessment we examined the users' in-degree, out-degree and lexical term distributions across lifecycle periods by computing the cross-entropy of one probability distribution with respect to another distribution from a lifecycle period (i.e. retrospectively), and then selected the distribution that minimises cross-entropy. Assuming we have a probability distribution (P) formed from a given lifecycle period (s), and a probability distribution (Q) from an earlier lifecycle period, then we define the cross-entropy between the distributions as follows:

$$H(P, Q) = -\sum_{x} p(x) \log q(x) \tag{9}$$

Figure 8 shows the in-degree, out-degree and lexical period cross-entopies for the four platforms. As above, we derived these plots by deriving the measures for each user across their disparate lifecycle periods, before then deriving the mean measure for each period for both the *churners* and *non-churners*, together with their 95% confidence intervals. We note that across all of the plots churner signals are lower in magnitude than non-churners signals, indicating that the properties of the non-churners tend to have a greater divergence with respect to earlier properties than the churners.

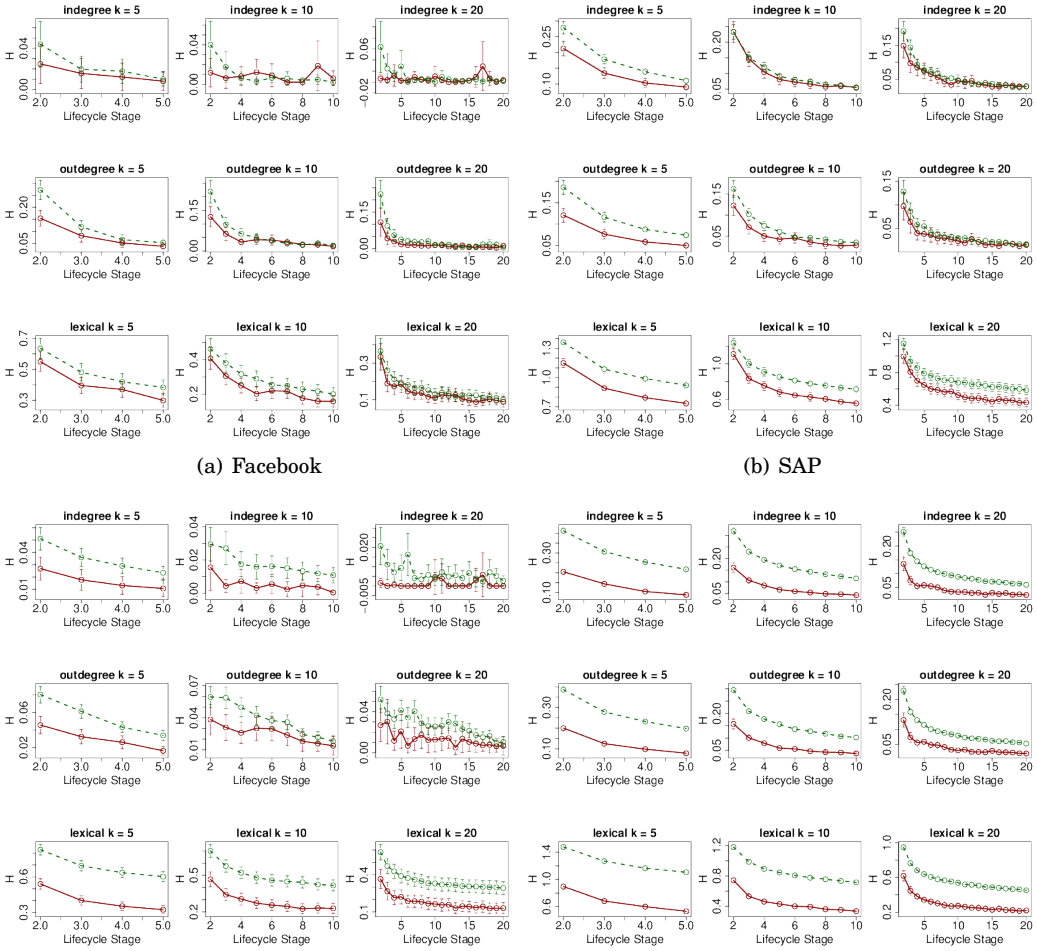


(a) Facebook (b) SAP

(c) Serverfault (d) Boards.ie

Fig. 7. Period entropy distribution for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics. The green dashed line shows the non-churners, while the red solid line shows the churners.

This suggests that churners' behaviour is more *formulaic* than non-churners, that is they exhibit less divergence from what has occurred beforehand. Looking at the different user properties in isolation we see that, in general, the curve of churners and non-churners diminishes towards the end of their lifecycles but with different gradients. For the *in-degree* property, churners on Facebook and Server Fault have noticeably flatter curves which reduce at a slower rate than the non-churners, while for SAP the curves are similar for churners and non-churners. For users' *out-degree* the cross-entropy of churners converges on a lower value much sooner than non-churners across all platforms; this effect is particularly marked for Server Fault indicating that non-churners tend to vary the people with whom they are communicating throughout their lifecycles markedly more than churners. For the cross-entropy of users' lexical term distributions we find the signals of churners and non-churners to follow a similar curvature (converging on a limit with a decaying rate) but with different magnitudes.



(a) Facebook (b) SAP

(c) Serverfault (d) Boards.ie

Fig. 8. Retrospective cross-entropy distributions for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics. The green dashed line shows the non-churners, while the red solid line shows the churners.

5.3. Community-Comparison Signals

For the third inspection of user lifecycles and how user properties change, we examined how users compare with the platform in which they are interacting over the same time interval. We used the in-degree, out-degree and lexical term distributions and compared them with the same distributions derived globally over the same time periods. For the global probability distributions we used the same means as for forming user-specific distributions, but rather than using the set of posts that a given user had authored (P_{u_i}) to derive the probability distribution, we instead used all posts. For instance, for the global in-degree distribution we used the frequencies of received messages for all users. Given the discrete probability distribution of a user from a time interval (P_s), and the global probability distribution over the same time interval (Q_s), we derived the cross-entropy as above between the distributions. ($H(P_s, Q_s)$).

Figure 9 shows the development signals when comparing users' individual distributions with that of the community platforms over the same time-interval (i.e. lifecycle period). Again, as with retrospective-comparison signals, we find churners' signals to have a lower magnitude than non-churners suggesting that non-churners' properties tend to diverge from the community as they progress throughout their lifetime within the online community platforms. There are some noticeably noisy signals however, in particular for Facebook and the in-degree distribution and lexical term distributions of non-churners. Generally for each signal we see a growing curve towards later lifecycle periods for both churners and non-churners, while the magnitudes of the curves are the salient differentiating features.

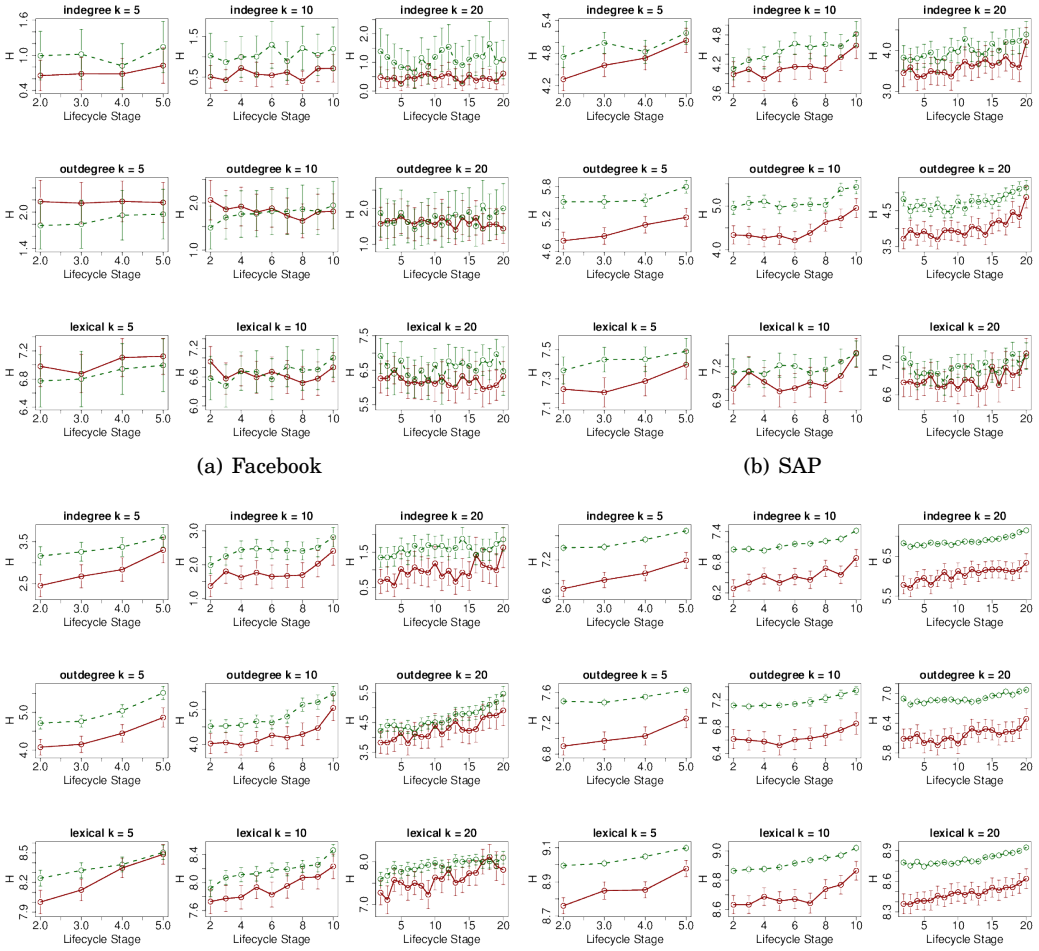
These findings both corroborate, and contribute further to, that found in the work of Danescu-Niculescu-Mizil et al. [Danescu-Niculescu-Mizil et al. 2013] as follows. Firstly, our examination looks at the evolution of users' social and lexical dimensions relative to their community of interaction over each lifecycle period for *differing* lifecycle fidelities and for *both* churners and non-churners. As a result, we find that for the top-most lifecycle fidelity ($k = 20$) that non-churners' language adapts to the community - what Danescu-Niculescu-Mizil et al. refer to as '*linguistic adolescence*' - before then diverging away: thereby corroborating what has been found previously. However, our findings show that for different social dimensions, for reduced lifecycle fidelities, and for churners this is not the case: instead churners tend to show clear signals of *diverging* from the community with little adaptation occurring. Therefore our examination with different fidelities and dimensions has extended current investigation in this space to reveal new insights into the development of churners and non-churners across other, additional platforms.

5.4. A Proposed Theory for Churner Development

As with any emerging body of research, little is known of how and why churners develop in the manner that they do. We believe that the observations offered above provide the basis for a theory of how churners develop differently to non-churners. Our hope is that by outlining the principles of this theory, that can be operationalised as above, that other researchers can repeat our experiments on alternative datasets - not necessarily restricted to online community platforms - in order to gather observations that either support or refute the theory. The (testable) principles of this proposed theory for churner development are as follows:

— *Churners vary their behaviour less and are thus more formulaic.*
— *Churners diverge from community norms over time, but less so than non-churners.*

The former of these principles is based on our observations of the entropy of users' dimensions and the retrospective-comparison entropy too; as in both cases churners exhibited behaviour with less variation and that diverged less from previous observed behaviour. The latter of these principles however is based on our comparison of the users with their communities of interaction, where both churners and non-churners diverged from community norms (i.e. we observed an increase in the cross-entropy over time for all measures) yet churners' divergence was clearly lower than that of normal users. To further examine the development of churners and non-churners we plotted the cumulative distribution of *innovations* per-lifecycle period between the two groups. We define an *innovation* here on a *per-user* basis where an innovation occurs when a user: (i) is contacted by a new person (in-degree); (ii) contacts a new person himself (out-degree), or; (iii) uses a new term (lexical). Figure 10 shows these distributions over the churners and non-churners. One thing is immediately obvious: *in-general* all users follow a similar pattern of communication and language used in terms of innovations; the differences occur when we consider the variance of user behaviour (as above) and



(a) Facebook (b) SAP

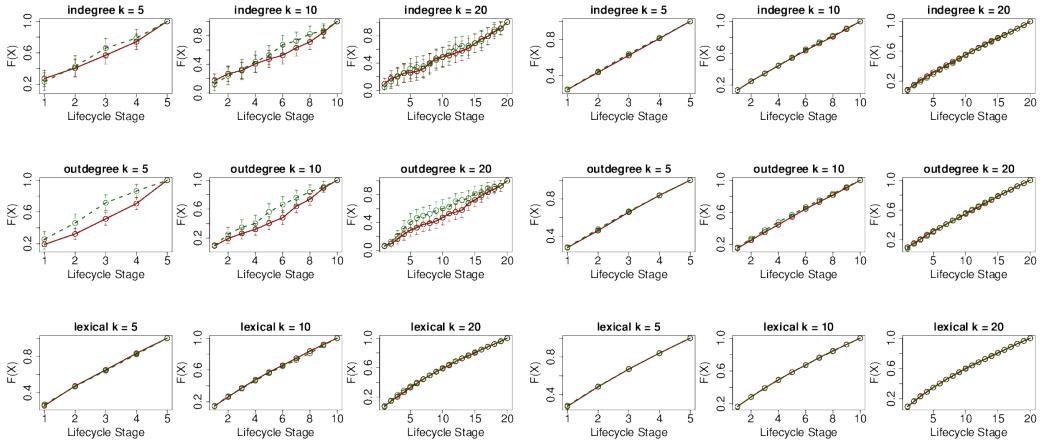
(c) Serverfault (d) Boards.ie

Fig. 9. Community cross-entropy distributions for different fidelity settings (k) for users' lifecycles and different measures of social (indegree and out degree) and lexical dynamics. The green dashed line shows the non-churners, while the red solid line shows the churners.

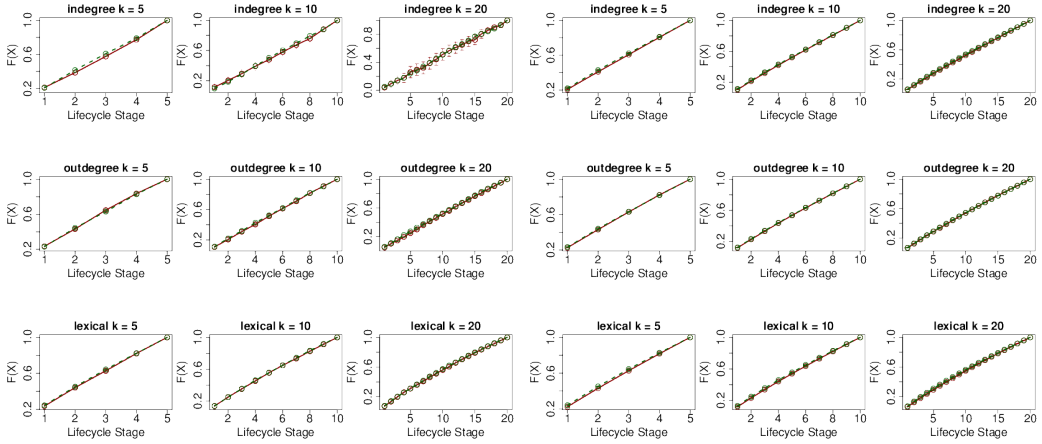
the individuals the person is contacting, and the language they are using, both *relative to their earlier behaviour* and *relative to their community of interaction*.

6. FEATURE ENGINEERING

In this section we describe the process of mining features from the development signals of users that are then used for the detection of churners. The earlier examination of how users develop based on different dimensions (indegree, outdegree, lexical) indicated differences between churners and non-churners across all lifecycle fidelities. Our aim therefore was to leverage this information in the form of features that some model could then be induced from for churn detection. We identified two types of features that could be harnessed: *static* features, that relate to the absolute value of a given dynamic (e.g. indegree) and entropy measure (e.g. period entropy) at a given lifecycle stage; and



(a) Facebook (b) SAP



(c) Serverfault (d) Boards.ie

Fig. 10. Cumulative innovation distributions across the platforms, lifecycle fidelities, and different measures for both churners and non-churners.

rate features, that measure the change in a given static feature from lifecycle stage to the next.

6.1. Static Features

We define the following function: $f^k_{dh}(u, s)$ that returns the value, or magnitude, of a given dynamic ($d \in D$) and entropy measure ($h \in H$) based on lifecycle fidelity k for user u in lifecycle stage s; the codomain is therefore a real valued number of the entropy value. As we have three dynamics, three entropy measures (period entropy and historical comparison entropy, and cross-community entropy) we measure static features for each lifecycle stage across the k stages, resulting in $|D| \times |H| \times k = 9k$ static features per user.[3]

[3]Therefore, by varying $k = \{5, 10, 20\}$ we produce 45, 90 and 180 static feature respectively, depending on the lifecycle fidelity that we choose.

Table III. Feature Counts for static and rate features at different lifecycle fidelity settings (k)

Fidelity (k)	#Static	#Rate	#Total
5	45	36	81
10	90	81	171
20	180	171	351

6.2. Rate Features

Our second type of features captures the change in a given static feature that a user goes through, from one lifecycle stage to the next. In this context we can capture how a user is *evolving* given the observed dynamic. To derive the rate of change of a given dynamic ($d \in D$) and entropy measure ($h \in H$) based on lifecycle fidelity k for user u *into* stage s, we used the following proportionate growth rate function:[4]

$$\delta^k_{dh}(u,s) = \frac{f^k_{dh}(u,s) - f^k_{dh}(u,s-1)}{f^k_{dh}(u,s-1)} \tag{10}$$

As with the static features, we have three dynamics and three entropy measures, however we only have $k-1$ lifecycle stages - as we cannot measure the rate of change of a static feature prior to the primary lifecycle stage - this results in $|D| \times |H| \times (k-1) = 9(k-1)$ rate features.

Table III summarises the number of features that are produced using this feature engineering technique for static features, rate features, and then in total, for different lifecycle fidelities: i.e. variance of k.

7. CHURNER DETECTION

In this section we now turn to the task of picking out which users will churn from the online community platforms, and which will not. Unlike previous approaches to detecting churners, we do not focus on mining social network features of users (i.e. in-degree at time t) or the number of previous friends that have churned, instead we use the features defined above that are mined from user development signals. We begin this section by first defining three detection models, before then evaluating their performance in detecting which users will churn.

7.1. Detection Models

The goal of the churn detection models is to identify which users are churners from their development signals to date. In essence, we can represent this problem of recovering the churn labels vector across all users ($\mathbf{y} \in \{0,1\}^m$) using the development signal features ($\mathbf{X} \in \mathbb{R}^{m \times n}$) and an induced weight vector ($\mathbf{b} \in \mathbb{R}^n$) such that: $\mathbf{y} \approx f(\mathbf{X}|\mathbf{b})$, with m users and n features in the given dataset. Each platform's dataset is provided in the following format: $D = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{0,1\}\}$, and we distinguish between the training, and test users splits using D_{train} and D_{test} respectively.

Our aim therefore is to induce some function f that has as its domain a given user's development signal modelled as a feature vector ($\mathbf{x}$) and the churn probability as its co-domain, hence: $f : \mathbb{R}^n \rightarrow [0,1]$. To induce this function we used three methods: (i) logistic regression; (ii) a dual-gaussian sequence model; and (ii) a linear model with elastic-net regularisation. We now explain each in turn.

[4]N.b. in this context, $\delta^k_{dh}(u,s) \equiv (\partial d \partial h)/\partial s$

7.1.1. Detection Model 1: Logistic Regression Model. We used the logistic regression model to predict the conditional probability of user u_i churning as follows:

$$Pr(Y = 1 \mid \mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{b}^\intercal \mathbf{x}_i}} \tag{11}$$

The model's coefficients ($\mathbf{b}$) define the weight attached to each feature within the linear model ($f(\mathbf{x}_i|\mathbf{b}) = \mathbf{b}^\intercal \mathbf{x}_i$). In order to derive the model's coefficients we used the maximum likelihood estimation $\hat{\boldsymbol{\beta}}$ of the model's coefficients. Following fitting, the derived model is used to predict the churn probability of each user within the test portion of the data. We also include a regularised version of the model that uses the L2 (ridge) penalty for control for overfitting.

7.1.2. Detection Model 2: Dual-Gaussian Sequence Model. When inspecting each different measurement (e.g. the period entropy of users' in-degree at lifecycle stage 1) for both churners and non-churners, we plotted the the development signals for both sets of users along with their 95% confidence intervals.. Our second model, presented initially in our earlier work [Rowe 2014], is based upon the premise that a given measurement (m) at a particular lifecycle stage (s) is normally distributed. Thus, for each measurement we have two signals (one for churners and one for non-churners) that each correspond to a *sequence* of Gaussians measured over the k lifecycle stages. We define this more concretely as follows: given a measurement m,[5] and a lifecycle stage s drawn from a set of lifecycle stages S, we *assume* that m is normally distributed at s and thus characterised by $\mathcal{N}\big(\hat{\mu}_{m,s}, (\hat{\sigma}_{m,s})^2\big)$ where $\hat{\mu}_{m,s}$ and $\hat{\sigma}_{m,s}$ denote the maximum likelihood estimates of the mean and standard deviation respectively. Then the Gaussian Sequence of m is defined as follows:

$$G_m = \Big(\mathcal{N}\big(\hat{\mu}_{m,1}, (\hat{\sigma}_{m,1})^2, \mathcal{N}\big(\hat{\mu}_{m,2}, (\hat{\sigma}_{m,2})^2, \ldots, \mathcal{N}\big(\hat{\mu}_{m,|S|}, (\hat{\sigma}_{m,|S|})^2\big)\Big) \tag{12}$$

In essence we have two *competing* gaussian distributions at a particular lifecycle stage: the *churn gaussian*, formed from measurements of the known churner users, and; the *non-churn gaussian*, formed from measurements of known non-churners. We can therefore specify the probability of the user u belonging to the churner class based on measurement m and lifecycle stage s as follows:

$$\begin{aligned} P(u|\beta_{m,s}) \propto \Big[& \beta_{m,s} \mathcal{N}\big(f(u,m,s)|\hat{\mu}^c_{m,s}, (\hat{\sigma}^c_{m,s})^2\big) \\ & - (1 - \beta_{m,s}) \mathcal{N}\big(f(u,m,s)|\hat{\mu}^n_{m,s}, (\hat{\sigma}^n_{m,s})^2\big) \Big]_+ \end{aligned} \tag{13}$$

Above we have modified the maximum likelihood estimates for the mean and standard deviation to correspond to the churner (c) and non-churner classes (n). We also incorporated the slack variable $\beta_{m,s}$ which is indexed by the measurement and lifecycle stage, and controls for over-penalising class membership - we learn this parameter as $\beta_{m,s} \in \mathbf{b}$. The subtraction of the churn-distribution membership probability by the $\beta_{m,s}$-scaled non-churn-distribution membership probability is wrapped within the positive value operand $[]_+$ in order to return a non-negative value. We can then calculate the joint churn probability over observed measures and lifecycle stages as follows - we term this the *Dual-Gaussian Sequence Model*:

[5] We defined a measurement, or measure, earlier as the combination of a given dynamics (e.g. in-degree) and a given development indicator (e.g. period entropy); hence M is the set of all 9 possible measurements.

$$Q(u|\mathbf{b}) = \prod_{m \in M} \prod_{s \in S} \rho \Big[\beta_{m,s} \mathcal{N}\big(f(u,m,s)|\hat{\mu}^c_{m,s}, (\hat{\sigma}^c_{m,s})^2\big) - (1-\beta_{m,s}) \mathcal{N}\big(f(u,m,s)|\hat{\mu}^n_{m,s}, (\hat{\sigma}^n_{m,s})^2\big) \Big]_+ \quad (14)$$

Here ρ acts as a *smoother* to chain together zero-probability values.[6] Now, for this detection model, our objective is to minimise the squared-loss between a user's forecasted churn probability and the observed churn label - given that the former is in the closed interval $[0,1]$ and the latter is from the set $\{0,1\}$ - our parameters are L2-regularised to control for over-fitting:

$$\arg\min_{\mathbf{b}^*} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \big(y_i - Q(u|\mathbf{b})\big)^2 + \lambda ||\mathbf{b}||^2 \quad (15)$$

Using this objective, we then used stochastic gradient descent to calculate the setting of each $\beta \in \mathbf{b}$ by minimising the loss between a single user's forecasted churn probability and his actual churn label (i.e. either 0 - did not churn - or 1 - did churn). We experimented with two learning procedures: stochastic gradient descent (SGD), and dual-stochastic gradient descent (D-SGD) - the latter being a novel contribution in our prior work [Rowe 2014] - however we found the difference in performance to be insignificant and thus favoured the former given its reduced computational complexity (i.e. $\mathcal{O}(m)$ per learning epoch rather than $\mathcal{O}(m \times m)$). We refer the reader to our prior work [Rowe 2014] for a more thorough presentation of the models and learning procedures used.

7.1.3. Detection Model 3: Linear Model with Elastic-Net Regularisation. Our third detection model combines a linear model with elastic-net regularisation to predict the probability of a given user churning using the linear combination of the user's feature vector and the learnt weight vector ($\mathbf{b}$). We combine both L1 and L2 regularisation within the predictive function to control for overfitting in the training segment by using α-weighting between the L1 and L2 penalties (i.e. Lasso and Ridge). To learn the parameters of the model we used two learning approaches: (i) *Average* Coordinate Descent proposed by Friedman et al. [Friedman et al. 2010], and; (ii) Stochastic Coordinate Descent, which we propose within this paper.[7]

Learning with Average Coordinate Descent. Our first learning routine, from [Friedman et al. 2010], seeks to learn the linear churn prediction by minimising the following objective function, using the entire training dataset:

$$\min_{\mathbf{b}*} \frac{1}{2N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \big(y_i - \mathbf{b}^\intercal \mathbf{x}_i\big)^2 + \lambda(1-\alpha)\frac{1}{2}||\mathbf{b}||_2^2 + \lambda\alpha||\mathbf{b}||_1 \quad (16)$$

Learning the model therefore requires updating the parameters in the n-dimensional weight vector ($\mathbf{b}$) based on the prediction error. We therefore update one parameter's value at a time based on the derivative with respect to β_j, derived using the chain rule from Equation 16, using the average error across the training dataset:

[6] See Appendix A for an examination of the effects tuning ρ on model performance.
[7] This is a modified version of Friedman et al.'s approach to update a given parameter vector's element one-at-a-time.

$$\nabla_j = -\frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \mathbf{x}_{ij}(y_i - \mathbf{b}^\intercal \mathbf{x}_i) + \lambda(1-\alpha)\beta_j + \lambda\alpha \tag{17}$$

The parameter ($\beta_j \in \mathbf{b}$) is then updated within the model using the following update rule:

$$\beta_j = \beta_j - \eta\nabla_j \tag{18}$$

Learning with Stochastic Coordinate Descent. The second learning routine extends Average Coordinate Descent into a Stochastic setting, by updating each element of the parameter vector using a single training set's instance and thus its resultant error. We modified the objective function to enable a gradient to be calculated for a single instance as follows:

$$\arg\min_{\mathbf{b}*} \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \left(\frac{1}{2}\big(y_i - \mathbf{b}^\intercal \mathbf{x}_i\big)^2 + \lambda(1-\alpha)\frac{1}{2}||\mathbf{b}||_2^2 + \lambda\alpha||\mathbf{b}||_1 \right) \tag{19}$$

Based on this, we calculated the derivative of $\beta_j \in \mathbf{b}$ using a single instance (i.e. $(\mathbf{x}_i, y_i) \in D$) at a time as follows, by applying the chain rule to Equation 19:

$$\nabla_{ij} = -\mathbf{x}_{ij}(y_i - \mathbf{b}^\intercal \mathbf{x}_i) + \lambda(1-\alpha)\beta_j + \lambda\alpha \tag{20}$$

Unlike with the average coordinate descent model, in this instance we use the stochastic learning routine of shuffling the order of $\mathcal{D}$ each training epoch and then iterating through the set of training instances, deriving the error in prediction and the derivative of each parameter and thus updating accordingly. As a result our update rule is the following, for a given training instance with index i:

$$\beta_j = \beta_j - \eta\nabla_{ij} \tag{21}$$

With the linear model using elastic-net regularisation we have two hyperparameters than must be tuned: the learning rate η and the regularisation weight λ. The use of elastic-net regularisation means that we can examine the spectrum between using solely a lasso penalty ($\alpha = 1$), or solely a ridge penalty ($\alpha = 0$), or somewhere in the middle ($\alpha = 0.5$). Rather than tuning α as a hyperparameter, we adopted a different approach and *indexed* linear models using the following settings: $\alpha = \{0, 0.5, 1\}$, thereby tuning a hyperpameter vector $\theta = \{\eta, \lambda\}$, for each setting. We explain in the following section the model tuning approach that was applied.

For the above linear model, our aim is to ensure that the induced function's codomain is constrained to the closed interval $[0, 1]$. However when learning, the resultant parameter vector can exceed those bounds. In order to control for this, when inducing the predicted probabilities of a set of users in a given test set we *normalise* the codomain to restrict it to the closed probability interval. For an arbitrary user vector $\mathbf{x}_i$, and having computed all predicted probabilities (which may exceed the bounds of $[0, 1]$) we normalise the churn probability of i as follows: $Pr(\mathbf{x}_i|\mathbf{b}) = f(\mathbf{x}_i|\mathbf{b})/(f_{\max} - f_{\min})$.

7.2. Experiments

In order to compare the above models and judge how well they fare against existing work, we conducted a series of experiments: firstly, to tune the different models' hyperparameters; and secondly, to apply them to a held-out test portion of users. We begin by first defining our experimental setup.

Table IV. Number of instances within the training and testing datasets used for the experiments across the different lifecycle fidelities. The number of instances decreases as the lifecycle fidelity increases as we require each user to have posted double the fidelity number of posts.

Fidelity	Facebook		SAP		ServerFault		Boards.ie	
	|Train|	|Test|	|Train|	|Test|	|Train|	|Test|	|Train|	|Test|
5	306	72	1,099	302	1,229	338	6,338	1,700
10	204	48	716	205	688	177	4,979	1,334
20	123	27	448	129	375	84	3,635	995

7.2.1. Experimental Setup. As mentioned above, for the four platforms' datasets we divided users into training and testing sets using an 80%:20% split respectively. Given that we experimented with different lifecycle fidelities ($k = \{5, 10, 20\}$), this reduced the number of users, and thus instances, in our dataset and hence the training and testing splits - Table IV shows the number of instances per split and lifecycle fidelity. The reason for this reduction is that we require each user to have posted $2k$ posts prior to the churn cutoff point at which we perform our analysis, this provides sufficient information from which to mine users' development signals from. For setting up our experiments we first performed model tuning (using the training set for each platform), and then applied the tuned models to the held-out test split - in this latter setting we repeatedly applied each tuned model 25 times and took the average area under the Receiver Operator Characteristic curve (ROC).

Model Tuning. For our experiments we had two models to tune: the dual-gaussian sequence model and the linear model using elastic-net regularisation; both of which require their hyperparameters to be selected. To tune the hyperparameters (λ and η) we ran 10-fold cross validation over the training splits and recorded the average ROC; we then selected the best performing hyperparameter combinations. Both λ and η were varied through $\{10^{-3}, 10^{-2}, 10^{-1}, 0.2, \ldots, 0.5\}$. For the dual-gaussian sequence model we also *tuned* ρ as an additional hyperparameter in this model - see Appendix A for more information on this. While for the linear model with elastic-net regularisation we tuned three variants of the model with different settings for α where $\alpha = \{0, 0.5, 1\}$ - this allowed us to examine the performance of solely L1 (i.e. lasso) regularisation ($\alpha = 1$), solely L2 (i.e. ridge) regularisation ($\alpha = 0$), or combining both equally ($\alpha = 0.5$). The logistic regression model did not require the tuning of hyperparameters, however the regularised version of the logistic regression did require tuning of the regularisation weight (λ). Once model tuning was completed all tested models, including the baseline to be defined below, was trained using the entire training split and, using the best tuned hyperparameters should the model require them, applied to the held-out test split of users.

Baselines. To provide a suitable baseline against which to compare our approach we implemented the approach defined in Karnstedt et al.'s work [Karnstedt et al. 2011]. This approach used features derived from the social network of users: *in-degree, out-degree, closeness-centrality, betweenness-centrality, reciprocity, average number of posts in initiated threads, average number of posts within participated threads, popularity (% of user authored posts that receive replies), initialisation (% of threads authored by the user), and polarity*. We first tested the J48 classifier, as used in [Karnstedt et al. 2011], but found this to perform poorly[8] therefore we used the Naive Bayes classifier instead which yielded the best performance of the available classification models at

[8]We also tested support vector machines and the perceptron classifier.

Table V. Area under the Receiver Operator Characteristic (ROC) Curve results for the different proposed models

	k	Baseline	Logistic	SVM	Dual-$\mathcal{N}$	Linear Elastic-Net (ACD / SCD)		
			None / Ridge			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
Facebook	5	0.461	0.560 / **0.571**	0.491	0.500	0.491 / 0.438	0.484 / 0.432	0.482 / 0.450
	10	0.492	0.599 / **0.625**	0.466	0.570	0.558 / 0.542	0.554 / 0.542	0.561 / 0.538
	20	0.444	0.528 / 0.528	0.500	0.664	0.679 / 0.544	0.674 / 0.523	**0.685** / 0.554
SAP	5	0.497	0.596 / **0.609**	0.599	0.572	0.518 / 0.596	0.505 / 0.539	0.498 / 0.604
	10	0.495	0.616 / 0.631	0.636	0.553	0.531 / 0.614	0.541 / 0.639	0.489 / **0.645**
	20	0.582	0.597 / 0.611	0.638	0.525	0.547 / **0.706**	0.518 / 0.701	0.548 / 0.675
ServerF	5	0.530	0.612 / **0.620**	0.569	0.523	0.529 / 0.598	0.520 / 0.590	0.527 / 0.604
	10	0.546	0.505 / 0.494	0.554	0.568	0.523 / 0.584	0.491 / **0.615**	0.510 / 0.608
	20	0.530	0.503 / 0.508	-	0.502	0.512 / 0.536	0.507 / 0.564	0.509 / **0.563**
Boards	5	0.611	0.513 / 0.514	0.522	0.542	0.540 / **0.648**	0.492 / 0.571	0.516 / 0.647
	10	0.593	0.515 / 0.515	0.501	0.501	0.497 / **0.621**	0.493 / 0.595	0.517 / 0.597
	20	0.553	0.521 / 0.521	0.503	0.500	0.585 / **0.624**	0.551 / 0.613	0.587 / 0.622

our disposal.[9] We also included the Support Vector Machine (SVM) classifier with L2 regularisation[10] as an additional model against which to compare our models' performance.

7.2.2. Results. Table V presents the ROC values that we produced when running the various models over each platform's test split and varying the lifecycle fidelity. The first thing that is striking about these results is how well the linear model with elastic net regularisation performs, in particular in relation to the logistic regression model. We achieve performance that is significantly ($p < 0.0001$) better than all other models for 8 out of 12 of the experimental settings;[11] thereby demonstrating how accurately we can detect who is likely to churn and who is likely to remain by mining user development information. Our results show that we consistently surpass the SVM baseline,[12] and the use of our stochastic coordinate descent learning routine *generally* yields the best performance.

The dual-quassian sequence model did not fare so well, in particular when compared against the logistic regression model. The reason for such poor performance is likely due to the complexity of the model, despite adding L2 regularisation, rendering the model's capacity to generalise to unseen users' development information limited. We tuned the smoothing parameter ρ as with the other hyperparameters, unlike setting $\rho = 0.1$ as in our prior work, yet this did not yield a satisfactory level of performance. Different lifecycle fidelities rendered no clear pattern in preference between α settings for the linear model with elastic net regularisation, nor did one platform's results demonstrate that one model was better than the rest. Appendix B shows the heat maps produced from tuning the hyperparameters for the various α-indexed linear models with elastic-net regularisation and the two learning routines used.

8. DISCUSSION AND FUTURE WORK

Our churn detection approach makes use of the development signals that users exhibit along both social and lexical dimensions in order to differentiate between who will churn and who will remain within the online community platform. In comparison with the work of Kernstadt et al. [Karnstedt et al. 2011], we use *dynamic* user information, that is: information about *how* the user has evolved throughout his life-

[9]We used the Weka machine learning toolkit for the classification models that were tested.
[10]This also used the Weka machine learning toolkit with the LibLinear library.
[11]N.b. Statistical significance testing was performed by comparing the distribution of ROC values from the 25 applications to the test set using Student's T-Test between models.
[12]N.b. When applying the SVM model to the ServerFault dataset with $k = 20$ the model repeatedly failed.

cycle to date, while Kernstadt et al. adopt *static* user information derived using a 6-month window prior to a given analysis point. The results from using this information within the presented linear model empirically demonstrate the superior performance that we can achieve. Additionally, we examined the use of social and lexical information in characterising differences in the development of churners and non-churners - in both cases using basic relative-frequency distributions to encapsulate such information over a variety of platforms. Existing work of Danescu-Niculescu-Mizil et al. [Danescu-Niculescu-Mizil et al. 2013] presented a more *in-depth* analysis of lexical information (e.g. first person singular pronouns, number of words) and how this can be used to forecast user lifespan. Our future work will examine how such information can be incorporated into our approach, and also how additional, more nuanced and fine-grained, social network features can be used: ultimately to boost our current prediction accuracy.

In this paper we have proposed a theory of how churners develop and how such evolution signals differ from non-churners; the listed principles are based upon the observations that we have made in this work across the four platforms, the three lifecycle fidelity settings, and the different examined dimensions (i.e. in-degree, out-degree, lexical terms). Our hope is that these principles, which can be operationalised in the approach that we have presented (i.e. using measures from information theory), will be tested by our colleagues from the data mining and social computing communities. Future work in this area will test how these principles hold along other dimensions, that can act as a measurement of behaviour variation (e.g. sharing behaviour, reciprocation of interaction, etc.), and on other platforms and domains also (i.e. not restricting this application to merely online community platforms, but subscription-based services also such as online gaming). We also wish to follow up our *quantitative* analyses of user behaviour, and its development for churners, with *qualitative* feedback on *why* users choose to leave. At present we have little idea behind such rationale, however the creation of effective retention campaigns depends such an understanding.

The natural extension of the classification task described in our work is to examine how we can predict the *point* at which a user leaves. Initial steps in this area will be to examine prediction functions that produce a ranked-list of users, from soonest-to-latest churners, and thus explore how objective functions from approaches that learn to rank can be incorporated - without loss of generality our existing approach can be easily adapted to this setting. In exploring this area, we will be able to provide community mangers and service providers with actionable information about prioritising whom to contact in order to retain their services with the community, or subscription to the service.

9. CONCLUSIONS

In this paper we examined two key research questions: *Are there salient differences in how churners and non-churners develop?* And; *How can such development information be incorporated into an approach to detect churners?* We presented an approach to mine the development signals of users based on their lifecycles in online community platforms, and found that churners and non-churners exhibited different development signals that were consistent across the various online community platforms and lifecycle fidelity settings - thereby advancing over our prior work which concentrated on one fidelity setting [Rowe 2013]. A method was presented that extracted static and rate features from users' development signals, and we were able to utilise those features in a range of detection models to identify churners. One of our proposed models, a linear model with elastic net regularisation learnt using a stochastic coordinate descent strategy, significantly outperforming existing models including baselines from current work [Karnstedt et al. 2011; Rowe 2014]. We believe that our work has implications

on the churn prediction community that spans both social computing and data mining research, in particular when applied to both telecommunications and online gaming providers' systems, as it provides a new approach to user modelling that can be leveraged for accurately detecting churners.

REFERENCES

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 44–54.

Zoheb Borbora, Jaideep Srivastava, Kuo-Wei Hsu, and Dmitri Williams. 2011. Churn prediction in mmorpgs using player motivation theories and an ensemble approach. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*. IEEE, 157–164.

Kon Shing Kenneth Chung, Mahendra Piraveenan, and Shahadat Uddin. 2012. Community Evolution and Engagement through Assortative Mixing in Online Social Networks. *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* 0 (2012), 724–725. DOI:http://dx.doi.org/10.1109/ASONAM.2012.130

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities, In Proceedings of the World Wide Web Conference. *WWW'13* (2013).

Koustuv Dasgupta, Rahul Singh, Balaji Viswanathan, Dipanjan Chakraborty, Sougata Mukherjea, Amit A Nanavati, and Anupam Joshi. 2008. Social ties and their relevance to churn in mobile telecom networks. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. ACM, 668–677.

Gideon Dror, Dan Pelleg, Oleg Rokhlenko, and Idan Szpektor. 2012. Churn prediction in new users of Yahoo! answers. In *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 829–834.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* 33, 1 (2010), 1.

Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song. 2012. Evolution of Social-Attribute Networks: Measurements, Modeling, and Implications using Google+. *CoRR* abs/1209.0835 (2012).

Sanjay Ram Kairam, Dan J Wang, and Jure Leskovec. 2012. The life and death of online groups: predicting group growth and longevity. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 673–682.

Marcel Karnstedt, Tara Hennessy, Jeffrey Chan, Partha Basuchowdhuri, Conor Hayes, and Thorsten Strufe. 2010. Churn in social networks. In *Handbook of Social Network Technologies and Applications*. Springer, 185–220.

Marcel Karnstedt, Matthew Rowe, Jeffrey Chan, Harith Alani, and Conor Hayes. 2011. The effect of user features on churn in social networks. *Proceedings of the ACM WebSci* 11 (2011), 14–17.

Gueorgi Kossinets and Duncan J. Watts. 2006. Empirical Analysis of an Evolving Social Network. *Science* 311, 5757 (2006), 88–90. DOI:http://dx.doi.org/10.1126/science.1116869

Haewoon Kwak, Hyunwoo Chun, and Sue Moon. 2011. Fragile online relationship: a first look at unfollow dynamics in twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1091–1100.

Haewoon Kwak, Sue B Moon, and Wonjae Lee. 2012. More of a Receiver Than a Giver: Why Do People Unfollow in Twitter?. In *ICWSM*.

Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. 2008. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 462–470.

Kevin Lewis, Marco Gonzalez, and Jason Kaufman. 2012. Social selection and peer influence in an online social network. *Proceedings of the National Academy of Sciences* 109, 1 (2012), 68–72.

Julian McAuley and Jure Leskovec. 2013. From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews. In *Proceedings of World Wide Web Conference*.

D. McGowan, A. Brew, B. Casey, and N.J. Hurley. 2011. Churn Prediction in Mobile Telecommunications. In *Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science*.

Giovanna Miritello, Rubén Lara, Manuel Cebrián, and Esteban Moro. 2013. Limited communication capacity unveils strategies for human interaction. (7 April 2013). http://arxiv.org/abs/1304.1979

Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *SIGCOMM conference on Internet measurement (IMC '07)*. 29–42. DOI:http://dx.doi.org/10.1145/1298306.1298311

Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *J. Am. Soc. Inf. Sci. Technol.* 60, 5 (May 2009), 911–932. DOI:http://dx.doi.org/10.1002/asi.v60:5

Daniele Quercia, Mansoureh Bodaghi, and Jon Crowcroft. 2012. Loosing friends on Facebook. In *Proceedings of the 3rd Annual ACM Web Science Conference*. ACM, 251–254.

Matthew Rowe. 2013. Mining user lifecycles from online community platforms and their application to churn prediction. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 637–646.

Matthew Rowe. 2014. Predicting Online Community Churners using Gaussian Sequences. In *6th International Conference on Social Informatics*. Barcelona, Spain.

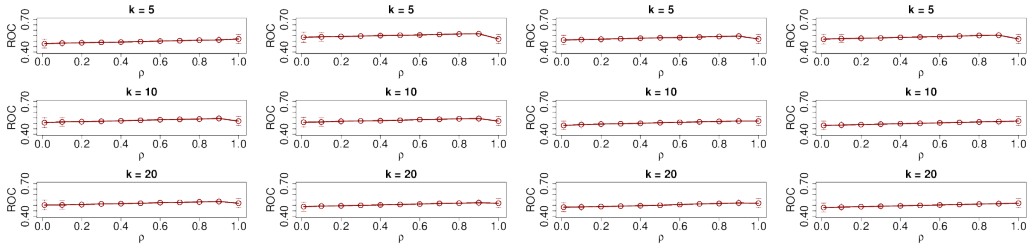
Daniel Stutzbach and Reza Rejaie. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 189–202.

Xiaohang Zhang, Zhiyu Liu, Xuecheng Yang, Wenhua Shi, and Qi Wang. 2010. Predicting customer churn by integrating the effect of the customer contact network. In *Service Operations and Logistics and Informatics (SOLI), 2010 IEEE International Conference on*. IEEE, 392–397.

Elena Zheleva, Hossam Sharara, and Lise Getoor. 2009. Co-evolution of social and affiliation networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1007–1016.

A. THE EFFECTS OF ρ ON GAUSSIAN-SEQUENCE PERFORMANCE

In our above-described Gaussian Sequence model, the parameter ρ acts as a *smoother* for zero-probabilities returned from the joint probability function. In our prior work we kept ρ fixed to 0.1, however in this paper we extended this experimental setting by testing different values of ρ and their impact on model performance. We therefore experimented with various settings of ρ to examine how different settings would impact accuracy during the tuning phase of the hyperparameters; along with the regularisation weight (λ), and the learning rate (η). In Figure 11 we have plotted the mean ROC values across all different hyperparameter settings (i.e. of $\theta = \{\lambda, \eta\}$) together with the 95% confidence intervals of these values. We find that the setting of ρ has *some* impact on the performance of the model where value of around $\rho = 0.9$ appear to the optimum. In fact the previous setting of $\rho = 0.1$ appears to be sub-optimal as this appears in the lower portions of the curves. This result implies that smoothing zero probabilities should be used with a *higher* weight, given the consistent *knees* we observe at higher values.



(a) Facebook (b) SAP (c) Server Fault (d) Boards.ie

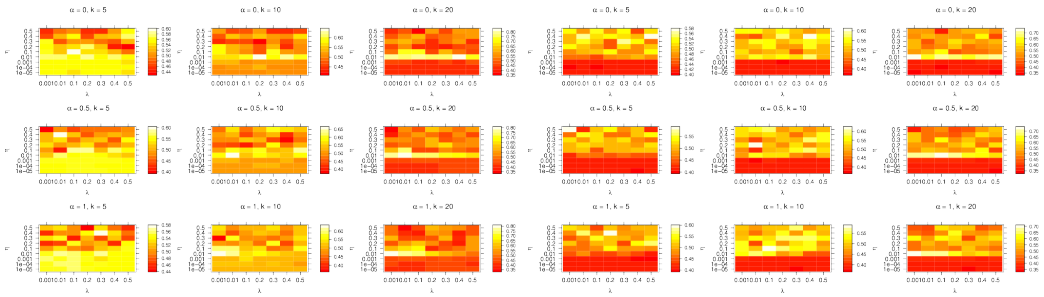
Fig. 11. Performance of different ρ-indexed Gaussian-sequence models as a function of variance settings of ρ.

B. HYPERPARAMETER TUNING FOR LINEAR DETECTION MODELS

One of the key challenges that we had when engineering the linear detection model using elastic-net regularisation, was tuning the hyperparameters of the model. As we were interested in seeing the impact of α-indexed models, where α controls the tradeoff between an L1 (hinge) penalty and an L2 (ridge) penalty when overfitting, we used 10-fold cross-validation over the training set and recorded the average ROC value. The learning procedure was varied between the use of average coordinate descent, and stochastic coordinate descent. Below we show the heat maps from tuning the learning rate (η) and the regularisation weight (λ) that make up each α-indexed linear model, learning using either ACD or SCD - where red indicates a reduction in ROC and thus an increase in the detection error, while white the contrary.

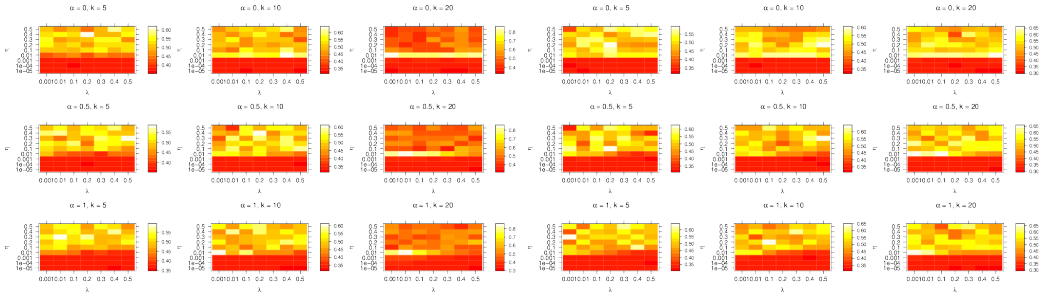
B.1. Learning via Average Coordinate Descent

When tuning we find that a lower learning rate, generally, results in a reduction in the ROC value, while the regularisation weight is mixed across both the platforms under inspection and the setting of α.



(a) Facebook

(b) SAP



(c) Server Fault

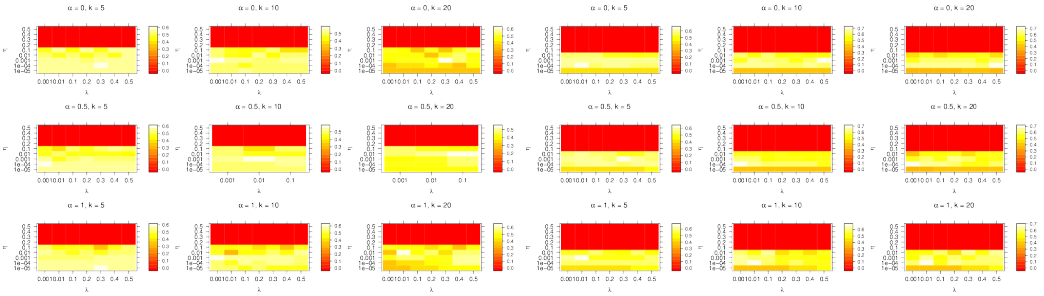
(d) Boards.ie

Fig. 12. Heatmaps showing the resultant ROC values following 10-fold cross validation with the different hyperparameters (λ and η) and indexed models (by k and by $alpha$) when learning using average coordinate descent (ACD).

B.2. Learning via Stochastic Coordinate Descent

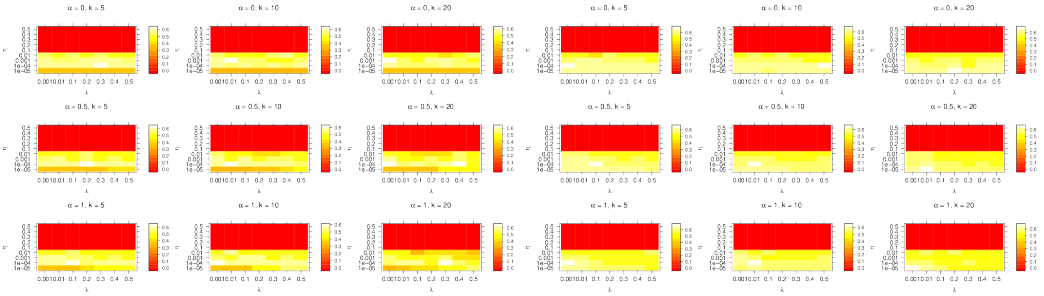
The heat maps produced from tuning the hyperparamters when learning using stochastic coordinate descent differ from those above - when using an average coordinate descent. We note that a reduced learning rate (η) yields higher ROC values across all models: given that we are updating the model using a stochastic learning

routine, this is not entirely surprising as larger updates lead to greater variance in the model's coefficients and thus moves away from a potential global optimum over time. Greater variance occurs when we consider the role of the regularisation weight (λ), but only at larger values of η, as at the lowest tested setting there is little variation of ROC values for different values of λ.



(a) Facebook

(b) SAP



(c) Server Fault

(d) Boards.ie

Fig. 13. Heatmaps showing the resultant ROC values following 10-fold cross validation with the different hyperparameters (λ and η) and indexed models (by k and by $alpha$).