

---

# Analyzing and Visualizing Collaboration between Software Developers on Slack

---

UNDERGRADUATE THESIS

*Submitted in partial fulfillment of the requirements of  
BITS F421T Thesis*

*By*

Rohit KAUSHIK  
ID No. 2015A7PS0115G

*Under the supervision of:*

Dr. Thomas FRITZ  
&  
Dr. Neena GOVEAS

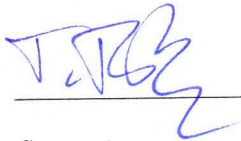


BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

December 2018

## Certificate

This is to certify that the thesis entitled, "*Analyzing and Visualizing Collaboration between Software Developers on Slack*" and submitted by Rohit KAUSHIK ID No. 2015A7PS0115G in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.



*Supervisor*

Dr. Thomas FRITZ

Faculty,

University of Zurich

Date:

*Co-Supervisor*

Dr. Neena GOVEAS

Asst. Professor,

BITS-Pilani Goa Campus

Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, GOA CAMPUS

## *Abstract*

Bachelor of Engineering (Hons.)

### **Analyzing and Visualizing Collaboration between Software Developers on Slack**

by Rohit KAUSHIK

Communication among software development team members is important for the project's success. With the growing size of software teams and its spread across different geographical location, it is even more important to use communication software. Team collaboration software is becoming popular in software development teams to manage projects, discuss ideas and ask questions. The communication over these collaboration softwares inadvertently affects the work flow and productivity of development teams. An ineffective or incongruent communication can often lead to delays and development failures.

In this thesis, we analyze slack conversation logs of open source projects to understand topics that software developers discuss over slack. We perform social network analysis on the collaboration network to understand how the communication patterns in terms of centrality, connectivity, and topics change over time. We try to develop accurate measures and visualizations that can describe the communication patterns, what affects the communication and how it changes over time.

The insight gained from the study can help us understand how the interaction between users change over time and the dependence of development teams on knowledgeable users. This can be used to study the behaviour during specific time periods like release periods, meetings or scheduled checkpoints.

# *Acknowledgements*

I hereby would like to thank Prof. Dr. Thomas Fritz for giving me this opportunity to work on the project and for his constant guidance and feedback throughout the project. I would also like to thank Prof. Ingo Scholtes for his valuable inputs and insights.

# Contents

<b>Certificate</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>4</b>
2.1 Team Communication in Software Development teams . . . . .	4
2.2 Communication Analysis . . . . .	4
2.3 Social Network Analysis . . . . .	5
<b>3 Approach</b>	<b>7</b>
3.1 Chat Analysis . . . . .	7
3.1.1 Extracting Topics and Keywords - Textrank . . . . .	8
3.1.2 Chat Disentanglement . . . . .	9
3.1.3 Using Keywords in understanding the communication . . . . .	10
3.1.4 Link Identification . . . . .	11
3.2 Network Analysis . . . . .	12
3.2.1 From Chat logs to Collaboration Network . . . . .	12
3.2.2 Time Window Selection . . . . .	12
3.2.3 Developing Good Statistical Metrics . . . . .	13
3.2.4 Visualization . . . . .	14
3.2.5 Summarizing Results . . . . .	15
<b>4 Dataset &amp; Validation</b>	<b>16</b>
4.1 Dataset . . . . .	16
4.1.1 Selecting Slack Channels . . . . .	16
4.1.2 Manually annotating dataset . . . . .	16
4.1.3 Limitation . . . . .	17
4.2 Validation . . . . .	17

---

<b>5</b>	<b>Architecture &amp; Design</b>	<b>19</b>
5.1	Authorization . . . . .	20
5.2	DataCollection: Slack Tracker . . . . .	20
5.3	Server: Analysis . . . . .	21
	5.3.1 Network Analysis . . . . .	21
5.4	Database . . . . .	22
5.5	Client: Visualization . . . . .	22
<b>6</b>	<b>Conclusion</b>	<b>24</b>
6.1	Limitations . . . . .	24
	6.1.1 Problems with TextRank . . . . .	24
6.2	Future Work . . . . .	25
6.3	Source Code . . . . .	25
	 <b>Bibliography</b>	 <b>26</b>

# List of Figures

2.1	Example of how node can affect interactions . . . . .	6
3.1	A Filter Table with user interaction data . . . . .	11
3.2	Mean of Degree Distribution . . . . .	14
3.3	Variance . . . . .	14
3.4	Degree statistics for collaboration network of slack channel . . . . .	14
3.5	Statistics for Collaboration graph of a slack channel . . . . .	15
5.1	The Workflow of Slack Analysis . . . . .	20
5.2	Screenshot of prototype . . . . .	23

# Chapter 1

## Introduction

Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

---

Conway 1968

Communication among software development team member is an important part of the development process and affects the project success. It can affect the workflow and the productivity of software team and ultimately the end project as suggested by Conway's law. Software team size has been growing, with teams located in different geographical location. The spread of software teams makes it, even more, to relay changes and information to team members. This has led to software teams adopting communication tools. Kauffeld et al in the paper [6] studies the emerging role in a software project and the satisfaction of member with current communication tools or approach. They classify the role of members in a meeting into clusters i.e complainer, solution seeker, problem analyst. They find that there was significantly lower satisfaction among member with current meeting approach. The following result suggests a problem with the communication approach.

Other works in studying team communication have focussed on understanding the roles and importance of users in the software team and establishing social-technical congruence between them. M. Schwind, A. Schenk, and M. Schneider[12] develop a tool for social network analysis in collaborative software development. They propose a method to calculate the productivity of the developer based on the contribution factor and quality metric in the collaboration network. Amrit and Hillegersberg [1] worked on mapping software team structure to their collaboration structure. Although they find similarity between team structure and collaboration structure, there were cases when developers working on the same module had no interaction. They classify such cases as socio-technical conflicts.



We chose to study slack communication because of its growing popularity among the software teams. Although some surveys of developers in a team using slack for communication suggest problems with slack, the past work has mostly focused on IRC channels and email threads. The paper by Siwei Fu et al. [5] presents a survey on slack usage and the problems faced by developers. The survey suggests that slack users have a difficulty navigating through chat histories and find it difficult to follow the conversation when multiple conversations take place simultaneously. Maja Saphir in this blog [10] describes how difficult it was for her software team to keep track of conversations and the difficulty in discussing the big-picture conversation. Since slack is among the most popular collaborative software used by development teams, it inadvertently affects the communication and hence the design of software. Researchers have been vastly studying the impact of slack in software teams. Some work tries and develops tools to help the user in easily understanding the communication in the slack team. Siwei Fu et al. [5] develops a calendar based visualization, with conversation threads and topics allowing the user to easily navigate the chat history and also search the chat in a more convenient way using tags such as user, keywords. Zhang and Cranshaw [13] develops a slack bot, TILDA, that allows users to easily understand the slack chat by collaborative tagging. The slack bot can tag chat messages into different categories i.e Answer, Question, Resource, Update and provide a detailed summary of the log. The survey conducted found that developers found it relatively easier to understand chat conversation which was collaboratively tagged by users and summarised by TILDA bot.

Most previous work has examined collaboration over email, meetings and to some extent, IRC and slack have been studied by past researchers. They have developed approaches to cluster participants into groups based on different factors such as information exchange, commits or task dependency. Metric such as response time and interaction frequency has been proposed to study the productivity of networks. Other works try to study socio-technical congruence. We believe that previous works are limited because they don't help slack users to identify the reasons for the observed pattern. Also, much of the work is focussed on static data and there is no tool to help them analyze the collaboration change over time. We believe it is also important to understand how these roles and interaction between team members change over time. A time dimensional study of the collaborative network can give insight into the interaction of team members in specific time frames particularly important ones like release period and scheduled deadlines and can also reveal issues that lead to these behavioral changes and its impacts on software development. In the thesis, we develop methods and visualizations to understand topics and conversations that take place in an open source software development teams that use slack. The goal is to understand the interaction between users and study the change over time. Understanding the interaction change can provide insight into the behavior of team members during specific time periods like release period, deadlines and also how the productivity of the team is affected due to these interaction changes.

Based on the problem described above we formulate our research problem in two parts:-

RQ1: How do software developers communicate over slack?

RQ1.1: What are the topics of discussion on a slack channel?

RQ1.2: Do developers interact only with a certain group of people?

RQ1.3: How does these interaction change over time? Are there any recurring patterns in the interaction?

RQ2: Can we develop an approach that accurately describes the interaction of developers and help us understand the effect of this interaction on software development?

To address the research problems we develop an approach for communication and network analysis. Communication analysis deals with the first research problem where we develop methods to cluster chat logs into conversations using a logistic regression classifier and describe these conversations using text-rank keyword extraction. We study the relation between users that can be used to describe the communication network and come up with two different links i.e thread link and direct link. The communication analysis and link identification is used in network analysis to generate network and develop metrics to describe the network. We develop a prototype for studying the variation in metric over time and try to address the change in pattern and how each user affects this change.

## Chapter 2

# Related Work

Our approach can be divided into two stage namely Communication analysis and Network analysis. Communication analysis helps us study our first research problem which is to understand the topic of conversation and interaction between users while Network analysis is aimed at the second problem of building a tool that can help us studying collaboration network and changes in collaboration patterns over the time. In this chapter, we related work by researchers in the field of communication and network analysis.

### 2.1 Team Communication in Software Development teams

Conway et al in his thesis [3] suggest that the design structures of the product are constrained by the communication structure. Since then Conway laws has been studied in various other field. Sabrina et al [2] present a literature review of Conway law and it's study in different fields. The literature review suggests a vast interest in studying Conway's law in field of software development teams.

### 2.2 Communication Analysis

With multiple conversations taking place simultaneously, the task of keeping track of conversation history is not trivial and can sometimes lead to ambiguity as well. Elsner and Charniak [4] proposes a supervised algorithm using logistic regression with input features based on user mentions, author and text of the message, to disentangle chat into conversation thread. Mehri et al [8] build up [4] work and comes up with a more extensive pipeline using recurrent neural network. The pipeline shows an improved performance on the IRC chat log used in the study of Elsner and Charniak. Some work have also focussed on presenting this conversation thread

to users helping them in easily navigating through the logs. TCAL [5] makes use of Elsner and Charniak work and builds visualizations for conversations as thread pulse integrated into a calendar view.

To better understand the usage of slack in software teams, some studies try to classify the topic of conversation into different categories. Most Researchers use text ranking to extract important keywords from the documents. Several algorithms use co-occurrence as a criterion for ranking words. Mihalcea and Tarau present a graph-based recursive algorithm that rank the words of a text. Other words try to incorporate the semantics and word2vec to improve. Zou et al [14] suggest an improved Text Rank algorithm using word2vec.

### **Change in discussion topics**

While knowing the conversation topics and threads can be helpful, It is also interesting to see if these topics recur over time periods or how discussion topics progress over time. One area where this can be useful is in understanding the sentiment of developers and correlating the variation with other network statistics that we try to evaluate.

## **2.3 Social Network Analysis**

Software development is collaborative effort and sub-groups in a team affects each other, a change in any group affects all the other subgroups of the team. This can be even worse when the team depends heavily on single developers or group. Figure 2.1 shows a simple example of removing the central node affects the knowledge sharing of the network. Such nodes are more central and facilitate knowledge sharing and the software teams are more dependent on these developers. Centrality has been used as a measure of functional or dysfunctional collaboration. Apart from topological changes, the changes in node degree and skewness can also reveal more central developers or developers through which most information flow happens. Roles and centrality of users changes over time. These changes may also affect the software development process and the interaction patterns among sub-groups. For example, during release periods most experienced developers will be involved in code merges and review which means the interaction between them would also be more frequent. Our goal is to develop an approach that can help users see the communication pattern over time. We want to develop good metrics that can help describe the collaboration network and the quality of collaboration. A highly central network with dependency on a single developer is not good for software team in the longer run. With a tool that helps the team understand collaboration over time and what factor lead to pattern change, it would be easier for teams to self-monitor and restructure their development and communication.

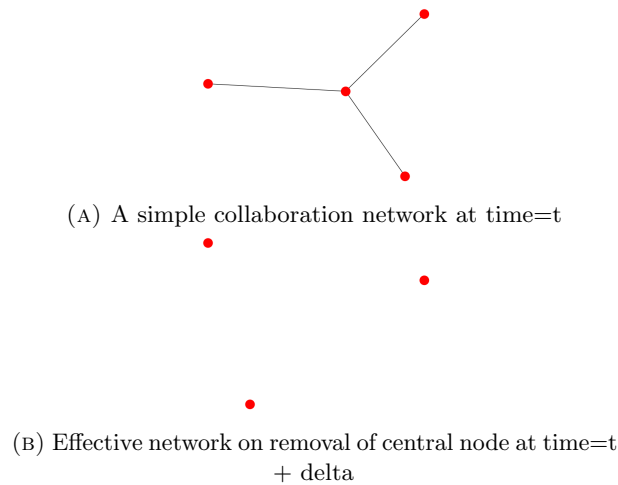


FIGURE 2.1: Example of how node can affect interactions

Graph-based analysis has been popular in various field including the social network. Graphs provide an abstraction from the raw data which can be used to study relations more easily. From the software collaboration point of view, researcher has been interested in studying the subgroups and their interaction which takes place in a collaboration team. Network derived from emails and recorded team meeting has been studied at depth. Michael et al in his paper [12] presents a tool to investigate collaboration network in software teams. The paper discusses different topological structure present in collaboration and how it differs from network in other fields. Mateli et al in [7] studies software team with members distributed globally where they find coordination among team members is a challenge because of time zone and geographical differences. They found sub-groups in the collaboration network often mapped to the location of members.

The work has been focussed on using social network analysis and identifying the structure of the collaboration network and trying to map it to subgroups in the team. Schwind et al. develops an analysis tool to identify clusters in the collaboration network and then later identify these cluster with the subgroups present in the team. Others have studied the network in terms of centralities of the users and try to relate social network Analysis of IRC channels, email threads and recorded team meetings has also been conducted in the past. Willenbrock et al [6] studies the emergence of developers roles in a software team. Recorded team meeting were coded into categories based on message, which later was clustered into five emerging role requirements. Sauer and Kauffed [11] studies the structure of social meetings and how functional and dysfunctional a meeting is based on distribution of central users.

## Chapter 3

# Approach

The current research in analysing communication only studies logs for a specific fixed time frames. They donot provide insight into how these communication pattern changes over time or during the specific phase of software project like release period, scheduled deadlines. Our aim is to develop an approach to understand the topics and interaction on the channel as well as to study how the communication change over time. The insight into pattern changes of the communication can help us understand how software progresses in different time frames and also be used for correlation with productivity or success metric of the project. We study the collaboration network and propose metric that can help us describe the communication. The metric choice is based on previous research proposal of centrality as a measure of functional or dysfunctional collaboration. We try to build a prototype for studying these metrics change for a rolling window.

This chapter present different analysis that we performed on the chat logs. We present how these different analyses can be helpful to a developer in self-monitoring. In the last section, we discuss the visualization and how both the analysis can be combined to form a holistic view of the communication patterns.

### 3.1 Chat Analysis

To understand the conversations on slack channels and present important and useful information that might help software team to monitor their slack communication retrospectively we perform analysis on the chat log text using natural language and machine learning method. Different approaches have been proposed for extracting information from texts but a combined analysis and visualization doesn't exist. [5] presents a visualization application aimed at helping users navigate through the slack logs easily. While our work also involves analysis and developing

visualization, the aim is not just limited to building a tool for helping users in navigation the logs but also understanding the patterns which can later be used in explaining the patterns detected in network analysis. In that sense, Chat Analysis is the precursor to network analysis which extracts information helpful in studying and explaining the collaboration network patterns.

### 3.1.1 Extracting Topics and Keywords - Textrank

There are various algorithms that can extract keywords from the text. The most predominantly used are RAKE, TextRank, TF-IDF. Both RAKE and TF-IDF relies on occurrence counts of words in the text. Mihalcea and Tarau proposed a graph-based recursive algorithm [9] which perform better than other algorithms like RAKE and TF-IDF. We use text rank algorithm, which is a graph based recursive method to rank nodes, to extract important topics from developers interaction.

TextRank works in three stages,

- **Preprocessing** - In this step, the text is first tokenized and stop words are removed. TextRank contains an extra filters based on word type. Based on context, words other than Noun and Adjectives are removed. In our study, we include noun, verbs, and adjectives. The tokenized text is tagged using a pre-trained part of speech tagger from the OpenNLP module. The remaining words are treated as candidate words.
- **Ranking** - The candidate words are used to form a graph with nodes as word and an edge between every word that appears in a window of fixed length. We considered a window of length 5 as was the case in the TextRank paper as well. This step assigns a score to every candidate word. Words with a higher score are more likely to be a better keyword.
- **Post Processing** - The post-processing step involves selecting top keywords.

Our TextRank algorithm is same as the one proposed in paper[9] with only modifications to the preprocessing step. The Preprocessing step is described in the following section in a greater detail.

#### **Preprocessing Stage**

Although our text rank algorithm is similar to the one proposed in the TextRank [9] paper, we do some extra preprocessing, for the chat logs like removing slack based smileys, and messages with length less than 3 and generic messages like greetings i.e Hello, Hi, Thank you are ignored since they do not come under keywords or give any information about the conversation topic. As a first step to preprocessing we first tokenize the text into words. Although the simpler way to do this is to split the text by space but to improve the textrank accuracy we use a pre-trained

text tokenizer from OpenNLP module. The tokenized words are filter based on, stop words or words that don't add any information to text and other based on word type. In the paper [9], the authors suggest considering only noun and adjective as candidate work and filtering out rest. For our research, we use both noun and adjective as well as verbs.

The preprocessing stage filters out the following noise from the text data:-

1. System messages
2. One word messages
3. Slack specific smileys
4. Parts of Speech tag other than noun, verb and adjective

The words remaining are filtering and preprocessing is called candidate words which form the text graph and are ranked for keywords.

### **Keyword as a feature for conversation clustering**

Previous research in field of conversation clustering, use similarity of the text in messages to classify if they belong to same thread or not. Although they deploy just stop word filtering and some similarity metric. In this work, we proposed to use similarity between the keywords extracted from messages as a metric for the classifier. Since text rank doesn't work on shorter message, we come up with a modified solution. To understand why simply using Text Rank on a single message, we should know that text rank is based on co-occurrence of text words. Hence words that co-occur more frequently are more likely to be a keyword. In smaller messages like that of slack chat it is difficult to capture co-occurrence and hence TextRank doesn't work well as expected. To Solve this issue, we use Text Rank with same preprocessing as described above to generate keywords for the entire dataset and then consider the common words in the message and keywords set as keywords for the particular message.

### **3.1.2 Chat Disentanglement**

The problem of chat disentanglement is non-trivial and it is sometimes difficult for humans as well. In the previous survey conducted by, the participants raise a coherent concern on difficulty to keep track of conversations. We first manually label chat logs from open source slack projects into clusters. We deploy the algorithm presented by Elsner and Charniak and build a logistic regression classifier based on know features but we use a new feature keyword similarity, with keywords extracted using Text Rank. This does improve the overall accuracy of prediction by 3-4%. The logs from Erlang and Kumunity Slack Workspace was used for training and validating



TABLE 3.1: Features Set for Chat Disentanglement

Feature	Explanation
timestamp_difference	difference between timestamp of two message
prev_mention_next	whether first message mention author of second message
next_mention_prev	whether second message mention author of first message
keyword_similarity	similarity of keywords extracted using text rank
same_author	whether both the messages have same author
same_mention	whether both message mention same users
is_question	whether the message contains question
is_answer	whether the message contains answers i.e yes, yep etc.
prev_is_long	whether first message is long ( $> 10$ )
next_is_long	whether second message is long

the classifier. The logs span across four months of activity and are quite entangled with many users talking at the same time.

### Identifying conversation topics

To solve our first research problem and understand the topics of conversation between users on slack, we run the TextRank algorithm on each of the threads identified (described in Section 3.1.3) in the chat logs. The preprocessing step and rank step is same as described in the previous section and in the post-processing step we select the top-n keywords where n depends on the length of the conversation. We considered n to one-third of total words in the conversation. The threads and keywords together provide us information of discussion topics and also to some extent user-interaction since we also store the user information that participated in the thread. With the information extracted using this approach we try and visualize it to provide users the information about topics, duration of the topic discussed and users who participated in the discussion. We believe this can be a helpful approach to help newcomers in the project. Although at the moment we don't have validation for keywords and it is a future planned work.

### 3.1.3 Using Keywords in understanding the communication

Our first research problem aims at studying the interaction patterns on slack channels. The goal is to understand groups and user interaction, the topics and the duration of interaction. Imagine two scenarios, one where a developer is away from work for some time and other where a new developer is hired for an ongoing project. It would be difficult in both the cases for developers to gather much insight about the project state or last discussions especially in scenarios where team are spread in different geographical location. To help solve this problem we try and combine text rank along with user interaction. The user interaction is determined by name mentions in the messages.

channel	to	from	duration	to
help-advanced	systemswiki	timh	1925	map access
help-advanced	rymohr	timh	1925	map access
help-advanced	timh	jeff	500	
help-advanced	timh	systemswiki	1230	map
help-advanced	systemswiki	timh	865	idea
slack-vs-kumu	mark-w4chl	jeff	1355	
help-advanced	alexvipond	watsonwilliamb	125	
help-advanced	watsonwilliamb	alexvipond	5535	hive kumu website co appropriate viz preview user image rawgraphs.
Count: 788				? Clear

FIGURE 3.1: A Filter Table with user interaction data

The visualization 3.1 was developed to help developers easily understand the interaction between the developers and possibly identify groups.

### 3.1.4 Link Identification

To perform any graph-based analysis we need to first define the graph nodes and relations between nodes. As our research focuses on the interaction between slack users, it is very easy to define the nodes as users but defining relation is not so trivial. While collaboration can be studied in different areas of development i.e through Github collaboration, commit review collaboration etc., given the slack data that we had, we came up with two relations that can be important in studying communication collaboration over slack. We define the two link as -

- Direct Links
- Thread Links

The direct links are calculated over entire slack log based on user mentioning. The slack follows a simple pattern, < UserName >, for allowing a user to mention another user. We process the entire log data and extract mention following the pattern. To calculate thread links we go through all the threads calculated from the Conversation classifier model and define a link between any two users if they interact or send a message in the same thread. Both the direct links and thread links are stored in the database along with the timestamps, which is later used during network analysis for generating the collaboration graph. We plan to extend this and add a validation for a link based on significance testing.

## 3.2 Network Analysis

Graph-based analysis has been in use wherever the relations in data can be represented in form of Graph. Graph provides good abstraction and representation of relation from raw data, which is much easier to study mathematically. In the context of studying social network or collaboration, graph based on task dependency and github commit contribution, geographical location and communication have been studied. Past work tries to study the quality of network based on response time and also problems associated with communication can be classified as socio-technical clashes. Other works have focussed on community detection and centrality.

To address our second research problem, our aim is to build a tool that can allow users to understand the communication patterns and time frames where the pattern changes from what is expected. Our focus is on determining these time frames with a sudden change in communication and then understand the reason for this change, which can be because of team-structure change, or approaching deadlines or software release. Keeping this in mind we build a prototype of a tool that allows a user to visualize metrics that can be used to describe communication pattern or collaboration network over time. The tool also allows understanding what nodes in the collaboration network adversely affects the metrics.

### 3.2.1 From Chat logs to Collaboration Network

We define a temporal collaboration network. Previous work do not study temporal network but [pathpy]. The links or relation in the collaboration network is defined as direct link and thread link. Both the links in a way suggest some information sharing between the users or developers. The accuracy of links depends on the thread-classifier accuracy and hence it is important to validate the links in the network. Although it wasn't possible to include the significance testing of links in the current work and is planned as a future extension.

To generate the collaboration network, we first calculate all the direct links and thread links and store it in a database along with the occurrence timestamp of the link. Then the identified links can be used to generate collaboration graph between any two timestamps.

### 3.2.2 Time Window Selection

We calculate the network metrics over a rolling window for different window size. To determine the maximum window size and then observe the metric in smaller frames, we compute the connectivity in the graph. Here connectivity is a parameter based to describe user participation. A x% connectivity would mean during the calculate window frame x% of total users on channel participated in the collaboration. Since in an open source project, anyone can join a slack

workspace and channel without restriction, we notice that it was difficult to gain a high percentage participation from user. So to have a maximum window size with good user participation we have a threshold of 50%.

The window size for two channels for 50% participation is described in the following table:-

TABLE 3.2: Window size and user participation

workspace	days
kumunity	31
erlang	35

### 3.2.3 Developing Good Statistical Metrics

In the last two section, we talked about generating collaboration network and selection of time frames to observe the collaboration. Now that we have both of this information, the challenge is to develop good metrics that can describe the communication and quality of the communication network. Observing these metrics values we should be able to understand the time frames in which there was a sudden change in interaction pattern. Nale et al. in his paper [6] have proposed centrality of the network as a measure of a functional or dysfunctional network. Think about a software team in which only 1 or 2 users are involved in most of the communication and development. Not only are these developer under constant pressure to meet their deadline but also help other developers with their queries. Such a project with a high dependency on few developers is likely to fail and is an example of dysfunctional communication. Also keeping in mind information exchange between different sub-group in a team is important, one way to also describe network is based on connectivity of the network.

Based on the aspects that we wish to explore for a collaborative network we come with up 4 different statistical metrics to describe the network. Since this is more of an exploratory work we chose simple metrics which are easier to understand and interpret.

The following metric was used in the prototype :-

1. Kurtosis
2. Skewness
3. Mean
4. Robustness

FIGURE 3.2: Mean of Degree Distribution

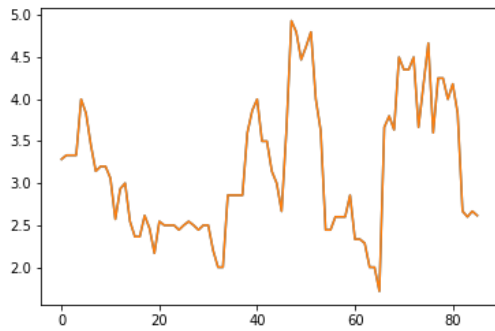


FIGURE 3.3: Variance

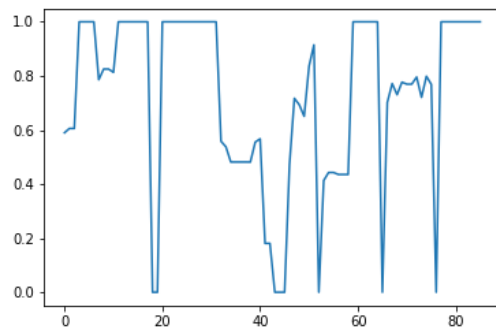
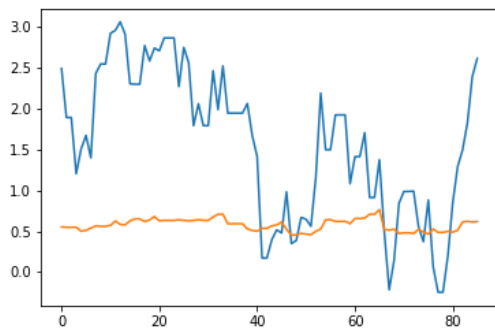
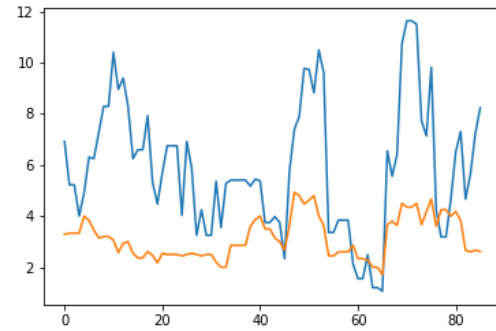


FIGURE 3.4: Degree statistics for collaboration network of slack channel

### 3.2.4 Visualization

The visualization for the network analysis was developed keeping in mind the research goal that is the tool should allow study of collaboration network over time-periods. The developed visualization has three components to it :-

- Metric chart, plot of proposed statistics for rolling window of different sizes.
- Slider, that allows user to focus on a particular time frame
- Graph, describing the collaboration network

To further address the problem of how each user/node in the network affects the communication patten we come up with a color scheme that assigns shade of green and red to each node in the collaboration network. The color intensity is based on effect of the node on a selected if the node is removed from the network. A dark shade of red would mean the network is affected adversely whereas green shade means the particular node doesn't affect the network statistics. For an example, if the selected metric is skewness, a dark shade of red would mean the particular

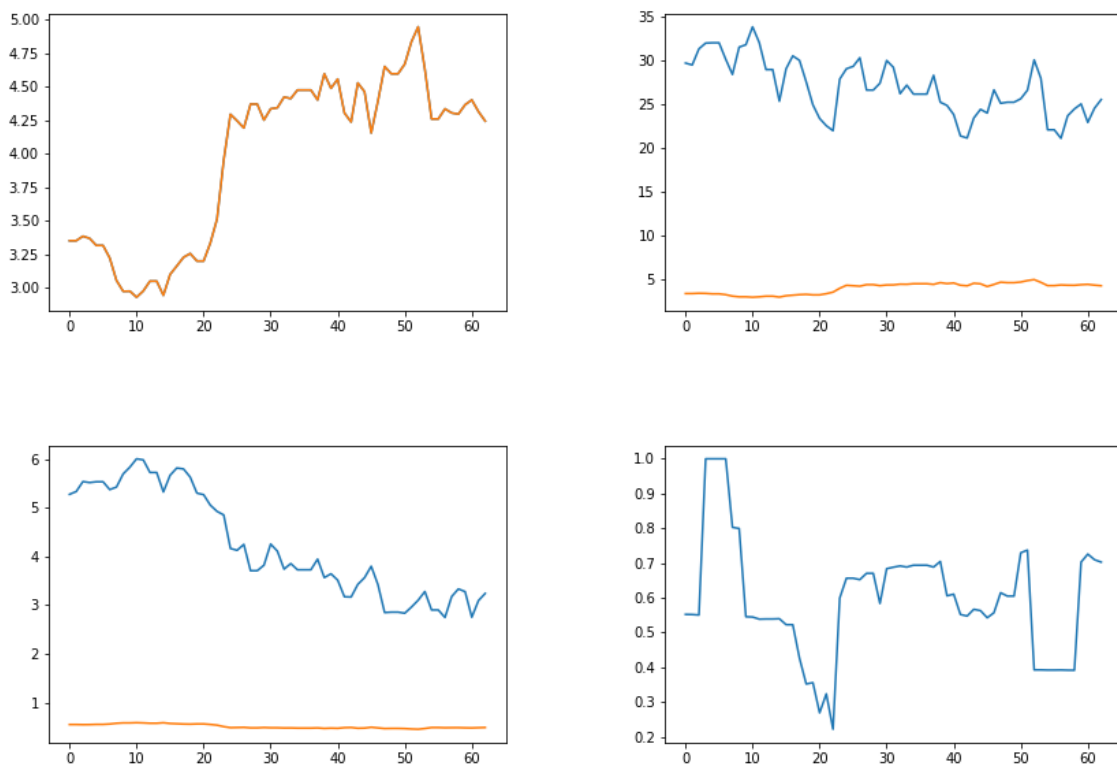


FIGURE 3.5: Statistics for Collaboration graph of a slack channel

node is more central to the conversation and if the user drops out of team, the communication pattern would change to greater extent. 5.2 shows a momentary screen shot of tool to study time varying collaboration network.

### 3.2.5 Summarizing Results

The plots for the selected metrics are shown in Figure 3.4 and 3.5. The observed pattern needs to be studied more and the explanation for the observed pattern can be derived from the network topology. At the moment we can draw two interesting conclusions from the observed pattern:

- Observed fluctuation in the pattern tells us that communication pattern changes in some particular time frames and studying the reason for these changes can be interesting. A possible reason that might cause a change can be approaching deadline or schedule release.
- The communication metric for some specific time frames is lower than the expected value than random Poisson distribution.

## Chapter 4

# Dataset & Validation

### 4.1 Dataset

#### 4.1.1 Selecting Slack Channels

To perform analysis described in Chapter 3 and validate the logistic classifier, we select software projects that use slack for communication. In this regard, we are limited to open source projects that allow public access to slack api token so that we can use the api for accessing channel history. Since the logs had to be manually labelled for threads and keywords, which is a time consuming process given that each workspace can contain more than thousand messages, we select the top two workspace with active participation from users. The workspace was also selected keeping in mind that it should have enough disentanglement so that it forms a good training dataset for the chat disentanglement classifier.

Based on the above described criteria of a good dataset and the limitation of time in manually annotating large dataset, we select two workspace Kumunity and Erlang for our analysis. Kumunity is a data visualization library where developers discuss the changes to library and also answer other users queries on how to use the library whereas on the Erlang channel, the discussion is related to developments of the language Erlang.

#### 4.1.2 Manually annotating dataset

The first task in validating the proposed analysis was to build the training dataset. We manually annotate the chat logs into threads and also assign keywords to each message. The annotation was done by two different person to understand if there are more conflicts than agreement on the thread classification. Based on the experience on annotators, it was difficult to classify the messages just based on the topic or context of message. The useful way to resolve such conflicts

was to see authors or mention in the message. The perceived important feature for classifications was keywords or context of message, author, mention in the message and timestamp difference in the order.

Overall annotators majorily agreed on the classification of messages into thread, with disagreement on only 5-10% of the messages. In some cases the cause of disagreement was whether to consider continuous conversation between same group of users but with different context or topics as same or separate threads. Other reason were in some cases it was genuinley difficult to identify the thread for message.

Below Table gives some statistics about the data and conversation thread detected on channels. The Kumunity workspace had more channels than mentioned in the table but there was almost no communication on these channels and hence they were excluded from the study.

TABLE 4.1: Kumunity Workspace

channel	no_message	threads	avg_thread_duration <i>inseconds</i>
general	326	224	218
help	670	280	933
help-advanced	124	73	483

TABLE 4.2: Erlang Workspace

channel	no_message	threads	avg_thread_duration <i>inseconds</i>
development	658	334	280
general	884	201	692
introductions	154	102	142

### 4.1.3 Limitation

The open source channels have limitations since it allows any user to join the channel and the conversation can at times be unrelated to software development. This is one of the limitations of the current work, since we don't have access to a private slack team the interaction of a more closed team with lesser members can be very different from that of an open source project.

## 4.2 Validation

For validating the chat disentanglement classifier we used 2 approaches, mainly splitting the data into train and test set for a single channel and also cross validating it across the channel i.e using data from kumunity channel as training set and data from erlang as test set or vice versa.



TABLE 4.3: Classifier accuracy

Dataset	Accuracy
Kumunity	81%
Erlang	70%
Kumunity as train set and Erlang as validation	75%
Erlang as train set and Kumunity as validation	64%

As can be seen by the result training on Kumunity data yields better results which is due to the fact that there were more disentangled conversation on Kumunity whereas large part of Erlang communication was between 2 users.

## Chapter 5

# Architecture & Design

The prototype at the moment spans in two different projects. We started with an aim of having a slack-tracker plugin integrated into Personal Analytics. Personal Analytics is an application developed by Andre Meyer, which collects user work and activity data and presents a retrospection for the same. Since our project is more focused on the team collaboration analysis, we are planning to transition the application into a separate flask-based python project.

The Design process tries to separate the analysis into three phases

1. Authorization
2. Analysis
3. Visualization

and similarly, the architecture of the project is divided into a Server-Client design. The Server stores all the chat logs and performs analysis parallelly. In the Authorization phase, the developer is requested to grant access to the application for accessing logs and other information like channel in the workspace and users present on channels. We don't store any private conversation and the logs are only collected from the open channel on the work. Using the SlackAPI which process POST requests, we poll the endpoint at regular interval and maintain a synchronized and incremental database. The response from slack is in form of json which is parsed and process to store the log in a format that can be easily processed. The json is process and save in database following the format <Sender><Receiver><Channel><Message><Timestamp>. In the analysis phase we perform communication analysis. The analysis is done asynchronously since heavy computation can take some time. All the result are stored in the SQLite database which is then used by web client to generate visualization. The network analysis as of now is done in python to make use of pathpy, a higher order network analysis library.

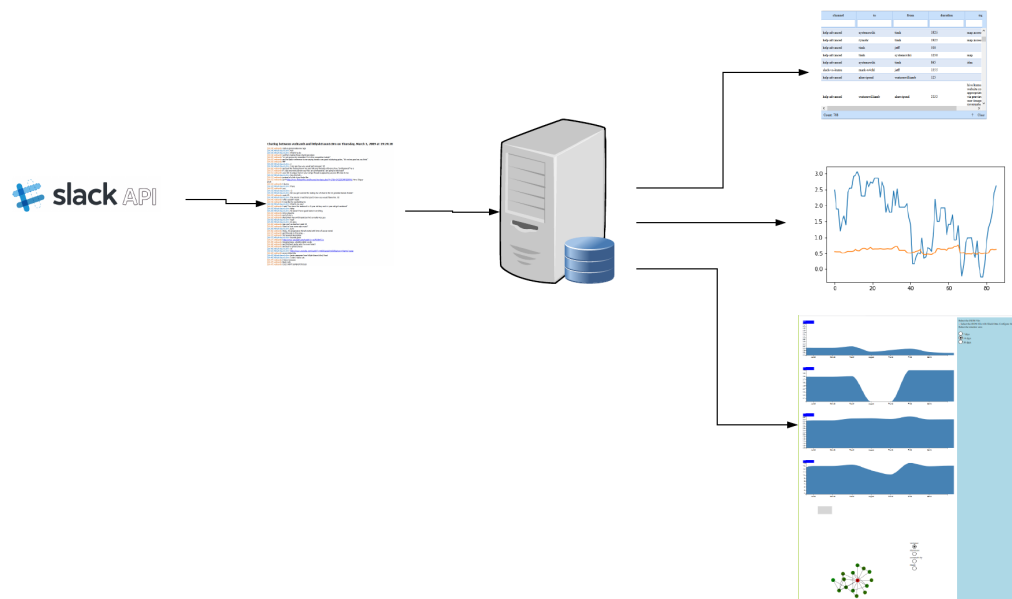


FIGURE 5.1: The Workflow of Slack Analysis

## 5.1 Authorization

Collecting user's slack data is possible via Slack API after the user provides access to the application. The access token can be obtained via the OAuth process which is built in Personal Analytics. When a user activates the slack tracker, they are redirected to the authorization endpoint. Once the user authorizes Personal Analytics for selected slack workspace, the access token is stored in user's local database which can be revoked at any point in future. The authorization request is a simple POST message to the slack endpoint.

## 5.2 DataCollection: Slack Tracker

The slack tracker implements three methods: `get_logs`, `get_channels`, `get_users` that sends a post request to Slack API endpoint and fetches information from workspace. To keep the database in sync, POST request to fetch channel history is sent every 20 minutes. Once the user authorizes the application, logs from slack workspace is stored in the database. The slack api provides `channel.history` method to allow querying slack for channel history. The parameters to the channel history are start and end timestamps etc. One limitation of this slack method is it can return a maximum of 1000 message in one response. If the mentioned time had more than 1000 message, the returned json is included with a key **has\_more** set as True. We implement a workaround by always checking for **has\_more** key and sending multiple post request.

### 5.3 Server: Analysis

Most of the analysis described in the previous Chapter is performed on local server through the Personal Analytics application. Although the network analysis and metric calculation are performed through a python script to make use of [**pathpy**] library which is higher order graph analysis library. The future work in this regard would be to move the analysis from Personal analytics to a separate application which can possibly be in python. The analysis module consist of submodules:-

- TextRank
- ChatDisentanglement
- Network

TextRank module makes use of OpenNLP library and the pre-trained tokenizer and part of speech tagger for the preprocessing stage. For building a logistic regression classifier in Chat Disentanglement module we make use of Accord Machine Learning library. The network module performs graph-based analysis i.e calculation of statistics for degree distribution and user interaction, and it makes use of pathpy library which provides graph algorithms.

#### 5.3.1 Network Analysis

We develop a separate python module teamviewer for network analysis to make use of pathpy and other python libraries. The teamviewer makes use of the links which are stored in form of a json file after communication analysis phase. Using pathpy we first calculate degree distribution for each window. Pathpy provide a `rolling_time_window` module that return an iter for aggregated graph. Once we have degree distribution of the graph we calculate the metrics value using scipy and numpy. The method used to calculate the metric is mentioned below.

1. skewness ->scipy.stats.skewness
2. kurtosis ->scipy.stats.kurtosis
3. varaince ->numpy.var
4. mean ->numpy.mean
5. connectivity ->pathpy.algorithms.spectral.algebraic\_connectivity

We also calculate the change in these metric values on removal of each node. The calculated metrics for rolling window and node effect is then stored as a json. The json is then used in the prototype for visualization of chart and network graph. The visualization is rendered as html page using flask on localhost. Flask also handles the python calls to make the visualisation interactive which allows the users to select time window size, json data file and color basis for the collaboration network.

## 5.4 Database

A Sqlite3 database is used to store all the raw chat logs and also the analysis results which are later used for visualisations. The raw information fetched from slack is stored in 3 tables namely the log, user and channel tables. These table contain all the necessary information required for analysis phase. To maintain privacy of user data, the database is stored locally.

## 5.5 Client: Visualization

The client side module is used to present the developed visualization. The client is a simple web server and the visualizations were built using d3 v5 which is latest and most powerful version d3 JavaScript library. We also introduce another module called teamviewer that provides a rolling window visualisation for the metrics discussed in analysis section. [5.2](#) show a momentary screenshot of the teamviewer module. The module was written in python to make use of pathpy library. Although currently the preprocessing that is link detection is done in the Personal Analytics and stored as a json, we plan to make teamviewer separate application and move the analysis to python. For interacting with the pathpy library and making the visualisation dynamic so user can configure the parameter, we use flask server where pathpy is used to calculate the degree distribution metrics. The visualisation are rendered as html on the localhost with port 5000 which is the default for flask.



FIGURE 5.2: Screenshot of prototype

## Chapter 6

# Conclusion

### 6.1 Limitations

#### 6.1.1 Problems with TextRank

TextRank works on co-occurrence and hence it is difficult to extract keywords from short messages. While there is some research that does topic extraction from short messages i.e from Tweets or status, we modify the current TextRank algorithm.

There are two stages where TextRank is used. First as a feature in the Chat Disentanglement Classifier and second as keyword extraction from the threads/cluster extracted from the log using the classifier.

1. Forming a keyword feature for log classifier For predicting whether two chat message belongs to the same thread or not we proposed a feature for the classifier based on the similarity of keyword set. We first tried using TextRank on these single messages but found that TextRank doesn't work well for such short messages and in many of the messages text rank couldn't identify any keyword. Given the length of the message we decided to calculate keywords for the entire log and then take common words from the tokenized message and the calculated keyword set as the keyword for a particular message.
2. Extracting Keywords from the conversation clusters Text Rank along with some modification and slack specific preprocessing as described in the above section was used on each of the conversation clusters. The keywords hence obtained is used in the Chat Disentanglement Classifier as a feature. Although we notice improvements to the classifier accuracy, we have not performed a validation over the text rank algorithm and measured the accuracy of the observed algorithm.

## 6.2 Future Work

Since this work was more explorative in nature, we could generate a lot of ideas that could be possible extended into separate use cases. The data used for building our chat-disentanglement classifier is limited and it would be interesting to see how it performs on IRC conversations or more closed slack channels. Accuracy of the classifier can also be improved by training it on a more diverse dataset. Since we had limited time and access to slack channels, it was not possible to explore more on improving the classifier. The other work can be validating the text rank method and keywords. One area that is also interesting in this regards is to study if word2vec can be used to improve keywords extraction.

We were not able to look at the sentiment of conversations which is another idea we find very useful. Sentiments along with the conversation network analysis can be further help in understanding the reasons for pattern change. Observing the change of sentiment over a different time period is also what we plan on adding to the current work.

The major idea that we plan on extending is the network analysis and understanding the change in network and relating it to time periods and sentiments. How does project progress when they are dependent on a few central users? What happens during release periods or schedule meeting and does sentiments over these periods relate to other network statistics like centrality. The idea is if the project depends on few people, then period of higher activity they might be under greater stress.

## 6.3 Source Code

The implementation of network analysis and the time slider visualization is present in this github repo. The implementation of Chat-Disentanglement classifier can be found at github-repo. The implementation of Slack Tracker to Personal Analytics is update and maintained in another git repo. Although this is not merged to master branch and in future we wish to build the slack-tracker as a separate application and not as a Personal Analytics module/plugin.



# Bibliography

- [1] Chintan Amrit Amrit and Jos van Hillegersberg. “Mapping Social Network to Software Architecture to Detect Structure Clashes in Agile Software Development”. Undefined. In: *Electronic Proceedings ECIS*. Theme of the conference is ”Relevant rigour - Rigorous relevance”. ECIS, June 2007, pp. –.
- [2] S. E. Bailey et al. “A Decade of Conway’s Law: A Literature Review from 2003-2012”. In: *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*. 2013, pp. 1–14. DOI: 10.1109/RESER.2013.14.
- [3] Melvin E. Conway. “How do Committee invent?” In: (1969).
- [4] Micha Elsner and Eugene Charniak. “Disentangling Chat”. In: *Comput. Linguist.* 36.3 (Sept. 2010), pp. 389–409. ISSN: 0891-2017. DOI: 10.1162/coli\_a\_00003. URL: [http://dx.doi.org/10.1162/coli\\_a\\_00003](http://dx.doi.org/10.1162/coli_a_00003).
- [5] Siwei Fu et al. “T-Cal: Understanding Team Conversational Data with Calendar-based Visualization”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: ACM, 2018, 500:1–500:13. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174074. URL: <http://doi.acm.org/10.1145/3173574.3174074>.
- [6] Nale Lehmann-Willenbrock, Stephenson J. Beck, and Simone Kauffeld. “Emergent Team Roles in Organizational Meetings: Identifying Communication Patterns via Cluster Analysis”. In: *Communication Studies* 67.1 (2016), pp. 37–57. DOI: 10.1080/10510974.2015.1074087. eprint: <https://doi.org/10.1080/10510974.2015.1074087>. URL: <https://doi.org/10.1080/10510974.2015.1074087>.
- [7] Manteli. “Adopting a Social Network Perspective in Global Software Development”. In: *2012 IEEE 7th International Conference on Global Software Engineering* ().
- [8] Shikib Mehri and Giuseppe Carenini. “Chat Disentanglement: Identifying Semantic Reply Relationships with Random Forests and Recurrent Neural Networks”. In: *IJCNLP(1)*. Asian Federation of Natural Language Processing, 2017, pp. 615–623.

- [9] R. Mihalcea and P. Tarau. “TextRank: Bringing Order into Texts”. In: *Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona and Spain, 2004.
- [10] Maja Saphir. *Problems with slack*.
- [11] Nils Christian Sauer and Simone Kauffeld. “The Structure of Interaction at Meetings: A Social Network Analysis”. In: *Zeitschrift für Arbeits- und Organisationspsychologie A&O* 60.1 (2016), pp. 33–49. DOI: 10.1026/0932-4089/a000201. eprint: <https://doi.org/10.1026/0932-4089/a000201>. URL: <https://doi.org/10.1026/0932-4089/a000201>.
- [12] M. Schwind, A. Schenk, and M. Schneider. “A Tool for the Analysis of Social Networks in Collaborative Software Development”. In: *2010 43rd Hawaii International Conference on System Sciences*. 2010, pp. 1–10. DOI: 10.1109/HICSS.2010.40.
- [13] Amy X. Zhang and Justin Cranshaw. “Making Sense of Group Chat Through Collaborative Tagging and Summarization”. In: *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (Nov. 2018), 196:1–196:27. ISSN: 2573-0142. DOI: 10.1145/3274465. URL: <http://doi.acm.org/10.1145/3274465>.
- [14] Xiaolei Zuo, Silan Zhang, and Jingbo Xia. “The enhancement of TextRank algorithm by using word2vec and its application on topic extraction”. In: *Journal of Physics: Conference Series* 887.1 (2017), p. 012028. URL: <http://stacks.iop.org/1742-6596/887/i=1/a=012028>.