Deep Learning Models for Large-Scale Collaboration

Aron Szanto and Sebastian Gehrmann
Department of Computer Science, Harvard University

Abstract

Members of loosely-coupled teams are often unaware of the work done by other team members. As such, identifying and sharing relevant information with a user about their teammates' activity can enhance collaborative success. Determining the relevance of information becomes more difficult as teams grow large and users dispersed. In this work, we introduce a model to predict information relevance in large-scale collaboration that addresses these difficulties. Our model also captures a one-to-many relationship between users and projects and allows for multiple types of user interaction. Using data from the GitHub Archive, we demonstrate that our approach can predict both user-repository interaction sequences and the likelihood of success for a given repository. Specifically, we show that our model predicts all types of user interactions at 65% precision and active user interactions at above 80%. It also predicts a repository's stars and forks, two important metrics of success, within 5.7 of the true value. These findings bode well for the creation of large-scale information sharing systems that facilitate successful teamwork at scale.

1. Introduction

Teamwork in loosely-coupled teams reduces task complexity for individuals and can lead to more efficient teamwork (Hutchins, 1995). To achieve this efficiency, a team requires an appropriate division of labor, information overlap between team members, and mutual knowledge of the shared goals, all of which depend on team members' being aware of others' work (Pinelle and Gutwin, 2005). Systems that aim to support such teamwork can leverage the history of the members' previous interactions with each other and with the project in order to determine the most helpful information to communicate (Olson and Olson, 2000). Large-scale collaborative platforms (LCPs) such as Wikipedia and GitHub are composed of multiple loosely-coupled teams and include users that can be members of multiple teams. This poses the challenge of how to extend current models of information sharing in loosely-coupled teamwork to incorporate the inter-project relationships of users and projects.

Recent work focuses on models that infer the relevance of different objects to a particular team member (Amir et al., 2016; Bălău and Utz, 2016; Xiao et al., 2016). In the LCP setting, an analogous goal is to infer which team has relevance for a given user. Users on LCPs can take on different roles in each project, influencing the way they interact with each one. In particular, users may contribute content, shape the design of a project, or passively consume its content.

Consider a user Alice who most actively contributes to a team Foo. A successful information sharing system would summarize recent changes about Foo for Alice, but be unlikely to recommend Foo as a team to watch. On the other hand, a user Bob, who collaborates with Alice on other projects, might benefit from a recommendation to watch Foo. Therefore, a system that intelligently shares information with users should not only take into account the teams relevant to a particular user, but also have knowledge of the types of interaction a user is likely to have with the project.

We formalize this problem as an extension

Email address: aszanto@college.harvard.edu
(Aron Szanto and Sebastian Gehrmann
Department of Computer Science, Harvard University)

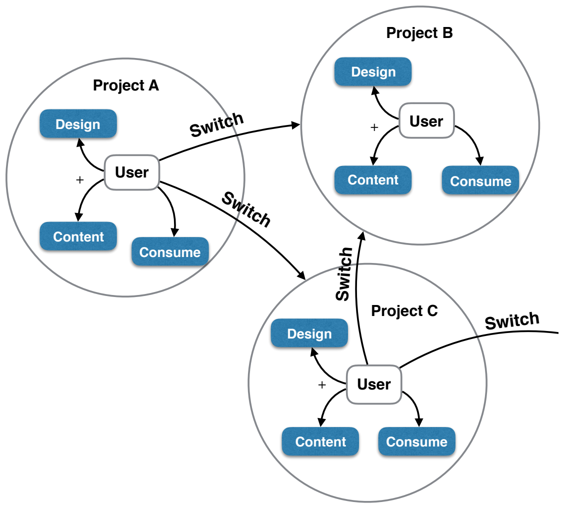

Figure 1: An illustration of the *TILLT* domain. Users can have different interactions with different projects. The problem is to identify relevant projects for each interaction type

of the *ISLET* problem by Amir et al. (2016), which we call the *TILLT* problem: **T**ype-specific **I**nformation Sharing in **L**oosely-coupled **L**arge-scale **T**eamwork. The problem domain is illustrated in Figure 1. We identify two challenges within the *TILLT* domain that we address in this paper. The first is how to model the future behavior of an individual user based on past behavior, considering the types of interactions and teams the user was involved in. The second challenge is to determine how to use this individual behavior model in a global model that uses knowledge about all teams to identify information to share with a user.

To address the first challenge, we propose two techniques based on Hidden Markov Models (HMMs) and on Long-Short Term Memory Neural Network (LSTM)-based autoencoders to learn a condensed representation of sequential user interactions. For the second challenge, we present a graph-based representation of interactions between users and teams, teams and other teams, and users and other users. We use the encoded user interactions as well as information about the interaction graph to build a discriminative model to predict both a user's likely future interactions with other users and teams. Our models are able to discern a real future interaction from 500 other possible interactions over 65% of the time with increases in performance to over 80% when only predicting active interactions. In a second analysis, we show that we can predict whether teams are within the top half for project success for over 85% of the repositories based on solely the representation of their sequential interactions.

In Section 2, we formalize the *TILLT* problem. In Section 3, we discuss the interaction graph representation of the problem and different measures we can extract from the graph. We define metrics for user-team interactions, general information about a team, and introduce user-representations. We then describe experiments in Section 4, our data in Section 5, and discuss the results in Section 6. Finally, we contrast our approach with related work in Section 7.

2. Large Scale Collaboration

Highly distributed collaborative platforms such as GitHub comprise many repositories that represent individual teams. Following the definition for the *ISLET* problem, we begin to formalize the *TILLT* problem with a set of Users U and a set of teams R. A study by Tausczik et al. (2014) showed that on LCPs, most interactions can be classified into a small set of interaction types. Following this finding, we classify user interactions into three distinct types. A user can contribute to any of the teams by actively adding content, by discussing the design, and by passively consuming the content. As such, we consider the set of contribution types $A = \{ADD, DES, PAS\}$.

Inter-user collaboration on LCPs often happens asynchronously. For example, consider two users Alice and Bob who both work on a project Foo. When Alice makes a change, Bob sees it and contributes to Foo accordingly, yet hours or days might have passed, and Alice may have moved on to different contributions in that time. To account for this asynchronicity, we define a set of time frames T, each of length Δ. Actions by different users that happen within the same frame $t \in T$ are considered to fall into the same interaction session. A single session s is thus defined as a tuple $(t, (\langle u_1, r_1, a_1\rangle, \ldots, \langle u_{|s|}, r_{|s|}, a_{|s|}\rangle))$, where $\langle u_i, r_i, a_i\rangle$ describes an action $a_i \in A$ taken by user $u_i \in U$

working with team $r_i \in R$. The set of all interaction sessions is called S.

We consider two problems within the *TILLT* framework: identifying information to share, and identifying successful teamwork. The goal of the *ISLET* problem is to determine which objects are relevant to each user within a team. Correspondingly, in the *TILLT* domain the equivalent goal is to determine which teams are relevant to each user. Since *TILLT* distinguishes between three contribution types, the goal is further specified as predicting the teams and corresponding contribution types that are relevant to a user. To illustrate this goal, we consider the user Carl, whose behavior within his teams spans the three categories of interaction. He is a contributor to team Gro and actively develops and pushes code. He is also the project lead for team Hao, and shapes its design by commenting on and opening issues. Finally, he passively watches team Iot so that he is up to date with its development. Given this variety of interactions and teams, he requires nuanced information such as "which other projects should I follow?", "Were there recent changes in Hao?", and "Are there other projects that would benefit from my contributions?". The first *TILLT* problem is therefore to determine a set of relevant teams and interaction types about which to inform the user. More formally, the problem is to devise a model that predicts an interaction session s_t given the information from all previous sessions $s_{1:t-1}$. The predicted session would form the basis upon which an information sharing system might inform the user of relevant information.

In the second problem, we aim to use the history of interactions that users had with a project to characterize its likelihood of success in the future. Formally, we utilize the sequence of sessions $s^r_{1:t-1}$ for a given project r to predict some success metric m^r_t of the project in time t. Characterizing success based on user interactions enables recommendations as to whether a project requires more interactions of a particular type a to increase its chances of success. Combining the results from both problems can yield a powerful system that can guide projects on a global level as well as provide helpful information to individuals.

3. Graph-based Representation of Teamwork

Amir et al. (2016) propose mutual influence potential networks (*MIP*-nets) to address the *ISLET* problem. A *MIP*-net is a graph structure that consists of nodes for users and for objects, as well as edges between users and objects and objects and other objects. This formulation assumes that every user in the problem is a member of the same team. In the *TILLT* problem, this assumption is not valid, since users can work on completely disconnected teams. Moreover, previous work by Tymchuk et al. (2014) shows that on LCPs, people often do not know others even in their immediate neighborhood within a collaboration graph. Therefore, we introduce edges between users to measure how tightly coupled the contributors within the globally spanning team are. We additionally define the graph to be a multi-graph and represent each contribution type as a separate edge between a user and a team. Formally, we define the following elements of the graph:

- N_U: a set of nodes representing users.
- N_R: a set of nodes representing teams.
- E: a set of edges of one of three types: (1) $\langle n_u, n_r, a\rangle$, connecting a user u with a team r for a specific action type a, (2) $\langle n_{r1}, n_{r2}\rangle$, connecting a team r_1 with another team r_2, or (3) $\langle n_{u1}, n_{u2}\rangle$, connecting a user u_1 with another user u_2.

For each user-team interaction $\langle u, r, a\rangle$ within a session, the graph updates the weight of the edge $\langle n_u, n_r, a\rangle$ where n_u is the node of the user, n_r is the node of the team, and a the interaction type. Additionally, the edge $\langle n_u, n_v\rangle$ is updated for each user v that also interacted with r within the same session. Finally, the edge $\langle n_r, n_q\rangle$ is updated for every other repository q that the user interacted with during the same session. We add a weight decay parameter γ by which old edge weights are discounted before each update. This process is repeated for every session to construct the final graph. An example of how weights of edges within the graph can be used to infer information to share with users is shown in Figure 2.

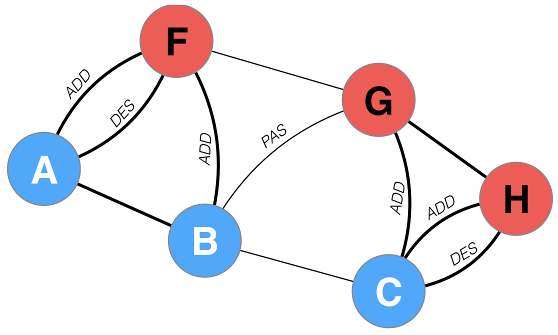

Figure 2: An example multigraph with three users and three teams. Users Alice (A) and Bob (B) both actively contribute to project Foo (F) and therefore have a high weight between them. Bob also watches project Gro (G). User Carl (C) is an active contributor to Gro and therefore, he and Bob have a weak weight between them. Due to Bobs interaction with both, Foo and Gro have a small weight as well. An information sharing algorithm can now recommend Gro to Alice as a potential new project to watch, as well as show relevant changes within Foo. Carl is also actively contributing to a second project Hao (H), which might in turn be interesting to Bob. Due to the multiple active interactions between Carl and Hao, the information sharing algorithm learns that Hao is more relevant to Carl than Gro and thus prioritizes information about Hao for him.

We extract several measures from the teamwork graph. Given a user u and a repository r, we compute a vector of concatenated features $\mathbf{x}$. Each individual feature is denoted by an index i. Following the approach of Amir et al. (2016), we compute the degree of interest (DOI) (Furnas, 1986), which uses team centrality and distance between the user and team as measures. Our proposed multigraph representation allows for additional computation of metrics for each user-team pair, individual team, and individual user.

3.1. User-Team Measures

The first metric $\mathbf{x}_1$ is the distance $D(n_u, n_r)$, which is a component of the DOI as presented by Amir et al. (2016). It is computed as the Adamic/Adar proximity metric (2003) between nodes n_u and n_r. To account for all interaction types A, we compute an additional distance $D(n_u, n_r, a)$ for every $a \in A$ that only takes into account edges of type a.

To measure how connected a user u is to other users within a team r, we define $\mathbf{x}_2$ as the connectedness

$$C(n_u, n_r) = \sum_{(n_u, n_v)\,:\, v \in U} \mathbf{1}(\mathrm{w}(n_v, n_r) > 0) \cdot \mathrm{w}(n_u, n_v)$$

where $\mathrm{w}(n_u, n_v)$ denotes the edge weight between n_u and n_v.

The last measure $\mathbf{x}_3$ is a binary value that indicates whether u is owner of r. While not every large-scale cooperative platform might have such a relationship, this owner relationship, if it exists, is a strong signal that a user will continue to contribute to a particular team.

3.2. Team Measures

We define $\mathbf{x}_4$ to be the a priori importance (API) of a node, which is the second part of the DOI computation. We follow Amir et al. (2016) by using the degree of a node as the API measure, computed as the fraction of total nodes to which n_r is connected.

We further compute two measures that together make up $\mathbf{x}_5$, a metric for how tightly coupled a team is. Within a loosely coupled team, members are more likely to switch to other projects than in tightly coupled teams. Therefore, we define the first part of $\mathbf{x}_5$ as the fraction of users in a team that have an edge of weight > 0 between them, and the second is the average weight between users in a team.

Other possible measures about a team that are not used in this general model, but that could easily be added for our domain-specific problem, include descriptive measures such as the number of stars or forks that a repository on Github received.

3.3. Encoding User Behavior

A first approach to modeling user behavior is to count the number of interactions of each type that a user had in the past. We encode this information as $\mathbf{x}_6$. However, this approach does not incorporate the sequential nature of interactions. If Alice switches back and forth between the projects Foo and Bar, while Bob stays at Foo, and then switches to Bar, it is conceivable that their total interaction counts would be the same, despite their true relationships with the projects being significantly different. Hence, we consider two approaches to encode temporal information about user behavior to comprise measure $\mathbf{x}_7$, a Hidden Markov Model (HMM), and an LSTM-based autoencoder.

3.3.1. Hidden Markov Model

Formally, a HMM is a 4-tuple, (S, O, π, Ω), where S is a set of states, O is a set of observations, $\pi : S \times S \rightarrow [0, 1]$ is the probability of transitioning from one state s to another s', and $\Omega : S \times O \rightarrow [0, 1]$ is the probability of emitting observation o while in state s. As such, an HMM models a generative process that emits sequences of observations. We propose that each user transitions between states that describe the unobserved characteristics of their work, and that we observe emissions generated from these states as the actions they take. Specifically, we observe a sequence of actions taken by users over time. For each observed action, we additionally note whether an emission o_t corresponded to the same project that the last emission o_{t-1} did. This models both the types of contributions a user makes to their various projects as well as the user's propensity to switch between projects. Given sequences of observed states, the HMM learns a distribution over the hidden state sequences, in essence describing the most likely workflow for each user. Using the observed emissions, we use the Baum-Welch algorithm to learn the parameters S, O, π, and Ω. From there, we use the Viterbi algorithm to estimate the most likely sequence of hidden states for each user. In this case, $\mathbf{x}_7$ is defined as the fraction of steps that a user spends in each state.

3.3.2. Autoencoder

Due to the Markovian assumption in an HMM, the probability of an emission is independent of a user's history of actions. To generate a representation of the whole sequence of events, we propose using an autoencoder based on an LSTM architecture (Hochreiter and Schmidhuber, 1997). An LSTM is a recurrent neural network (RNN) that learns to map a sequence of inputs of length T, $o_1, \ldots, o_T$, to a sequence of representations $\mathbf{h}_1, \ldots, \mathbf{h}_T$ by learning and recursively applying a function **RNN** at each time step $t \in 1 \ldots T$:

$$\mathbf{h}_t = \mathbf{RNN}(o_t, \mathbf{h}_{t-1})$$

In an autoencoder, two RNNs are chained together, the encoder and decoder. For each time step t, the encoder computes $\mathbf{h}_t$. The final representation $\mathbf{h}_T$ contains an encoding of the whole sequence of observations.

The decoder uses the encoding $\mathbf{h}_T$ from the encoder to initialize its $\mathbf{h}_1$. From there, it makes a prediction $\hat{o}_1$ that aims to restore the true o_1, and recursively uses $\hat{o}_1$ as an input to compute $\mathbf{h}_2$ as well as to predict $\hat{o}_2$. This process repeats T times to compute the sequence $\hat{o}_1 \ldots \hat{o}_T$. In other words, the encoded vector $\mathbf{h}_T$ is a lossy compression of the whole original sequence, which we use as our feature $\mathbf{x}_7$.

4. Experiments

4.1. Predicting Users' Actions

Let $\mathbf{X}$ be a design matrix composed of the $\mathbf{x}$-type rows that describe the relationship between each user u and each team r given the information in $s_{1:t-1}$ at a time t, as defined in Section 3. Let $\mathbf{y}_{all}$ be 1 if there is any interaction between u and r in session s_t, and 0 otherwise. Analogously, for each $a \in A$, let $\mathbf{y}_a \in \{0, 1\}$ be 1 if and only if an interaction of type a exists at time t.

We model this problem by learning a discriminative probabilistic model $p(y|\mathbf{x}; \Theta)$, and learn the parameters Θ. Unlike in previous work (Amir et al., 2016), it is not possible to hand-pick Θ, since the number of features in $\mathbf{x}$ is so large. We utilize both a linear and a nonlinear model to compute $p(y|\mathbf{x})$, namely a logistic regression and a neural network.

In total, we consider four different models: (1) Logistic regression using the full feature set $\mathbf{x}$, (2) Neural network using $\mathbf{x}$, (3) MIP-DOI by Amir et al. (2016), which is equivalent to a logistic regression using only $\mathbf{x}_1$ (Distance) for all interactions and $\mathbf{x}_4$ (API), and (4) a neural network using all measures except those in MIP-DOI. The reason behind choosing (4) is to investigate whether it is possible to predict interactions without a distance metric between a user and a repository. This model has to rely solely on indirect features between a n_u and the users of the repositories that n_u is connected to.

Table 1: Description of different events on GitHub.

Name	A	Description
ReleaseEvent	none	Release is published.
PublicEvent	DES	Private repository is made public
PullRequestReviewCommentEvent	DES	Comment on the diff of a pull request
ForkEvent	PAS	Fork of a repository
GollumEvent	DES	Wiki page is created or updated
MemberEvent	none	User is added as collaborator
DeleteEvent	ADD	Deleted branch or tag
IssueCommentEvent	DES	Issue comment is created/edited/deleted
PushEvent	ADD	Push to a branch
PullRequestEvent	ADD	Pull request is opened/updated/closed
CommitCommentEvent	DES	Commit comment
WatchEvent	PAS	Someone starred a repository (not watch!)
IssuesEvent	DES	Issue is created/updated/closed
CreateEvent	ADD	New Branch or Repository

We train the models to predict each of $\mathbf{y}_{\{all, ADD, DES, PAS\}}$ and measure the following success metrics: (1) Average rank of the correct answer in the set of predictions made by the model (μ_{Rank}), (2) precision of making one prediction, and (3) recall when making 5, 10, and 20 predictions.

4.2. Predicting Successful Teamwork

In the GitHub domain, two obvious choices for success metrics are the number of stars and the number of forks that a repository has. Starring a repository indicates interest in and usefulness of a repository, while forking indicates an active desire to contribute to and improve the repository. In this model, we construct our feature sets similarly to the user-project features, but instead of considering sequences of actions for each user-project pair, we use the full sequence of actions related to each repository. That is, we train our HMM and RNN models on the sequences of actions taken with respect to each repository regardless of which user actually took the action.

Let $\mathbf{x} = \mathbf{x}_7$ for either the HMM or RNN model for the project-centric, user-agnostic training sets. Let $\mathbf{y}_{fs}$ be the sum of the number of forks and stars for each repository. Further, let $\mathbf{y}^b_{fs}$ be an indicator variable for whether a repository's combined number of stars and forks is in the top 50% of all values. We learn a discriminative model $p(y|\mathbf{x};\Theta)$ to predict the above labels given only the output of the HMM or RNN. This results in two modeling tasks: predicting the number of forks and stars for an new repository, and classifying a new repository into the top or bottom halves of the range using a particular $\mathbf{x}_7$. Due to the complex, noisy, and non-linear nature of the dataset, we experiment with a wide variety of predictive models. For both tasks, we report results for the HMM and RNN encodings and the three highest performing models we considered. For regression, these are (1) ridge regression, (2) SVM regression, and (3) random forest regression, and for classification, these are (1) logistic regression, (2) SVM classification, and (3) random forest classification.

We additionally conduct the same experiment while omitting actions from the input that correspond to a repository receiving a star or a fork. This prevents the models from learning to extrapolate future success from the current number of forks and stars that a repository has. The results are a direct indication of whether certain sequences of interactions are more likely to occur in successful repositories.

5. Data

We use a dataset comprising all $10,231,246$ public events that occurred during January 2015 on Github, downloaded from the GitHub Archive[1]. The data are formatted as a list of events and corresponding information such as a timestamp, a user, and a repository. A list of all possible events, a short description of each, as well as our assignment of each to an interaction type a can be found in Table 1.

Previous work found that most projects on GitHub are not collaborative (Lima et al., 2014). To capture only collaborative users and projects, we exclude repositories with fewer than 3 contributors or 40 interactions, as well as users who contributed to fewer than 3 repositories or had fewer than 40 interactions. This limits the data to $1,096,459$ interactions between $16,987$ users and $10,684$ repositories. The full interaction graph has $1,320,485$ edges. The minimum number of stars and forks for a repository is 0, and the maximum is $9,459$. We set Δ to 3 days, which results in a total of $T = 10$ sessions.

5.1. Details on Model Training

For our first experiment, we use training data $s_{1:T-2}$ to predict actions in s_{T-1}. We then update the graph and the encodings with the true data from session s_{T-1}, ultimately testing the trained parameters on the previously unseen interactions in s_T. While this means that past data from the training set is part of the input to the test, this most accurately follows what a real application would do. An alternative strategy, not followed in this paper, would be to test the parameters on a completely separate dataset (e.g. a different month).

When training the model to identify future interactions, we follow a negative sampling strategy to address the quadratic complexity of possible user-repository interactions. For every future interaction, we sample 12 incorrect repositories and use these to train the algorithm to distinguish between them and the correct repository. When testing the algorithm, we compute the likelihood for the correct case as well as 500 additional repositories for $1,000$ users (the sampling numbers are chosen to balance compute time and accuracy of the algorithm). We additionally normalize every feature in $\mathbf{X}$ to z-scores to be able to compare the magnitude of different Θ_i.

Table 2: Results of the information sharing task. Precision and recall are percentages. For the average rank, a lower number is better. For all other measures, higher is better.

a = all	μ_{Rank}	Pre@1	Rec@5	Rec@20
LR	56.25	58.02	53.82	65.70
NN	**49.39**	**65.89**	**57.68**	**70.09**
MIP-DOI	52.41	56.02	54.19	67.63
No DOI	51.54	57.69	55.09	68.75
a = ADD	μ_{Rank}	Pre@1	Rec@5	Rec@20
LR	14.95	76.84	89.25	**93.86**
NN	**14.24**	**83.38**	**91.17**	**93.86**
MIP-DOI	21.40	70.30	87.52	91.75
No DOI	24.54	53.41	76.20	88.48
a = DES	μ_{Rank}	Pre@1	Rec@5	Rec@20
LR	24.76	73.36	83.62	88.94
NN	**23.54**	**81.07**	**86.64**	**89.51**
MIP-DOI	39.16	71.26	82.61	87.36
No DOI	24.90	67.05	81.90	88.22
a = PAS	μ_{Rank}	Pre@1	Rec@5	Rec@20
LR	117.27	10.28	13.04	28.80
NN	**89.34**	12.62	**17.75**	**37.5**
MIP-DOI	107.14	**13.08**	15.58	28.62
No DOI	91.34	**13.08**	15.22	36.05

The results shown for the NN model are for a neural network architecture that uses a two-layer feed-forward neural network with a tanh activation function and 25 hidden states. As user encoding, we use the autoencoder with an encoding size of 5, and $\gamma = 0.9$. These parameters were selected for their performance on a validation set.

For the success prediction task, we use 70% of repositories to train the models, and 10% to optimize hyperparameters. The remaining 20% of the data comprises the held-out test set for which we report results. For the HMM model, we compute

[1]`https://www.githubarchive.org/`

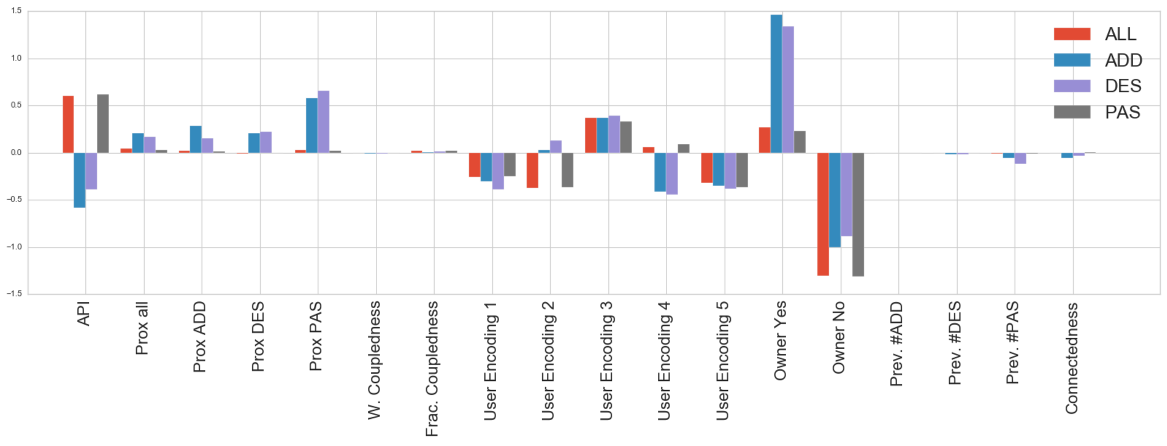

Figure 3: The trained coefficients of the logistic regression for predicting each of the interaction types.

results for models with 3-50 hidden states, finding that a model with 5 hidden states gives optimal results, which we report.

6. Results

6.1. Predicting Users' Actions

The results of the information sharing tasks are shown in Table 2. Our proposed non-linear neural architecture proves to perform best for most of the tasks, showing an improvement in precision of over 8% compared to the logistic regression in the task to predict all interactions. The results also show that the MIP-DOI algorithm maintains competitive performance despite its limited number of features. Our approach without metrics from MIP-DOI shows strong performance as well. This indicates that our proposed metrics are as informative as those from MIP-DOI, with the combination of them working best.

The results also show a clear divide between the performances of predicting active and passive interactions. Intuitively, that means that while it is comparatively easy to identify that Alice will continue to contribute to Foo, it is hard to identify a new project that Alice will likely watch. To further investigate this divide, we show the parameters of the logistic regression for each task in Figure 3. Due to the normalization of input features to z-scores, we can directly interpret and compare features with each other. We observe that the strongest

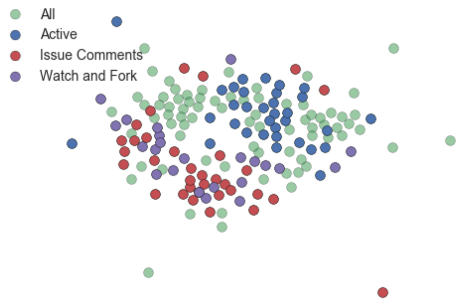

Figure 4: A two-dimensional t-SNE representation of learned user embeddings for a sample of users.

signal for a prediction in the ADD and DES tasks is whether a user owns a repository. Additionally, some of the signals are specific to active or passive contributions, namely the API, the proximity, and dimensions 2 and 4 within the user encoding. The relevance of the API is explained by the fact that more central repositories tend to be better known and thus receive more passive attention. This corroborates results that show that popularity plays a central role within a network when users decide to follow other users Krafft et al. (2016). Further, our results confirm those of Tymchuk et al. (2014), who found that users are generally unaware of other specific users in their projects. In the resulting parameters of our model, neither the connectedness, nor the measure of how tightly coupled a team is, has any impact on future interactions.

Another interesting result shown in the figure is that the sequential user embeddings have a major impact on a prediction, even more than the proximity measure. This is especially interesting because the raw counts of previous interactions a user had, which are a non-sequential embedding, do not have the same predictive power. This confirms our hypothesis that non-sequential models fail to capture as much information as is available. To further investigate the learned user-embeddings, we analyze them by projecting them into two-dimensional space using t-SNE (Maaten and Hinton, 2008). Figure 4 shows the projects for a sample of users. The users represented in blue had actions of type ADD over 75% of the time. The users shown in red were particularly active in opening and commenting on issues (over 20% of the time). The users shown in purple had actions of type PAS over 40% of the time. As shown in the figure, similar users tend to have similar representations. Heuristically, there is a clear divide between active contributors and passive/designing users.

6.2. *Predicting Successful Teamwork*

The results for the teamwork success prediction task can be found in Table 3. Our primary finding is that the autoencoded interactions for each repository have significantly more predictive power than the HMM average state vectors. Using the LSTM states as features and a support vector machine to regress, our median prediction error of the number of stars and forks for a repository is 5.7. Taking into consideration that the range of the labels was nearly $10,000$, this is impressive accuracy. On the classification task, the support vector machine classifier is able to determine whether a repository is in the higher or lower half of success at a statistically significant ($p < .001$) rate. This lends credence to the idea that the encoding learned is informationally rich and as such is able to differentiate successful repositories from unsuccessful ones.

We additionally observe that the HMM feature set has far less predictive power, though models based on the HMM features still retained a modest ability to discriminate between successful and unsuccessful repositories. This finding confirms

Table 3: Results of the teamwork success prediction task. For regression, we report median absolute error; lower is better. For classification, we report the accuracy rate; higher is better.

Regression	HMM	RNN
LinReg	23.0	12.3
SVR	22.9	**5.7**
RFR	22.8	10.0
Classification	**HMM**	**RNN**
LogReg	0.62	0.83
SVC	0.61	**0.85**
RFC	0.61	0.84

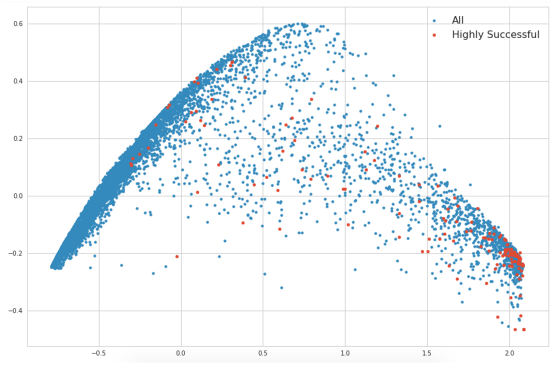

Figure 5: A two-dimensional PCA representation of learned repository embeddings, showing the clustering of highly successful repositories

our hypothesis that the HMM throws away information because it is unable to capture relationships between actions in the sequence that are more than one time step apart. In contrast, the RNN takes into account the entire sequence of actions, allowing it to infer unobserved behavior more correctly, thus lending it more predictive accuracy.

To further investigate the model's success, we perform a dimensionality reduction of the learned encodings via principal component analysis to two dimensions. Figure 5 illustrates this reduction, showing that repositories with stars and forks above two standard deviations from the mean are tightly clustered in reduced embedding space.

Finally, when we run the experiments while withholding information about past star and fork actions and using SVMs for both regression and classification, the median regression error increases to 10.8, while the classification accuracy drops to

0.74. Although the predictive power of the model degrades slightly, the learned encoding is still able to perform at a statistically significant level, despite omitting any direct information about success in the repository's history. Moreover, any real system will be able to include the number of stars and forks that a repository has received in the past in order to predict its future success. In essence, this secondary experiment shows that the RNN model captures nonlinear and indirect relationships between workflows and repository success, learning a useful encoding from high-dimensional and sequential data in a robust manner.

7. Related Work

Elliott (2016) considers many of the same challenges that we do, providing a model through which cooperation emerges in loosely-coupled teams of any scale. Describing stigmergic cooperation and a discrete set of possible interactions with a collaborative document, his work parallels our investigation into teams whose members are not necessarily co-located or aware of each other. Cress et al. (2013) specifically consider Wikipedia, studying users' preferences and their relation to knowledge creation, developing network models for contribution, and describing the shift in Internet culture that made large-scale collaboration possible. There have additionally been studies that investigate different graph representations for collaboration on LCPs (Biazzini and Baudry, 2014; Tymchuk et al., 2014). Most related to our work is a study by Thung et al. (2013) that defines a graph comprising users and repositories and computes metrics about individual teams from the information about which user contributed to which project. By computing page-rank statistics, the model identifies the most influential users and projects. However, this work does not consider the findings' application to the *TILLT* problem. Lima et al. (2014) also analyze GitHub as a social network, focusing on the follower dynamic. They treat the GitHub user base as a large directed graph in which an edge represents a user who follows another user. Their findings indicate that collaboration on GitHub is rare, since most projects only have one active contributor. Moreover, they find that there is no correlation between the activeness of a user and their number of followers, or between activeness of a repository and the number of stars it has. A similar study by Wu et al. (2014) corroborates this result using another large code repository (Homebrew), showing that collaboration does not lead to people following each other. The second part of our model is the computation of a single vector that describes sequential user-behavior.

A study by Kopeinik et al. (2017) used LDA to classify user behavior to enhance collaborative filtering. The idea of autoencoders to learn a representation has previously been used to study behavior of smart phone users (Rajashekar et al., 2016). However, our approach to augment an interaction graph with encoded user behavior to improve information sharing is novel, as is our application to predicting repository success.

8. Conclusion

In this paper, we have presented the *TILLT* problem, an extension of information sharing problems to a multi-team setting. We proposed a novel combination of sequential encoding of past user behavior and an interaction graph to identify relevant teams for different users and interaction types. Our results show that this approach can identify actual future interactions with a higher accuracy than comparative approaches. Additionally, we showed that an action-encoding on a per team basis can be applied to predict whether a team is successful.

In a future extension of this work, our model could be augmented with more information for both nodes and edges. Further testing could give insights into the amount of data necessary to make correct predictions without artificially restricting the data to active users and established repositories. Additionally, a test with real-time GitHub data could help to better understand the actual information that should be shared with individual users for each interaction type. The results could be implemented as a browser add-on that might help users to keep an overview of their current and future projects. It could also monitor repositories over time and guide users' contributions after predicting the types of interactions necessary to lead projects to success.

References

E. Hutchins, Cognition in the Wild, MIT press, 1995.

D. Pinelle, C. Gutwin, A groupware design framework for loosely coupled workgroups, in: ECSCW 2005, Springer, 65–82, 2005.

G. M. Olson, J. S. Olson, Distance matters, Human-computer interaction 15 (2) (2000) 139–178.

O. Amir, B. J. Grosz, K. Z. Gajos, MIP-nets: enabling information sharing in loosely-coupled teamwork, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI Press, 4192–4193, 2016.

N. Bălău, S. Utz, Information sharing as strategic behaviour: the role of information display, social motivation and time pressure, Behaviour & Information Technology (2016) 1–17.

Y. Xiao, H. Zhang, T. Basadur, Information Sharing Always Helps Team Decisions? The Hidden Profile Condition, in: Lets Get Engaged! Crossing the Threshold of Marketings Engagement Era, Springer, 411–411, 2016.

Y. R. Tausczik, A. Kittur, R. E. Kraut, Collaborative problem solving: A study of mathoverflow, in: Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, ACM, 355–367, 2014.

Y. Tymchuk, A. Mocci, M. Lanza, Collaboration in open-source projects: Myth or reality?, in: Proceedings of the 11th working conference on mining software repositories, ACM, 304–307, 2014.

G. W. Furnas, Generalized fisheye views, vol. 17, ACM, 1986.

L. A. Adamic, E. Adar, Friends and neighbors on the web, Social networks 25 (3) (2003) 211–230.

S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

A. Lima, L. Rossi, M. Musolesi, Coding together at scale: Github as a collaborative social network, arXiv preprint arXiv:1407.2535 .

P. M. Krafft, J. Zheng, W. Pan, N. Della Penna, Y. Altshuler, E. Shmueli, J. B. Tenenbaum, A. Pentland, Human collective intelligence as distributed Bayesian inference, arXiv preprint arXiv:1608.01987 .

L. v. d. Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (Nov) (2008) 2579–2605.

M. Elliott, Stigmergic Collaboration: A Framework for Understanding and Designing Mass Collaboration, Springer International Publishing, Cham, ISBN 978-3-319-13536-6, 65–84, 2016.

U. Cress, B. Barron, I. Halatchliyski, A. Oeberst, A. Forte, M. Resnick, A. Collins, Mass collaborationan emerging field for CSCL research, To see the world and a grain of sand: Learning across levels of space, time and scale: CSCL 2013 Proceedings 1 (2013) 557–563.

M. Biazzini, B. Baudry, May the fork be with you: novel metrics to analyze collaboration on GitHub, in: Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics, ACM, 37–43, 2014.

F. Thung, T. F. Bissyande, D. Lo, L. Jiang, Network structure of social coding in github, in: Software maintenance and reengineering (csmr), 2013 17th european conference on, IEEE, 323–326, 2013.

Y. Wu, J. Kropczynski, P. C. Shih, J. M. Carroll, Exploring the ecosystem of software developers on GitHub and other platforms, in: Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing, ACM, 265–268, 2014.

S. Kopeinik, D. Kowald, I. Hasani-Mavriqi, E. Lex, et al., Improving Collaborative Filtering Using a Cognitive Model of Human Category Learning, The Journal of Web Science 2 (4) (2017) 45–61.

D. Rajashekar, A. N. Zincir-Heywood, M. I. Heywood, Smart Phone User Behaviour Characterization Based on Autoencoders and Self Organizing Maps, in: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE, 319–326, 2016.

D. M. Romero, D. Huttenlocher, J. Kleinberg, Coordination and Efficiency in Decentralized Collaboration, arXiv preprint arXiv:1503.07431 .

U. Cress, J. Kimmerle, A systemic and cognitive view on collaborative knowledge building with wikis, International Journal of Computer-Supported Collaborative Learning 3 (2) (2008) 105, ISSN 1556-1615, doi:\bibinfo{doi}{10.1007/s11412-007-9035-z}, URL `http://dx.doi.org/10.1007/s11412-007-9035-z`.

M. A. Elliott, Stigmergic collaboration: A theoretical framework for mass collaboration, Ph.D. thesis, 2007.

F. Van Ham, A. Perer, Search, show context, expand on demand: supporting large graph exploration with degree-of-interest, IEEE Transactions on Visualization and Computer Graphics 15 (6).

O. Amir, B. J. Grosz, K. Z. Gajos, S. M. Swenson, L. M. Sanders, From care plans to care coordination: Opportunities for computer support of teamwork in complex healthcare, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ACM, 1419–1428, 2015.

N. E. Friedkin, A. V. Proskurnikov, R. Tempo, S. E. Parsegov, Network science on belief system dynamics under logic constraints, Science 354 (6310) (2016) 321–326.

A. Rzhetsky, J. G. Foster, I. T. Foster, J. A. Evans, Choosing experiments to accelerate collective discovery, Proceedings of the National Academy of Sciences 112 (47) (2015) 14569–14574.

A. Frieder, R. Lin, S. Kraus, Agent-human coordination with communication costs under uncertainty, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3, International Foundation for Autonomous Agents and Multiagent Systems, 1281–1282, 2012.