PREDICTION OF USER BEHAVIOUR ON THE WEB

NIKOLAY BURLUTSKIY

A thesis submitted in partial fulfilment of the requirements of the University of Brighton for the degree of Doctor of Philosophy

2017

Contents

Acknowledgments

Foremost and the most importantly, I owe my sincere and endless appreciation to my supervisors Miltos Petridis, Andrew Fish, and Nour Ali. This thesis would not have been possible without their continuous guidance, support and advice. I also very thankful to Dr Stelios Kapetanakes, Dr Alexey Chernov, Dr Manos Panaousis, and Dr Babangida Abubakar. These people are very experienced academics who are always around for friendly advice. I am happy that I was delighted to share my office and have chats with other PhD students, including Mohammed Al-Obeidallah, Mousa Khubrani, Shaun Shei, Mansour Ahwidy, Micah Rosenkind, Ioannis Agorgianitis, Mithileysh Sathiyanarayanan, Marcus Winter, Tobias Mulling, Vasileios Manousakis Kokorakis, Eisa Alharbi, Eleftherios Bandis. I cannot mention everyone who helped and encouraged me to keep going in my research. Last but not least, I would like to thank Sam Lynch for many fruitful discussions on various topics during our regular coffee breaks and lunch time. I appreciate Joseph Cosier's effort in proofreading the thesis and providing some valuable input on the ideas challenged in this thesis. I would like to give an extra mention to Andrew Fish for his continuous enthusiasm and sincere interest in my PhD work. I could not even think about a better support during these long, hard but exciting days constituting for more than three years of my PhD programme. Also, I am thankful to Dr Frederic Stahl and Roger Evans for examining this work and providing useful feedback on my thesis. Finally, I would like to thank my parents, my sister, and my nephew for their support and their unconditional belief in me.

Declarations

I declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the original work of the author. The thesis has not been previously submitted to this or any other university for a degree, and does not incorporate any material already submitted for a degree.

[illegible] Nikolay Burlutskiy

Abstract

The Web has become an ubiquitous environment for human interaction, communication, and data sharing. As a result, large amounts of data are produced. This data can be utilised by building predictive models of user behaviour in order to support business decisions. However, the fast pace of modern businesses is creating the pressure on industry to provide faster and better decisions. This thesis addresses this challenge by proposing a novel methodology for an efficient prediction of user behaviour. The problems concerned are: (i) modelling user behaviour on the Web, (ii) choosing and extracting features from data generated by user behaviour, and (iii) choosing a Machine Learning (ML) set-up for an efficient prediction.

First, a novel Time-Varying Attributed Graph (TVAG) is introduced and then a TVAG-based model for modelling user behaviour on the Web is proposed. TVAGs capture temporal properties of user behaviour by their time varying component of features of the graph nodes and edges. Second, the proposed model allows to extract features for further ML predictions. However, extracting the features and building the model may be unacceptably hard and long process. Thus, a guideline for an efficient feature extraction from the TVAG-based model is proposed. Third, a method for choosing a ML set-up to build an accurate and fast predictive model is proposed and evaluated. Finally, a deep learning architecture for predicting user behaviour on the Web is proposed and evaluated.

To sum up, the main contribution to knowledge of this work is in developing the methodology for fast and efficient predictions of user behaviour on the Web. The methodology is evaluated on datasets from a few Web platforms, namely *Stack Exchange*, *Twitter*, and *Facebook*.

Abbreviations

AG - Attributed Graph

AGM - Attributed Graph Model

ANN - Artificial Neural Networks

BM - Boltzmann Machine

BN - Bayesan Network

CBR - Case-Based Reasoning

CNN - Convolutional Neural Network

CPU - Central Processing Unit

DBN - Deep Belief Network

DL - Deep Learning

DT - Decision Trees

EDA - Exploratory Data Analysis

EM - Ensemble Methods

GPU - Graphics Processing Unit

HMM - Hidden Markov Models

k-NN - k Nearest Neighbours

LR - Logistic Regression

LSR - Least Squares Regression

MAG - Multiplicative Attributed Graph

MART - Multiple Additive Regression Trees

ME - Maximum Entropy

ML - Machine Learning

MLE - Maximum Likelihood Estimation

MLP - Multi-Layer Perceptron	
NB - Naive Bayes	
NLP - Natural Language Processing	
PCA - Principal Component Analysis	
PDF - Probability Density Function	
PA - Passive-Aggressive	
Q&A - Question-and-Answer	
RBFN - Radial Basis Function Network	
RF - Random Forest	
RBM - Restricted Boltzmann Machine	
RNN - Recurrent Neural Network	
SGD - Stochastic Gradient Descent	
SSM - State-Space Models	
SVM - Support Vector Machines	
TAG - Time Aggregated Graph	
TVG - Time-Varying Graph	
TVAG - Time-Varying Attributed Graph	
XGB - Extreme Gradient Boosting	

Chapter 1

Introduction

The main motivation of the work in this thesis is to explore the challenge of predicting user behaviour on the Web. As a result, a novel methodology for an efficient prediction of user behaviour on the Web is proposed. The methodology is tested and evaluated in several experiments on real datasets, such as datasets from Stack Exchange, Twitter, and Facebook.

In this Chapter, the problem of predicting different aspects of user behaviour on the Web is introduced. First, the problem overview is explored in Section 1.1. Second, the motivation for predicting user behaviour is presented in Section 1.2. Third, the research questions along with the adopted research methodology and contribution to knowledge are discussed in Sections 1.3, 1.4, and 1.5 accordingly. Last, the organisation of the thesis is presented, the papers published as a result of this work are listed, and the whole work is summarised in Sections 1.6, 1.7, and 1.8 correspondingly.

1.1 Problem Overview

Recently, the scale of user behaviour on the Web, more specifically human interaction and communication on the Web, has grown very fast. As a result, the size of data generated by such behaviour has increased significantly. For example, the content generated only on Facebook in early 2016 is counted in thousands of terabytes per day and this number continues to grow. However, most of this data remains unused and forgotten.

In this thesis, user behaviour on the Web is defined as *human interaction* and *relationship development* on the Web. Human interaction involves messaging activities of users, for example, sending messages between users on Facebook, posting and sharing pictures, audio and video content. In case of Twitter, posting tweets and retweeting other tweets is considered as human interaction. The second part of user behaviour is relationship development. These relationships can be, for example, professional, friendship or romantic relationships. On Facebook these relationships involve friendship ties since users can 'friend' or 'unfriend' someone. Users of Twitter can follow each other or be followed by others. These relationships are dynamic since they can develop overtime, a relationship between people can appear and then disappear later.

The Web is an umbrella term for a number of platforms that allow people to communicate, interact, and exchange the generated information. Such media include social networking platforms, email services, web blogs, and Q&A forums. However, the understanding of user behaviour in these communities, how people interact with each other, and what kind of information can be deduced from their activities is still limited.

Nevertheless, researchers have tried to utilise data generated on the Web for different prediction tasks. For example, predicting user satisfaction of customer support on the Web [Karnstedt et al., 2010; Glady et al., 2009], user behaviour in social media [Zhu et al., 2013], predicting response times to questions [Yang et al., 2011; Dror et al., 2013] and tweets [Weerkamp and De Rijke, 2012], churn of users [Zhu et al., 2013], predicting effectiveness of online advertisements [McDuff et al., 2015] is a short list of such tasks.

Some researchers have worked on establishing factors influencing user behaviour on the Web. However, understanding the factors influencing user behaviour, the methods of extracting this information from the Web, and the approaches for predicting the behaviour is still at an early stage of development. Indeed, recent studies on predicting user behaviour have shown that the behaviour depends on many factors such as social, contextual, spatial, and temporal factors [Scellato et al., 2011; Guille and Hacid, 2012]. Due to the scale of data generated on the Web, studying these relationships and building predictive models is a challenging task involving substantial data mining efforts [Adedoyin-Olowe et al., 2013].

The problem scenario addressed in this thesis is the scenario when the efficiency of predicting user behaviour on the Web is crucial. To illustrate this scenario more specifically, a few examples of predicting user behaviour on the Web where the efficiency of a prediction is crucial are presented below. First, for example, predicting whether a tweet on Twitter will be retweeted within one hour or not fits well in the problem scenario. It is evident that if a very accurate ML model is built and the this model predicts that a particular tweet will be retweeted within one hour but to produce such a prediction takes more than two hours then such prediction is useless for a practical application. Another example is predicting whether a question asked on Q&A forum will be answered within 10 minutes or whether it will take longer than 10 minutes. In order to produce a prediction using ML approach, a set of features must be extracted, then a predicting model must be trained on these features and finally a prediction must be performed. However, if all this process requires more than 10 minutes then the prediction cannot be used for a practical application. Thus, this problem scenario where the efficiency of a prediction of user behaviour on the Web is crucial is addressed in this thesis.

Modelling User Behaviour on the Web

Modelling is a great tool for understanding and exploring the data generated by users on the Web. Researchers tried to build models of user behaviour on the Web using data such as [Shannon, 1948; Yu and Kak, 2012]. These models are always incomplete due to the complexity of the task, but the models can be useful for specific tasks. The era of Social Networks has made graph models popular to explore and analyse relationships between people as well as for analysing the dynamics of their communication [Xiong and Donath, 1999; Donath et al., 2010; Frau et al., 2005]. Also, modelling user behaviour can facilitate the exploration of the data and can help to find hidden patterns of human communication and interaction on the Web.

Exploring User Behaviour Visually

Visualisation can facilitate exploration of user behaviour. Indeed, visualisation can help to uncover hidden relationships and answer questions on how users communicate and how their relationships evolve over time. For example, visualisation of users' emails activity can provide insight into the conversational history of users' communication [Samiei et al., 2004]. Even though there has been long lasting research into exploratory visualisations of human communication, the choice of a visualisation for a particular task and a dataset is challenging [Smith and Fiore, 2001; Venolia and Neustaedter, 2003].

Extracting Features for Machine Learning Prediction

In order to build a good predictive model, it is necessary to understand what drives user behaviour on the Web as well as to find important attributes or factors the behaviour depends on. Once the data or the model representing user behaviour on the Web is explored and understood, these factors can be extracted as features.

Then these features can be used to build a predictive model of user behaviour on the Web [Cheng et al., 2011; Steurer and Trattner, 2013]. Several researchers have analysed different online communities, their motivation and behavioural dynamics on the Web [Wang et al., 2013; Li et al., 2012; Horowitz and Kamvar, 2010; Morris et al., 2010]. Since the data generated by online communities is large, the feature extraction step often involves reducing the size of the data. Ideally, this step is performed without much loss of information on the user behaviour but the reduced data can still be used for building an accurate model of user behaviour.

Choosing a Machine Learning Set-up

Accuracy and speed of prediction are crucial in many businesses. For example, the ML company called 'SkyTree'[1] reports several business challenges in fraud detection where it is required to provide prompt and accurate results. This may be achieved by frequent model updates as well as by using a combination, or an ensemble, of ML methods [SkyTree, 2014].

There is a tendency for more complex models to provide more accurate but slower predictions than simpler models [Loumiotis et al., 2014]. Thus, there is a need to find a trade-off in the accuracy and the speed of the prediction, especially in the cases when the speed of the prediction is crucial [Bifet et al., 2015].

Choosing an appropriate ML set-up for building a good predictive model from extracted features is a challenging task due to the number of existing ML algorithms and their variations. Indeed, there is no algorithm dominating others in terms of performance. Most ML techniques involve a substantial time of trial and error to find the most appropriate ML set-up for a particular task and domain [Domingos, 2012]. However, researchers are actively working on building guidelines for how to choose a particular ML set-up depending on the size and provenance of a dataset, prediction task, available hardware and software resources, and the technical complexity of the set-up [Sculley et al., 2015].

1.2 Motivation

Predicting user behaviour on the Web has a wide range of practical applications. A list of possible applications includes:

- *Predicting user purchasing activities.* By analysing how people discuss a particular advertised product and whether a product is recommended by people

[1]skytree.net

to others, it is possible to predict sales of a product [Badea, 2014]. Also, the retail industry is interested in analysing and predicting customer behaviour and their purchasing patterns to develop marketing and stock replenishment strategies [Park and Seo, 2013].

- *Predicting churn of users.* Churners are users who leave a Web service for the benefit of a competitor [Karnstedt et al., 2010]. Sometimes churn is defined as a significant decrease in user activity. Telecommunication and computer network providers are interested in identifying such users to evaluate their loyalty [Glady et al., 2009]. By predicting users who are likely to leave the service soon, it is possible to take an action, for example, call a customer or send him/her an email in an attempt to keep the customer.

- *Identifying spammers and criminals.* Predicting unusual user behaviour can help to detect spammers and criminals [Aneetha et al., 2012]. Also, predicting high risk customers can help banks and the financial sector to minimise fraud and financial losses [Wei et al., 2012].

- *Predicting user complaint activities.* By predicting users who are likely to complain about a service, it is possible to take an action to improve users' experience with a service [Jernite et al., 2013]. Another related application is predicting users who are not satisfied with a service [Larivière and Van den Poel, 2005].

- *Predicting user posting activities.* Predicting trends of user behaviour, for example, explicitly highlighting and predicting the dynamics of tweets in Twitter, facilitates users' understanding of popular tweets at a particular time [Zhang et al., 2014].

- *Improving user satisfaction.* Providing a user with an expected time for when his/her tweet will be retweeted can improve user experience on the Web [Artzi et al., 2012]. Also, predicting whether user's question on a Q&A forum will be answered can improve user satisfaction of using the forum [Yang et al., 2011; Dror et al., 2013]. For example, by providing a user of a Q&A community with an estimation of the response time to their questions, the frustration of silence on an online community can be mitigated. Furthermore, users can be recommended to change their questions in order to make their questions answerable faster [Bhat et al., 2014; Lezina and Kuznetsov, 2012].

There are plenty of possible architectural implementations for such applications. However, most of these architectures can be generalised in a pipeline (see Figure

1.1) [Pentreath, 2015]. User behaviour on the Web leads to data ingestion which can be streamed in real time. For example, thousands of tweets are posted on Twitter per second in 2017. This data is usually stored in a database. However, the size of the data is large and it requires a preprocessing step before building a ML model. For this purpose, a Data Cleansing and Transformation step is performed (see Figure 1.1). This step includes a feature extraction procedure in order to decrease the size of the data and to use only relevant data. Afterwards, a model training and testing step is performed and finally a model deployment and integration is conducted.

There are a few possible issues of processing speed and response time of such a ML pipeline. First of all, due to the fact that a lot of data is generated by user behaviour on the Web, the feature extraction step must be performed in an efficient way (see Figure 1.1, Data Transformation step). Otherwise, in the case when this step takes long time, the data for building a model will be derived and processed too late. As a result, the model will be outdated or even the prediction made by the model will be irrelevant. For example, in case of predicting when a user replies to a tweet on Twitter, a model can provide a prediction of the tweet reply to be expected in two minutes but feature extraction step might take more than two minutes what makes such a system too slow for a practical application. Second, model building and testing can be a crucial step as well. Similar to the feature extraction step, if model training and testing takes long time then the predictions made by such a model can be obtained too late for them being useful. Again, for predicting question response times on a Q&A forum where the response times are often less than a minute, a predictive model must be capable of providing predictions faster than fractions of minutes.

The requirements for the speed of classification, as was mentioned previously, depends on the problem. For predictions of response time on Twitter, Facebook, and popular Q&A forums, such as Stack Exchange forums, the prediction must be made in a few seconds or minutes since the prediction must be made before the predicted event will happen. The requirements for the accuracy of predictions are also task dependent. For example, predicting a question to be answered within a certain interval better than predicting using a naive approach can add value to a Q&A platform. For instance, improving the prediction accuracy by a few percent it is possible to redirect more questions to experts. Thus, the speed of answering questions on Q&A forums can be improved. However, in some sensitive scenarios, for example, predicting criminal behaviour on the Web requires more confident predictions which in some cases must be near 99% due to high risks caused by possible false predictions. In this thesis, the stress is on less sensitive scenarios

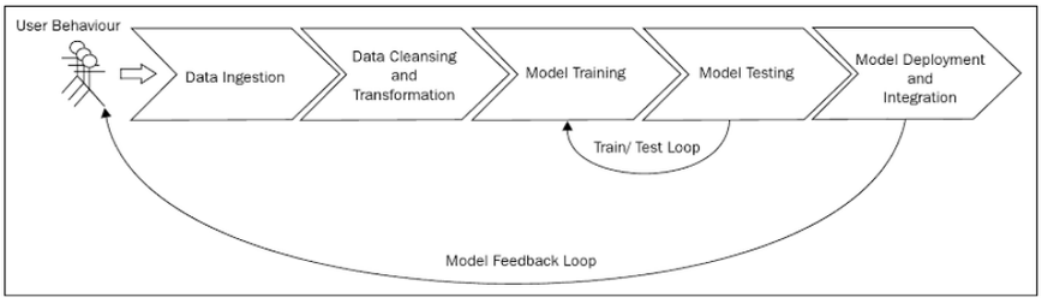


Figure 1.1: A general pipeline for a machine learning system.

where improving naive approaches by a few percent can lead to important benefits due to the large size of data.

In this thesis, a continuous pipeline for a ML system (Figure 1.1) is considered and two crucial steps of feature extraction and then building and testing ML models are of interest. Predicting user behaviour on the Web efficiently depends on these two steps.

1.3 Research Questions

The main research question formulated in this thesis can be stated as follows:

"How can one model, explore, and predict user behaviour on the Web efficiently using data mining techniques?"

The main research question is split into six sub-questions addressed in this thesis. The first step in answering the main research question leads to the first sub-question addressing the problem of modelling user behaviour on the Web, more specifically modelling human interaction and relationship development on the Web:

1. How can one model human interaction and relationship development in order to capture, explore, and facilitate understanding of user behaviour on the Web?

The second step is to extract features for building predictive models of user behaviour on the Web. This step leads to the second and the third sub-questions addressing the problem of extracting these features for fast but accurate predictions as well as evaluating how temporal features are important for such predictions:

2. How can one construct a set of features for an *accurate* but *fast* prediction of user behaviour on the Web?

3. How important are temporal features for prediction in terms of accuracy and time efficiency?

The third step is to build a predictive model of user behaviour efficiently. This step involves choosing an efficient ML set-up for building such models. One can choose an online or offline setting, a state-of-the-art algorithm or a more advanced deep learning algorithm for training and testing the predictive models. As a result, the following two sub-questions are addressed:

4. How do *online* and *offline* algorithms compare in terms of their time complexity and accuracy performance in predicting user behaviour?

5. How does the performance of state-of-the-art ML algorithms compare? Is deep learning advantageous compared to other algorithms?

Finally, there is an ongoing work on visually exploring user behaviour on the Web. Visualisation is a powerful tool for exploring user behaviour on the Web, however, choosing appropriate visualisations for exploring user behaviour is challenging. Thus, the last sub-question addresses the problem of how to automatise the process of choosing visualisations:

6. How can one automatically choose appropriate visualisations to explore user behaviour on the Web?

1.4 Research Methodology

The research methodology adopted in this thesis consists of five steps. Each step is discussed below.

The first step is developing a model of user behaviour on the Web in order to capture user behaviour on the Web, defined as human communication and relationship development on the Web. Since the model is designed using a fusion of a Time-Varying Graph (TVG) and an Attributed Graph (AG), the resulted Time-Varying Attributed Graph (TVAG) model explicitly captures temporal and structural properties of user behaviour on the Web. In order to demonstrate the applicability of the proposed model, three datasets are used, Stack Exchange, Twitter, and Facebook. As a result, it is shown that it is feasible and straightforward to model real world social platforms using the proposed model.

The second step is to extract features efficiently from the proposed TVAG-based model of user behaviour on the Web. All features are classified into two groups, semantic and computational, and then the effect of these groups of features on accuracy and time performance is estimated for a binary classification task across three different datasets from Stack Exchange and Twitter. Four state-of-the-art ML algorithms are used, namely k-NN, LR, DT, and XGB. As an evaluation metric, accuracy, F-measure, and time performance are measured.

The third step is to compare the performance of the predictive models trained using two different learning modes, online and offline (batch) learning modes, to predict user behaviour on the Web. Since these two learning modes are conceptually different, an evaluation procedure is proposed and executed. As an experiment, four offline (batch), and three online learning algorithms are executed across three different datasets from Stack Exchange and Twitter. As an evaluation metric, accuracy and time performance are used for comparing the performance of these two modes.

Next, a Deep Learning (DL) algorithm, Deep Belief Network (DBN), is designed for predicting user behaviour on the Web and then its performance is compared to the previously introduced four offline (batch) and three online ML algorithms. Again, as an evaluation metric, accuracy and time performance are used.

Finally, the problem of choosing visualisations of user behaviour on the Web automatically is addressed. For that purpose, a Case-Based Reasoning (CBR) approach for choosing visualisations of user behaviour on the Web is proposed and then tested on a real dataset collected from the IBM Many Eyes platform. Then the results of choosing visualisations automatically by the proposed CBR approach are compared to the results of choosing visualisations by participants in an empirical study. Finally, the difference in the results is analysed both qualitatively and quantitatively.

1.5 Contribution to Knowledge

The main contribution to knowledge of this thesis is through providing a systematic methodology to predict user behaviour on the Web. In more detail, the list of contribution is as follows:

1. Proposed a novel methodology to predict user behaviour on the Web.

2. Designed a Time-Varying Attributed Graph (TVAG) where each node and edge of the graph is a time-varying attributed object. Then designed a novel model of human communication and relationship development based on two such graphs. Demonstrated how this model can be used for modelling user behaviour on Facebook, Twitter, and Stack Exchange platforms.

3. Introduced a novel systematic approach of feature extraction from the pro-

posed model to predict user behaviour on the Web. These features are classified into semantic and computational complexity groups. Evaluated contribution of these feature groups to accuracy and time performance of a binary classification task. Proposed a guideline for efficient feature extraction and discussed scenarios when this guideline is advantageous.

4. Predicted users' response time for the largest Q&A platform, Stack Exchange, using the proposed methodology. Showed that temporal features are important predictors in predicting user behaviour on the Web.

5. Predicted tweets being retweeted for the largest micro-blogging platform, Twitter, using the proposed methodology. Showed that structural features are important predictors in prediction of user behaviour on the Web.

6. Proposed a procedure for comparison of online and offline (batch) learning modes. Showed empirically that online learning can be much faster than offline learning without much loss in the accuracy of predicting user behaviour on the Web.

7. Showed that Deep Learning (DL), more specifically Deep Belief Network (DBN), can be superior to other ML methods in terms of accuracy but slower in training and predicting.

8. An additional contribution is a proposed novel method based on Case-Based Reasoning (CBR) to choose visualisations of user behaviour on the Web. The method was evaluated on IBM Many Eyes platform as well as an empirical study for an evaluation of the proposed method was conducted.

1.6 Thesis Organisation

This thesis is organised as follows:

Chapter 1 introduces the problem addressed in this thesis, motivation for working on this problem, research questions, methodology, contribution to knowledge, organisation of this thesis, and papers published as a result of working on this thesis.

Chapter 2 discusses Background and Related Work.

Chapter 3 introduces a model based on a Time-Varying Attributed Graph (TVAG) to model user behaviour on the Web.

Chapter 4 introduces a feature engineering procedure and an approach on how to estimate the efficiency of feature extraction.

Chapter 5 explores the problem of choosing a ML set-up for a prediction task in the context of user behaviour on the Web. Online and offline (batch) learning modes are compared. Also, a deep learning approach is designed and compared to other state-of-the-art ML approaches.

Chapter 6 provides an approach on how can one automatically choose visualisations for the exploration of user behaviour on the Web.

Chapter 7 revisits the research questions, summarises the main findings and contributions of this thesis, and, finally, proposes a few directions for further research.

1.7 Publications

This thesis resulted in the following publications:

1. **N. Burlutskiy, M. Petridis, A. Fish, and N. Ali, *Modelling User Behaviour on the Web with Time-Varying Attributed Graphs*, Social Networks, An International Journal of Structural Analysis, Editors: M. Everett, T.W. Valente (submitted in April 2017).** The content of this paper describes the proposed TVAG-based model of user behaviour on the Web introduced in Chapter 3.

2. **N. Burlutskiy, M. Petridis, A. Fish, N. Ali, and A. Chernov, *An Investigation on Online versus Batch Learning in Predicting User Behaviour*, Thirty-sixth SGAI International Conference on Artificial Intelligence (AI-2016), England, Cambridge (accepted).** The content of this paper describes the experiments introduced in Chapter 5.

3. **N. Burlutskiy, M. Petridis, A. Fish, and N. Ali, *Predicting Users' Response Time in Q&A Communities*, IEEE International Conference on Machine Learning and Applications (ICMLA 2015), USA, Miami, pp. 618-623, 12/2015.** The content of this paper describes the experiments introduced in Chapter 4 and 5 of this thesis.

4. **N. Burlutskiy, M. Petridis, A. Fish, and N. Ali, *How to Visualise a Conversation: A Case-Based Reasoning Approach*, 19th UK Workshop on Case-Based Reasoning (UKCBR 2014), England, Cam-**

bridge, pp. 27-38, 12/2014. This paper introduced the proposed method for choosing visualisations for data exploration described in Chapter 6.

5. **N.Burlutskiy, A. Fish, M. Petridis, and N. Ali, *Enabling the Visualization for Reasoning about Temporal Data*, IEEE Conference on Visual Languages and Human-Centric Computing, Australia, Melbourne, pp. 179-180, 08/2014.** This paper introduced the importance of visualisation for reasoning over data as well as requirements for such a system (Chapter 6).

While working on this thesis, the author contributed to a few related investigations which led to the following publications:

6. **M. Sathiyanarayanan, and N. Burlutskiy, *Visualizing social networks using a treemap overlaid with a graph*, Procedia Computer Science 58, pp. 113-120, 08/2015.**

7. **M. Sathiyanarayanan, and N. Burlutskiy, *Design and evaluation of Euler diagram and treemap for social network visualisation*, Communication Systems and Networks (COMSNETS), 2015 7th International Conference on, India, Bangalore, pp. 1-6, 06/2015.**

8. **M. Petridis, S. Kapetanakis, J. Ma, and N. Burlutskiy, *Temporal Knowledge Representation for Case-Based Reasoning Based on a Formal Theory of Time*, International Conference on Case-Based Reasoning (ICCBR 2014), Cork, Ireland, 09/2014.**

1.8 Summary

This Chapter introduced the problem of predicting user behaviour on the Web. First, the problem of predicting user behaviour on the Web was discussed. Second, the motivation for producing such predictions was stated. Third, the research questions addressed in this thesis along with the research methodology and the contribution to knowledge were declared. Finally, the structure of this thesis and the list of peer-reviewed papers published by the author were introduced.

In the next Chapter, background and related work are discussed.

Chapter 2

Background and Related Work

This Chapter leads the reader through the most recent research findings in the area of predicting user behaviour on the Web. Also, the Chapter provides the necessary background for modelling and visualising of user behaviour, as well as the background for feature extraction and state-of-the-art ML techniques to predict user behaviour on the Web.

First, an overview of prediction of user behaviour on the Web is introduced in Section 2.1. Second, prediction tasks are listed in Section 2.2. Third, models of user behaviour are reviewed in 2.3. Fourth, visual data exploration of user behaviour on the Web is over-viewed in Section 2.4. Fifth, feature extraction techniques are discussed in Section 2.5. Sixth, the state-of-the art ML methods for prediction are discussed in 2.6. Finally, the Chapter is summarised in Section 2.7.

2.1 User Behaviour on the Web

Many users regularly use the Web to share their interests, stay connected, discuss various topics, share and obtain information. As a result, vast amounts of data are generated by these users and then one can use this data to produce useful predictions of certain user related behaviour. These predictions can be used in various domains, including finance, product marketing, social dynamics, public health, and politics. One of the consequences of this is that an increasing number of researchers have been attracted to this subject [Yu and Kak, 2012].

There are many platforms where users can communicate, share their interests, stay connected, discuss various topics, and exchange information. These platforms include, for example, social networks, Q&A forums, and digital communication networks. Following is a short overview of those platforms.

Figure 2.1: Different Social Networking Sites on the Web.

Social Networks

Social networks are popular platforms for socialising, interacting, and sharing information on the Web (see Figure 2.1). Social Networks platforms include *Twitter* and *Facebook*; Q&A forums include, for example *Quora* and *Stack Exchange*. *Instagram* and *Flickr* are examples of platforms where users share pictures. Also, digital newspapers, such as the *Guardian* and the *New York Times*, allow users to leave their comments and interact with other users. The common features of these social platforms are, first, the presence of a *social structure* and, second, the presence of mechanisms to *interact* and *disseminate* information. In 2016, two of the largest and most successful social networks were *Facebook* and *Twitter*:

- *Facebook*: The largest social network where people can 'friend' each other and communicate with each other. A common abstract model for representing the *Facebook* structure is a *social graph* where people are nodes and their relationships are edges of the graph. Also, since the individuals can send messages to each other, a second graph, an *interaction graph*, can be constructed where nodes represent people and directed edges represent messages being sent.

- *Twitter*: In this platform, people can 'follow' each other, be followed by others, or follow someone. Thus, a directed *social graph* of people (nodes) and their relationships (edges) can be constructed. Also, people can 'tweet' or 'retweet' each other which allows one to model the tweeting activity of users in the form

Figure 2.2: Different Q&A Communities on the Web.

of a dynamic *interaction graph*.

Q&A Forums

Question-and-Answer (Q&A) communities are represented by thousands of independent forums as well as forums which are part of another platforms (see Figure 2.2). The largest Q&A communities are *Stack Exchange*, *Quora*, *Baidoo*, *Yahoo!*, and *Naver*. The functionality to ask and answer questions is the characteristic which unites all these Q&A forums. Two of the most popular Q&A forums are presented below:

- *Quora*: This platform is a Q&A social network where people can follow each other. Users can also ask and answer questions. Again, both a social graph as well as an interaction graph can be constructed.

- *Stack Exchange*: This Q&A platform is represented by a family of Q&A websites where users ask and answer questions on various topics. These websites do not have an explicit social graph where people can 'friend' each other but an interaction graph of question asking and answering activity can be constructed.

Digital Communication Networks

The Internet has given people the ability to communicate globally through email and instant messengers. *Microsoft*, *Google*, and *Yahoo* are examples of such email services. *Facebook* messenger, *Kakao Talk*, *WhatsApp*, and *Kik* are examples of instant messengers. Some of these communication networks allow the construction of a social graph, a graph where people are interconnected by social ties. From another side, the interaction between people can form another graph, an interaction graph.

Similarities and Differences of the Platforms

The aforementioned platforms allow people to communicate and to build various social ties. As a result, the platforms form at least two different types of networks, namely a social or friendship network and interaction network. Also, all these social media platforms have temporal characteristics: for example, users add new friends and 'unfriend' others over time. Therefore, the topology of the social network formed by these communities can change over time. The same can happen with the interaction network since messages are timestamped and the temporal properties of the interaction network are known too. Nevertheless, every platform has its own data format representing communication between people. As a result, first, different predictions can be performed over different platforms and, second, each platform requires an individual approach to understand and predict user behaviour on the Web. This fact motivates one to provide a general graph-based model of user interaction and relationship development on the Web.

Next, a set of the most common prediction tasks is introduced and discussed.

2.2 Prediction Tasks

There is a vast number of papers in which researchers have tried to predict user behaviour on the Web. These papers include but are not limited to the prediction of purchasing activities [Badea, 2014], customer behaviour [Park and Seo, 2013; Zheng et al., 2013], churn of users [Karnstedt et al., 2010], users' loyalty [Glady et al., 2009], identification of spammers and criminals [Aneetha et al., 2012; Wei et al., 2012]. To add to this list, researchers predicted user complaint activities [Jernite et al., 2013], and predicted users who will not be satisfied with a service [Larivière and Van den Poel, 2005]. Predicting user satisfaction [Yang et al., 2011; Dror et al., 2013; Artzi et al., 2012] and improving user's experience on the Web by predicting temporal properties of user's activity [Bhat et al., 2014; Lezina and Kuznetsov, 2012] is an active area of research as well. Predicting one's location at a particular time [Sadilek and Krumm, 2012] is another area of interest in predicting user behaviour on the Web.

Due to the diversity of user behaviour on the Web and the variety of prediction tasks, there are at least two common problems associated with these predictions. First, the accuracy of the prediction and, second, computational complexity and the resources required to perform the prediction. For example, in [Zheng et al., 2013] the authors predicted customer restaurants preference based on check-ins posted in social media. The authors analysed 121,000 Foursquare check-ins in restaurants in the Greater New York City area and then they built two predictive models. However, it was not mentioned how computationally expensive it was to train an accurate model which can be unacceptable when a near real-time prediction is required.

An approach for predicting human location at a particular time was proposed in [Sadilek and Krumm, 2012]. The authors used historical data of past human location to extract patterns in user behaviour. Also, it was shown that predicting one's location in the distant future is, in general, highly independent of the recent location of that person but, on the contrary, one's location exactly one week from now is a good predictor. Again, the author concentrated on providing an accurate prediction, however, the time spent on preprocessing and then training a model can be unacceptable for near real-time predictions.

Another group of researchers predicted human activities at home [Nazerfard and Cook, 2013; Choi et al., 2013]. These activities include having breakfast or, for example, taking medication. The authors also proposed an approach to predict the start time of the next user activity. Again, the problem of having a fast and accurate prediction was not a concern.

Predicting user posting activities [Zhang et al., 2014] has attracted many researchers as well. Predicting such user behaviour in social media includes predicting the posting time of messages in Q&A forums [Yang et al., 2011; Dror et al., 2013; Zhu et al., 2013; Burlutskiy et al., 2015], churn of users [Zhu et al., 2013], response time to a tweet [Weerkamp and De Rijke, 2012]. For example, an initial attempt to build a predictive model for *Twitter* was conducted in [Spiro et al., 2012]. In this paper the authors constructed a model for predicting the time between information dissemination and redistribution on *Twitter*.

Due to the variety of user behaviour on the Web and, as one of the results, the variety of prediction tasks, the research community has attempted to provide different models of user behaviour. These models are discussed in the next Section.

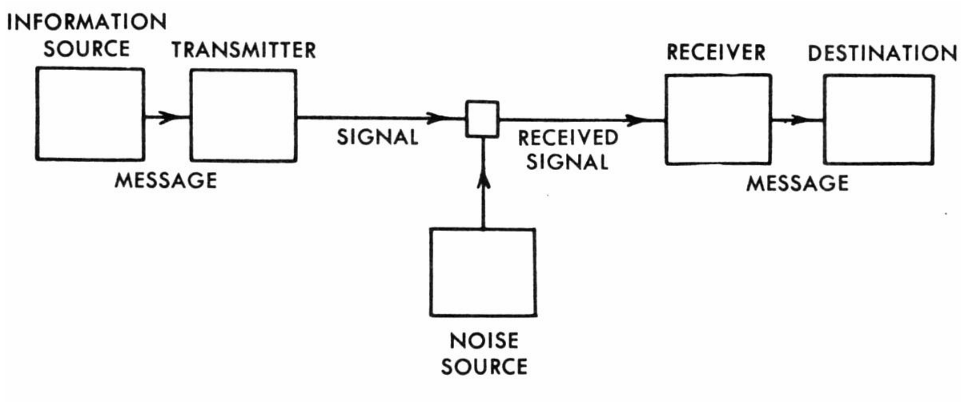


Figure 2.3: The model of communication from [Shannon, 1948].

2.3 Models of User Behaviour on the Web

User behaviour on the Web can be considered as a process of communication and interaction between users on the Web. One of the first and simplest models of communication is reflected in the work of [Shannon, 1948]. This model consists of a *transmitter*, a *message*, a *channel* where the message travels, noise or interference, and a *receiver* (see Figure 2.3). Wilber Schramm in his work [Schramm and Roberts, 1971] extended the model of communication with an emphasis on *feedback* in communication. The traditional communication model can be easily applied for social media. For example, by substituting keyboard for *encode/decode* and screen-to-screen for *channel*. The people communicating in social media can be considered being both *senders* and *receivers* worrying about correct *feedback*.

In this thesis, modern models of user behaviour are split into two large groups, dynamic models and graph-based models. Dynamic models are based on control theory and system theory whereas graph-based models use graph theory and social networks theory. First, dynamic models are introduced.

2.3.1 Dynamic Models

Generally speaking, a dynamic model represents the behaviour of an object over time. Usually such models are considered as a set of states ordered in a sequence. In case of user behaviour on the Web, such objects are people and dynamic models represent their behaviour. For example, in [Pentland and Liu, 1999] the researchers proposed to consider a human "as a device with a large number of internal mental states, each with its own particular control behaviour and inter state transition probabilities". Each "internal mental state" can be written as a single dynamic process:

$$\begin{aligned} \dot{x}_k &= f_k(x_k, t) + \xi(t) \\ y_k &= h_k(x_k, t) + \nu(t) \end{aligned} \tag{2.1}$$

Where the function f_k models the dynamic evolution of the state vector x_k at time k. Both ξ and ν are white noise processes with known spectral density matrices. The observations y_k is a function h_k of the state vector x_k.

Then this system described in Equation 2.1 can be used for analysis, exploration and predictions. The predictions can be performed by using, for example, Kalman filters [Pentland and Liu, 1999; De la Rosa et al., 2007].

In control engineering a concept of State-Space Models (SSM) has been applied for various domains [Sontag, 1998]. SSM is a mathematical model of a physical system. Such systems consists of input, output and state variables which are usually related by differential equations [Sontag, 1998]. An example of using SSM for modelling user behaviour on the Web was proposed in [Radinsky et al., 2012]. The authors showed how to use this model for predicting user activity such as query clicks and url clicks. This model uses temporal features as well as some domain-specific features such as the number of clicked urls and query-click entropy.

Nevertheless, graph-based models are more common models for modelling user behaviour compared to the dynamic models described in this Subsection. In the next Subsection graph-based models are introduced and discussed.

2.3.2 Graph-based Models

A graph-based model assumes that a graph is used for modelling user behaviour. Usually, the nodes of such a graph are associated with people and the edges connecting the nodes are associated with some sort of communication or connection between the people. Modelling user behaviour as a graph allows to capture structural properties of the network formed by these people. Indeed, a plenty of graph-based model have been proposed. Below is a brief list of such related research works.

A graph-based model of a *Twitter* network was proposed and then built in [Zhang et al., 2014]. In their paper, the authors proposed using a graph $G = (U, E)$ for describing users activity on *Twitter* where the nodes U were associated with people and the edges E were associated with the relationships of following users or users being followed by other users. Modelling a *Twitter* network as a graph allows one to learn about the structure of relationships between people, facilitates understanding which people are more active and who are central in the network. The structure of such networks can by analysed formally using the graph theory.

Indeed, such *social graphs* have shown importance for many applications such as improving security and performance of network systems in detecting email spammers [Garriss et al., 2006], improving Internet search [Gummadi et al., 2006], and defending against Sybil[1] attacks [Yu et al., 2006].

Enhancing a basic graph-based model of user behaviour introduced earlier is a large area of research. For example, in order to capture the structure of social networks as well as the features of the nodes and the edges in a network, a few statistically sound models of social networks have been proposed [Toivonen, 2009]. Nevertheless, many models consider only the networks where the nodes have no attributes which limits the scale of multifaceted nature of user behaviour to be captured [Toivonen, 2009]. As a result, several recent works has attempted to extend classical graph models for real-world social networks. One of the goals of such extension is to enable a powerful mathematical analysis for real-world social networks. One direction is to consider nodes of such networks having *attributes* [Toivonen, 2009; Kim and Leskovec, 2011; Pfeiffer et al., 2014] and the second direction is to use graphs in combination [Zhang and Zhang, 2013].

The first direction for extending graphs is performed by defining *attributes* for nodes of a graph. There are two graph-based models associated with such graphs extended by attributes. These are a 'Multiplicative Attribute Graph' (MAG) and an Attributed Graph Model (AGM). In [Kim and Leskovec, 2011] a MAG was proposed and then it was shown that the nodes with attributes allow to capture the interactions between occurrence of links in a clean and tractable manner. The nodes of such a graph have categorical attributes which encode some information on the people in the network. The probability of an edge in this graph is the product of individual attribute link formation affinities. As a result, this graph can be used for predicting, for example, unobserved edges. By predicting whether there is an edge existing between a pair of nodes, one can deduct the structure of such a social network. However, their model was developed only for a specific purpose, to capture the edges of a graph. Nevertheless, it was shown that MAG allows one to capture structural properties of real-world networks as well as attributes of nodes in such networks, for example, the age, status, gender of people in a social network.

An Attributed Graph Model (AGM) framework for capturing network structure and node attributes simultaneously was proposed in [Pfeiffer et al., 2014]. The authors outlined an efficient method for estimating the parameters of AGM. In another paper [Campbell et al., 2013], the authors mentioned that an attributed graph

[1]The Sybil attack is an attack in peer-to-peer networks wherein a system is corrupted by faking identities of the networks.

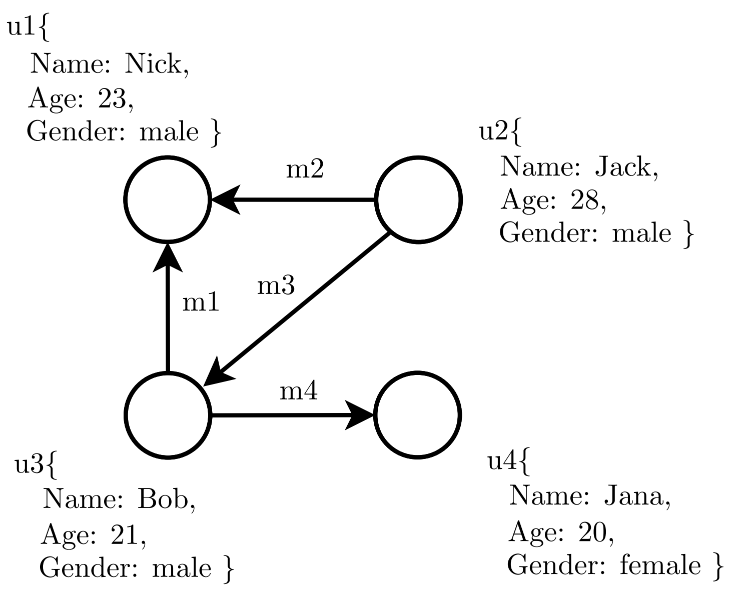


Figure 2.4: An example of an attributed graph with four attributed nodes u_1, u_2, u_3, u_4 and four edges m_1, m_2, m_3, m_4. Each node has three attributes, *Name, Age, Gender.*

is a powerful representation of social media networks. The authors stated that introducing attributes on edges can enhance the expressiveness of a graph. An example of an attributed graph is shown in Figure 2.4.

The second direction is to improve expressiveness of a graph-based model by using a few graphs in combination. For example, in [Zhang and Zhang, 2013] the authors introduced a combination of an *uncertain graph* and an *attributed graph* which is called the 'uncertain attribute graph' as a model of a social network. The authors argue that this graph allows one to capture both rich information of social networks as well as uncertainty of this information.

All these aforementioned works [Toivonen, 2009; Kim and Leskovec, 2011; Pfeiffer et al., 2014; Zhang and Zhang, 2013] only capture the attributes of the nodes of a graph. However, introducing attributes not only for the nodes but to the edges as well can improve expressiveness of a model of user behaviour. Even more, extending the model into the temporal domain can be advantageous as well. This extension is important, firstly, because a model with temporal properties allows one to capture and analyse temporal behaviour of users on the Web. Indeed, the time is a natural and inseparable characteristic of user behaviour. Secondly, the temporal properties of a model can be used for extracting temporal features for ML prediction tasks. Lastly, explicit temporal properties allow to cut graphs into sub graphs for faster computations of graph properties on the resulting subgraphs rather than using the whole graphs.

Indeed, the majority of graph-based models are a-temporal or time is captured implicitly. However, there exist a few temporal models of human communication on the Web. For example, Time-Varying Graphs (TVG) which are introduced below.

Time-Varying Graphs (TVGs)

Time-Varying Graphs (TVGs) have been developed to describe a wide range of dynamic networks [Casteigts et al., 2010]. The *nodes* of a TVG are defined as a set of entities U, and the *edges* are a set of relations E between these entities. Also, an alphabet L accounts for any property of a relation. That is, $E \subseteq U \times U \times L$. The definition of labels L is domain specific and left open. Having a distinct label L, the set E allows multiple relations between entities. The relations between entities are defined over a time span $T \in \mathbb{T}$ called the lifetime of the system. The temporal domain $\mathbb{T}$ is $\mathbb{N}$ for $discrete - time$ systems or $\mathbb{R}^+$ for $continuous - time$ systems. The dynamics of the system can be described by a TVG, $G = (U; E; T; \rho; \zeta)$, where

- $\rho : E \times T \to \{0, 1\}$, called presence function, indicates whether a given edge is available at a given time.
- $\zeta : E \times T \to \mathbb{T}$, called latency function, indicates the time it takes to cross a given edge if starting at a given date (the latency of an edge could vary in time).

The sequence $S_T(G) = sort(\cup\{S_T(e) : e \in E\})$, called characteristic dates of G, corresponds to the sequence of dates when appearance or disappearance of an edge occur in the system. Such events can be viewed as evolution of the graph G. The evolution of G is described as the sequence of graphs $S_G = G_1, G_2, .., G_{n-1}, G_n$ where G_i corresponds to a static snapshot of a graph at time $t = i$. In general case, $G_i \neq G_{i+1}$.

By restricting the timespan $\mathbb{T}$ of a graph G, a temporal subgraph $G' = (U; E'; T'; \rho'; \zeta')$ can be derived.

TVG is a powerful mathematical model for capturing temporal properties of a social network. However, more research on how to use TVGs for real-world applications as well as how to enrich TVGs with attributes on both nodes and edges is required since it potentially can help to capture diverse temporal information of the real-world user behaviour on the Web.

User behaviour on the Web is multifaceted and dynamic. In order to understand patterns of user behaviour, explore a model of user behaviour on the Web, a set of techniques has been proposed and developed. One of the most powerful tools to explore user behaviour on the Web is visualisation. Visual exploration of user behaviour on the Web is introduced and discussed in the next Section.

2.4 Visual Data Exploration

User behaviour on the Web involves communication via e-mails, social networks, chats, forums, and instant messengers. However, the complexity and scale of such user behaviour motivates people to find ways how to explore this communication. Visualisation is a powerful exploration tool. There is a large number of researches on how to visually represent user communication on the Web and understanding which visualisation is more efficient for a specific reasoning task [Venolia and Neustaedter, 2003; Smith and Fiore, 2001; Jovicic, 2000; Sathiyanarayanan and Burlutskiy, 2015a,b]. In addition to the research work on visualisation, there are many commercially available visualisation tools for different data produced by users [Google, 2014; IBM, 2014; TIBCO, 2014].

The field of information visualisation has seen a rapid growth due to advances in software and hardware development and, as a result, a number of software visualisation tools have appeared on the market. These tools allow users to create a complicated visualisation on the fly by a single user's click. Moreover, a user can interact with visualised data and share it with others [Google, 2014; IBM, 2014; TIBCO, 2014]. For instance, data in *IBM Many Eyes* can be uploaded in text format or tab-delimited data and then a user can choose a visualisation type to explore the data.

To sum up, visualisations facilitate data exploration and assist in further transformation of discovered knowledge into a meaningful representation [Burlutskiy et al., 2014a]. Even more, visualisations help to discover and present patterns in social networks. For instance, in [Nohuddin et al., 2015] the authors reported a technique for discovering and presenting patterns in temporal social network data.

Nevertheless, due to the large number of possible visualisations, choosing an appropriate visualisation is a challenging task. This choice significantly depends on user's task. A list of exploration tasks and methods for choosing a visualisation are presented in the next Subsections.

2.4.1 Exploration Tasks

The first step in data exploration is to analyse the data and check its quality. Visualisation can assist a user in verification for existence of *missing values*, *errors*, and *outliers*. The data can have *missing values*, for example, some instances can have missing timestamps – the time points when the instances were generated. *Errors*, for example, can include typographical errors. Finally, *outliers* are observations which are sufficiently extreme that they can be taken as being caused by some undesired factor, for example, an error in recording or in observation [Gilchrist, 1984].

Visualisation can help to understand user behaviour on the Web and the characteristics of such behaviour. For example, visualisation allows one to explore the following:

- *Understand Temporal Granularity.* User behaviour exists at different temporal granularity levels. The microscopic behaviours happen in a short time frame. For example, a user can communicate via instant messages with another user in fractions of seconds. On the other hand, a user can exhibit behavioural patterns over years, for example, change in political views or a circle of close friends.

- *Find parameters for Feature Tuning.* Some temporal features, for example, aggregated temporal features, require to specify time intervals. Visual exploration can help to determine and verify such intervals.

- *Reduce Data Size.* User behaviour generates large data and, as a result, it is crucial to reduce the size of the data without much losing the information required for a particular task. Visual exploration can help to select and leave only important data. Also, a set of local and aggregated features can be determined and then formed via visual exploration of the data.

- *Explore Parameters of a Graph Model.* Visualisation of a temporal graph representing user behaviour on the Web can help to choose a proper aggregation level for temporal features.Visualisation can help to choose features for building a predictive model of user behaviour. These facts have caused many tools for graph visualisation to appear.

2.4.2 Choosing a Visualisation

A traditional way for choosing a visualisation is usually performed in accordance with user's experience in visualising data. Another approach is to use a guideline [Keim, 2002] or a recommender system for choosing a visualisation [Vartak et al., 2015; Kaur et al., 2015].

For example, in [Keim, 2002] the author classified visualisation techniques into five groups, namely (1) standard 2D and 3D graphics, (3) geometric techniques, (2) icon-based techniques, (4) dense pixel techniques, and (5) hierarchical techniques. The author provides with a guideline for a visualisation expert based on the introduced classification.

From another side, development of visualisation recommender systems is an area of active research. A combination of user's dataset and a chosen visualisation type can be considered as an experience of a user in visualising their data. Nevertheless, choosing a visualisation for user's data is a challenging task. As a result, there is research in methods for providing a user with a recommendation of a visualisation for their data. For example, in [Freyne and Smyth, 2010] the authors proposed a case-based recommendation system that is capable of suggesting popular visualisations to users based on the characteristics of their datasets and users' preferable visualisations in IBM Many Eyes [IBM, 2014; Freyne and Smyth, 2010]. Nevertheless, the approach did not address the problem of visualising user behaviour on the Web.

A broader vision on automatic identification and visualisation recommendation for a particular analytical task is discussed in [Vartak et al., 2015]. This work extends the work presented in Chapter 3 where an approach for automatically choosing visualisations is proposed. Another work on developing a visualisation recommender system was proposed in [Kaur et al., 2015]. The authors introduced a semi-automatic visualisation recommendation based on captured domain knowledge. However, according to the literature explored, there are no methods proposed for choosing a visualisation of user behaviour on the Web.

2.5 Extracting Features for Prediction

Predicting user behaviour on the Web involves building predictive models of user behaviour, for example, from historical data. In this thesis, Machine Learning (ML) is employed for building such models. One of the requirements for such models is being able to produce an accurate prediction. Feature engineering is an important aspect of an accurate prediction and it can significantly improve the accuracy of a prediction as well as reduce the time for training and prediction [Page et al., 2014; Hira and Gillies, 2015]. However, there are two major problems related to feature engineering. First, how to construct and calculate the features. For example, an automatic process of calculating features like in deep learning can be utilised or an approach with hand-crafted features can be constructed using knowledge of domain specialists. Second, once a feature set is constructed and calculated, there are scenarios when the features are computationally complex and it will take unacceptably long time and effort to calculate the features. Thus, there is a trade-off problem of spending as little as possible time on feature engineering and computation while including the most valuable features in the final feature set.

Indeed, improving accuracy of a prediction is very important for any ML task. However, from an engineering point of view, reducing the number of features, complexity of features and ML algorithms, as well as improving reproducibility and stability of a system is important as well [Sculley et al., 2015].

Next, the process of identification of features for predicting user behaviour is discussed below.

2.5.1 Identifying Features

There is a large number of possible ways to construct and calculate features. Some of these features are important for producing an accurate prediction and some of them are useless. In this Subsection, the features which showed being important in various prediction tasks of user behaviour are introduced.

A set of features important for *Twitter* trend prediction was analysed in [Zhang et al., 2014]. The authors used three types of features, namely *content*, *structure*, and *node* features. The content features were extracted from tweets, considering features such as, for example, the number of tweets. The structure features were related to the metrics on the topological structure of the network. Finally, the node features were associated with the information on users. This thesis considered to classify features in according to the model of user behaviour as a graph. However, the feature design was not discussed thoroughly. Also, the temporal aspect of the model describing trends in user behaviour was omitted in their paper.

In [Kim and Leskovec, 2011] it was shown that the users with similar attributes are likely to link to one another which allows one to build a model for predicting users' attributes. In this paper, it was identified that the features related to users are important for such predictions. The model was built using a public dataset *AddHealth* where the information on users include their school name, school type, grades, and some demographic characteristics [Kim and Leskovec, 2011].

In [Radinsky et al., 2012], the authors concentrated on using temporal features for predicting user's activities of clicking urls. These features included aggregated features of time series, features describing the shape of time series, and other domain-specific features, for example, query class features. Indeed, predicting user's activities frequently involves dealing with large amounts of time series. As a result, there is a need for aggregating the features over time.

A representative model for predicting Telecom churn was introduced in [Sharma and Panigrahi, 2013]. The features used in the model were related to *users*, for example, user's phone number, and user's area code. Other features were related to *calls*, for example, the number of user's night calls, and user's evening calls. Finally, the features related to *charges* were considered, such as the total day charge feature, and international calls charge feature. The features related to user's calls and user's charges were split in accordance with the temporal properties of the calls, whether the call happened during the day, evening, or night. Even using these simple features the authors reached relatively high accuracy. The *calls* can be considered as messages exchanged between the users and the charges can be seen as properties or attributes of these messages. As a result, these features can be viewed from a more general prospective where users exchange messages and the features can be associated to users or messages (calls) and the messages have attributes (charges). Also, the authors used temporal information for splitting the features into day, night, or evening features.

Some authors tried to improve predicting accuracy by introducing domain-specific features. For example, in [Comarela et al., 2012; Teevan et al., 2011] the authors predicted response time in Q&A forums by constructing the features representing question difficulty, how well the question is formulated, if there are experts on the forum who can answer the question, whether the experts are willing to answer and have time to answer, whether the question is interesting, and whether it was asked at night, weekend or holiday [Comarela et al., 2012; Teevan et al., 2011]. However, the problem with this type of feature is the fact that they are hard to generalise to different domains, as well as the fact that they require to be hard-coded. As a result, there is a need for a more general guideline on how to construct and extract features.

Twitter, as one of the largest social platforms, has attracted many researchers who tried to build a few predictive models. As a result, researchers tried to identify features which can be extracted from *Twitter* data and which features are important for predictions. For example, in [Spiro et al., 2012] the authors predicted the response time of tweets. It was shown that some *Twitter* hashtags are strongly correlated with smaller waiting times. On the contrary, the authors claimed that the number of user's followers was associated with longer waiting time for a retweet. A very similar prediction model was constructed in [Mahmud et al., 2013; Lee et al., 2014]. The authors constructed features based on the past history of user's retweeting wait times in order to predict their next retweeting time for a new tweet. The past history included features such as the number of retweets per status message, an average number of retweets per day, and tweeting statistics for the last days and hours. In [Artzi et al., 2012], a method for the prediction of whether a tweet will be retweeted, replied to or ignored was proposed. The authors used six types of features, such as historical, social, lexical, content, posting, and sentiment features. These works [Artzi et al., 2012; Spiro et al., 2012; Mahmud et al., 2013; Lee et al., 2014] motivate including a diverse set of features for constructing a predictive model in this thesis. Nevertheless, due to the difference in various types of social media, such as Q&A communities, *Facebook* and *Twitter* communities, these features are generalised in the proposed methodology for predicting user behaviour on the Web.

Most of the prediction models ignore temporal and dynamic natures of user behaviour, but there is evidence that temporal characteristics of social media communities are important [Cai and Chakravarthy, 2013]. An attempt to use temporal features to predict the quality of answers at Q&A forums was undertaken in [Cai and Chakravarthy, 2013]. The authors introduced the following temporal features: 1) the number of answers, 2) the number of best answers, 3) the number of questions asked or given by a user, and 4) the best answer ratio during a time interval. The authors argued in their work that these temporal features are crucial for predicting the quality of answers. Also, in [Guille and Hacid, 2012], it was shown that temporal features were crucial for social network prediction tasks on *Twitter*. These works inspired to design and introduce temporal features in this thesis as a part of the methodology for predicting user behaviour on the Web.

Another study on the factors that influence the response time on *Twitter* was conducted in [Comarela et al., 2012]. As a part of this study, factors that influence users' response or retweet probability were identified. The authors found that these factors include previous response times to the same tweeter, the sending rate of the tweeter, the age of the tweet message and some basic text elements of it, such as presence of a hashtag, mention of someone, and embedded url links. In [Teevan et al., 2011] the factors influencing response time for questions asked on *Facebook* were investigated. It was discovered, firstly, that phrasing a question well leads to better responses from users. Secondly, explicitly stating a question in the status message as opposed to a statement message, increases the chances of response. Thirdly, explicitly scoping the audience and, finally, using only one sentence led to more, better, and faster responses. A study on whether a question asked at a Q&A community requires a better answer was undertaken in [Anderson et al., 2012]. The authors selected eighteen attributes for learning and then predicted whether an answer satisfied the questioner or not. It was shown that the selected attributes by the authors were well chosen predictors for the quality evaluation of answers. In

[Hu et al., 2013], the large number of questions on broad topics at *Baidu Zhidao* Q&A forums motivated the authors to exploit *social* features and topic based features of the forums for answer quality prediction. As a result, it was found that the answer length, answer position, and user's activity can be used for distinguishing high quality answers from low quality answers. These papers helped in selecting the features used in the proposed methodology and then generalising and automating the process of extracting features from a model of user behaviour on the Web.

Since human communication and interaction can be modelled as a graph, features related to the topological properties of the formed graph can be used for prediction. Even more, it was shown that these topological or so-called structural features of human communication can improve the accuracy of prediction tasks [Santoro et al., 2011]. In their paper, the authors showed how to calculate some structural features of time-varying graphs. In another paper [Tang et al., 2009], a few temporal distance metrics for capturing temporal characteristics of human communication represented as a time-varying graph were introduced. These metrics can be used as features for predicting user behaviour on the Web. However, one can notice that calculating such features, especially for large graphs, is a computationally expensive task.

Predicting which users in a communication network will either increase or decrease their activity after a given period of time was demonstrated and evaluated in [Teinemaa et al., 2015]. The features related to usage of network (chat days, audio days, video days), structural features (number of edges, average total degree, internal density of the social network formed by a community), and profile features (number of countries, cities in the community, gender, age of users) were extracted for building a predictive model. Again, structural features showed high importance for the prediction.

A model for predicting how users are connected in online location-based social networks was proposed in [Scellato et al., 2011]. The authors classified the features into *social* features computed for friends-of-friends, *place* features computed for place-friends, and *global* features. Place features include, for example, the number of check-ins, the number and the fraction of common places between two users. Social features defined for every pair of users are the number of common neighbours, Jaccard coefficient, and Adamic-Adar measure based on the degrees of the shared neighbours [Scellato et al., 2011]. Finally, global features, for example, the geographic distance between users' home locations, the same distance divided by the product of the number of check-ins each user made in their home location, was demonstrated to be a good predictor. This paper showed that spatial features are important for producing location-based predictions.

In [Bhat et al., 2014], the authors showed that tag-related features strongly influence response time for the questions asked in *Stack Overflow*. The so-called tag-related features are the following: the average frequency of tags, number of popular tags, average co-occurrence rate of tags, number of active subscribers, percentage of active subscribers, number of responsive subscribers, percentage of responsive subscribers. As a result, the authors showed that in their experiment set-up the tag-related features outperformed features not related with the tags. However, the authors did not address the computational complexity of calculating such features.

Identifying the features for building a predictive model is one side of predicting user behaviour on the Web. Another side is to calculate the features which can be challenging due to high computational complexity of the identified features. The next Subsection concerns the computational complexity involved in calculating and transforming features before building a predictive model.

2.5.2 Complexity of Feature Extraction

Identifying the features which significantly influence the success of a ML task is one side of most ML tasks. However, computing the identified features may involve several transformations and mathematical calculations which can negatively affect the performance of extracting features, training a model and then predicting user behaviour on the Web using this model. The complexity of calculating such features can be analysed using Big O notation [Knuth, 1976]. First of all, several transformations of feature space are introduced. For example, these transformations can be the following:

- *Standardisation*: features can represent comparable objects but be measured in different units. For instance, two different features can represent the same measurement but the first one is in seconds and the second one is in minutes. Standardisation prevents this from happening, for example, by the following centring and scaling transformation of features: $x'_i = (x_i - \mu_i)/\sigma_i$, where μ_i is the mean and σ_i is the standard deviation of the feature x_i over all training samples;
- *Normalisation*: to remove the dependence of a feature x_i on the size of the feature, the following transformation can be done: $x'_i = x_i/||x_i||$;
- *Binarisation*: categorical features must be represented as multiple boolean features. For example, instead of having one categorical feature for the weather, introduce three binary features *sunny*, *rain*, and *snow* and then set these features to 0 or 1. As a result, every categorical feature will be transformed

into k binary features where k is the number of values that the categorical feature takes;

- *Non-linear expansions*: Even though for complex data dimensionality reduction is crucial, increasing the dimensionality of a problem can be advantageous. For example, in case when the problem is very complex and the available features are not enough to derive good results in terms of the accuracy of prediction;
- *Aggregation*: Several features can be aggregated. For example, a set of features can be substituted with only one feature which has the value which is an average of the values of the set of features;
- *Feature discretisation*: some ML algorithms do no handle well continuous data. Thus, the data must be discretised into a finite discrete set. Even more, this step can simplify the data description and improve understanding of data.

Some transformations change the dimensionality of a problem while others do not. For example, *standardisation* and *normalisation* do not change the dimensionality of a problem whereas *aggregation* decreases the problem dimensionality.

2.5.3 Selecting Features

The main goal of feature selection is to select relevant and informative features. However, feature selection can achieve other goals, for example [Guyon and Elisseeff, 2006]:

- *To decrease the size of the data*: to limit storage requirements and increase algorithm speed;
- *To reduce feature set*: to save resources in the next round of data collection or during utilisation;
- *To improve performance*: to gain in predictive accuracy;
- *To understand the data*: to gain knowledge about the data or simply visualise the data.

Ideally, an exhaustive leave-one-feature-out selection process should be executed on a regular basis. This will help to keep the size of a feature set efficient by identifying useless features and removing them from the feature set. There are many methods for feature selection. The methods can be classified into *filter methods*, *wrapper methods*, *embedded methods*, and *hybrid methods*. [Bolón-Canedo et al., 2012]:

- *Filter methods* rank features in accordance with some measure of 'usefulness' of each feature for classification. Then a feature set composing of the best N features in accordance with this ranking is created.

- *Wrapper methods* 'wrap' a classifier up in a feature selection algorithm. This type of methods usually chooses a set of features and then the efficacy of this set is evaluated. Then the original set is changed in accordance with some rules and then the efficacy of the new set is evaluated. The problem with this approach is that feature space is vast and looking at every possible combination is computationally expensive.

- *Embedded methods*: In contrast to *filter* and *wrapper* approaches, *embedded methods* do not separate the learning from the feature selection part.

- *Hybrid methods* are a combination of the previous methods.

Choosing a method for feature selection depends on the size of the data, the total number of features, learning algorithms, and algorithm computational complexity [Kojadinovic and Wottka, 2000; John et al., 1994]. For example, it is challenging to use computationally intensive learning algorithms with wrapper methods.

The next Section introduces state-of-the-art ML algorithms for predicting user behaviour on the Web. Also, two popular learning modes are discussed, online and offline learning as well as the idea behind deep learning, one of the most successful types of learning for several domains, is introduced.

2.6 Machine Learning for Predicting User Behaviour

Machine Learning (ML) is an area of research where approaches for enabling computers to learn from data without explicitly programming the computers are developed. ML approaches have shown promising results in predicting user behaviour on the Web [Sadilek and Krumm, 2012; Zheng et al., 2013; Nazerfard and Cook, 2013; Choi et al., 2013; Zhu et al., 2013; Burlutskiy et al., 2015]. One direction in creating models for predicting user behaviour is to build a high-level model of information flows on the Web [Spiro et al., 2012; Mahmud et al., 2013]. Indeed, building high-level statistics on how data propagates over time is a useful method in understanding user behaviour on the Web. Another direction is to use various features of data for a more detailed picture of user behaviour. The former direction gives only a very coarse insight in people communication and relationship development. On the contrary, the feature-based approach for developing a model at a lower granularity level can potentially provide a more precise model [Zhu et al., 2013; Zheng et al., 2013; Nazerfard and Cook, 2013; Choi et al., 2013; Burlutskiy et al., 2015].

Formulating the problem of predicting user behaviour in the terminology of ML can be performed naturally and in a straightforward way. More specifically the problem can be formulated as a classification task. In this case, the goal is to predict the outcome Y for new 'test' samples of user behaviour. For this purpose, one must build a predictive model M which is typically a function with adjustable parameters. The training examples X are used to select an optimum set of parameters:

Definition 1. *Let X be a set of input variables x_i and Y be a label with classes c_i. Then define D_{tr} as a training set of instances (x_i, c_i), $D_{tr} = \{(x_i, c_i)\}$, $x_i \in X$, $c_i \in Y$ and $1 \leq i \leq n$, where n is the total number of instances. The classification problem is to determine a model $M(X, Y)$ such that maps x_i to target classes c_i.*

The process of training and testing the model is preceded by dividing data D in two sets: training dataset D_{tr} and test dataset D_{td}. The training dataset D_{tr} is used for training the classifier whereas the test dataset D_{td} is used for the actual prediction. The data D can be split in the training and test datasets in different proportions, for example, 80% of the data D becomes the training dataset and other 20% is a test dataset. The variable for prediction Y is considered known in the training dataset D_{tr} but unknown in the test dataset D_{td}; thus, the variable for prediction Y is predicted by the trained model $M(X, Y)$ on a test dataset D_{td}.

Predicting user behaviour in real-time implies the following requirements for a prediction algorithm. First, since the dimension of the feature space is high, there should be a feature selection step. Usually there is a large number of features available: user information, location, messages, time of day, day of week, activities of users. Second, the algorithm should calculate and provide the prediction fast enough to support decision-making. Last, the algorithm should have a continuous learning nature by which it keeps learning over time. In other words, the algorithm has to constantly use the historical data for learning and adapt its predictions to the dynamics of human behaviour.

Supervised ML algorithms showed promising results for these requirements [Sadilek and Krumm, 2012; Zheng et al., 2013; Nazerfard and Cook, 2013; Choi et al., 2013; Zhu et al., 2013; Burlutskiy et al., 2015]. As a result, a variety of ML approaches have been used for predicting user behaviour. Generally speaking, advance ML algorithms along with intelligent feature engineering tend to demonstrate higher accuracy but the complexity of such models and features negatively affects their time performance [Yang et al., 2011; Dror et al., 2013]. However, analysis and evaluation of the time needed for training and predicting user behaviour is often omitted [Choi et al., 2013; Radinsky et al., 2012].

2.6.1 Machine Learning Algorithms for Supervised Learning

In supervised learning, training of a classification model is performed with known labels Y. For example, a binary classification task assumes that the label has only two classes $Y = \{0, 1\}$.

In [Yang et al., 2011] an approach for predicting if a question will be answered or not was proposed. First, the unanswered questions at *Yahoo! Answers* were analysed and then a prediction model was proposed by formulating the problem as a supervised learning task. As a result, the authors demonstrated how to use ML for predicting whether a question will remain unanswered.

Customer restaurants preference was predicted based on check-ins posted in social media [Zheng et al., 2013]. The authors analysed 121,000 Foursquare check-ins in restaurants in the Greater New York City area and then they built two models using *Support Vector Machines* (SVM) and *Artificial Neural Networks* (ANN) to predict customer behaviour. As a result, the authors showed that an model based on ANN provides with a quite accurate prediction compared to a SVM-based model. However, it was not mentioned how computationally expensive it was to train an accurate model which can be unacceptably hard and long for near real-time prediction requirements.

A representative model of predicting Telecom churn was presented in [Sharma and Panigrahi, 2013]. The authors used an ANN, namely Multi-Layer Perceptron (MLP), and the Radial Basis Function Network (RBFN) for solving a binary classification problem.

The authors in [Zhu et al., 2013] built a predictive model which was trained on a set of extracted features, such as user activity, social connections, and temporal features. The authors proposed using a modified Logistic Regression (LR), the efficiency of which they compared to a classic LR, *Random Forests* (RF), and *Node* classification algorithms [Ma et al., 2011]. A similar prediction was accomplished in [Weerkamp and De Rijke, 2012] where a novel approach of activity prediction was proposed. The problem formulated by the authors of [Weerkamp and De Rijke, 2012] was "given a set of tweets and a future timeframe, to extract a set of activities that will be popular during that timeframe". The authors focused on aggregated behaviour rather than individual behaviour as in the paper of [Zhu et al., 2013]. However, in these papers the authors did not address the problem how to build a *fast* and accurate model for predicting user behaviour.

Trade-off between accuracy and time in the context of wireless network was investigated in [Loumiotis et al., 2014]. In their paper, the forthcoming network traffic demand was predicted using an ANN-based model; such predictions require a regular training process, which incurs a high computational cost. In this thesis, the trade-off in the different context of user behaviour on the Web is investigated.

In [Ding et al., 2016] the authors proposed a graph-based ML approach for predicting hidden or unknown attributes of users by using the known information provided by users in a social network. The authors tried to use both topological properties of the social network as well as the known attributes of users for predicting the unknown or hidden attributes of users. The authors considered the nodes of their graph representing users of the network. Also, the nodes had attributes, for example, age, gender, university, and hobbies of the users. As a result, the authors demonstrated that it is possible to infer users' hobby, age, and users' university using the proposed model.

The choice of a ML algorithm is influenced by the results presented in the literature review [Kotsiantis, 2007] where a summary of state-of-art algorithms for classification tasks is presented, as well as papers where ML algorithms showed evidence of usefulness in predicting user behaviour on the Web (see Table 2.1[2]). In addition, the specificity of the analysed data influences the choice of features. LR proves to be useful in solving predictive tasks due to scalability and transparency of results interpretation. Practically, other more technically demanding approaches such as SVM and ANN can show higher accuracy in prediction but they are much more computationally expensive and hard in interpreting the results of prediction. Ensemble ML approaches, for example, Boosting approaches, can improve the accuracy but require extra computations.

Five diverse ML algorithms, namely LR, SVM, DT, k-NN, and XGB algorithms, are described below in more detail since these algorithms are chosen as state-of-the-art ML algorithms for experiments in this thesis.

[2]Most of these machine learning approaches are standard and can be found in a textbook on Machine Learning, for example [Murphy, 2012]

Abbr.	Full name	Description	Reference
LR	Logistic Regression	A probabilistic binary statistical classification model; can have linear or non-linear kernel, for example, a radial kernel	[Zhu et al., 2013; Cheng et al., 2011; Bhat et al., 2014; Lee et al., 2014; Hu et al., 2013]
ME	Maximum Entropy	Multi-class generalisation of LR	[Artzi et al., 2012; Hu et al., 2013]
MLE	Maximum Likelihood Estimation	A method used for parameter estimation in statistical models	[Spiro et al., 2012]
k-NN	k Nearest Neighbours	An instance based learning algorithm choosing k nearest instances	[Raikwal et al., 2013; Goel and Batra, 2009]
SVM	Support Vector Machines	A non-probabilistic binary linear classification model	[Jernite et al., 2013; Bhat et al., 2014; Lezina and Kuznetsov, 2012; Comarela et al., 2012; Hu et al., 2013]
NB	Naive Bayes	A probabilistic classifier based on Bayes' Theorem	[Yang et al., 2011; Scellato et al., 2011; Comarela et al., 2012; Lee et al., 2014]
BN	Bayesan Networks	A family of probabilistic classifiers	[Nazerfard and Cook, 2013]
DT	Decision Tree	A probabilistic graph-based classification model; C4.5 and J48 are some of implementations of the method	[Yang et al., 2011; Scellato et al., 2011; Cheng et al., 2011; Bhat et al., 2014]
RF	Random Forest	A probabilistic graph-based classification model similar to DT but with a correction to avoid overfitting of training data	[Zhu et al., 2013; Scellato et al., 2011; Larivière and Van den Poel, 2005; Lezina and Kuznetsov, 2012; Lee et al., 2014; Teinemaa et al., 2015]

MART	Multiple Additive Regression-Trees	A regression classifier with gradient boosting	[Artzi et al., 2012]
ANN	Artificial Neural Networks	Algorithms inspired by the way a human brain works	[Zheng et al., 2013; Sharma and Panigrahi, 2013; Choi et al., 2013]
EM	Ensemble methods	Approaches using multiple ML algorithms to achieve better performance	[Yang et al., 2011; Lee et al., 2014]

Table 2.1: List of some approaches used for predicting user behaviour.

Logistic Regression (LR)

Logistic regression for a binary classification can be described by the following equation:

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \tag{2.2}$$

where $p(x)$ is the probability of a variable Y to be '1' and $1-p(x)$ is the probability of Y to be '0'. The variable $x = (x_1, ..., x_n)$ is the variable of features used for prediction of Y. The coefficients $\{\beta_i\}$ are the coefficients of logistic regression and they show the contribution of each x_i to the prediction of the variable Y. Usually, it is predicted that $Y = 1$ if $p(x) \geq 0.5$, and $Y = 0$ otherwise.

An implementation of LR exists in most common ML software tools, for example SAS[3], SPSS[4], scikit-learn[5], Matlab[6]. The main advantages of LR are the ability to interpret the built model, and the performance for large datasets with a multidimensional feature space.

Support Vector Machines (SVM)

SVM is a non-probabilistic classifier which builds a model that assigns labels for prediction into one category or the other. As a result, SVM builds a boundary between labels in n-dimensional space where n is the number of features.

[3]www.sas.com
[4]www.ibm.com/analytics/us/en/technology/spss/
[5]www.scikit-learn.org/stable/
[6]www.mathworks.com/products/matlab/

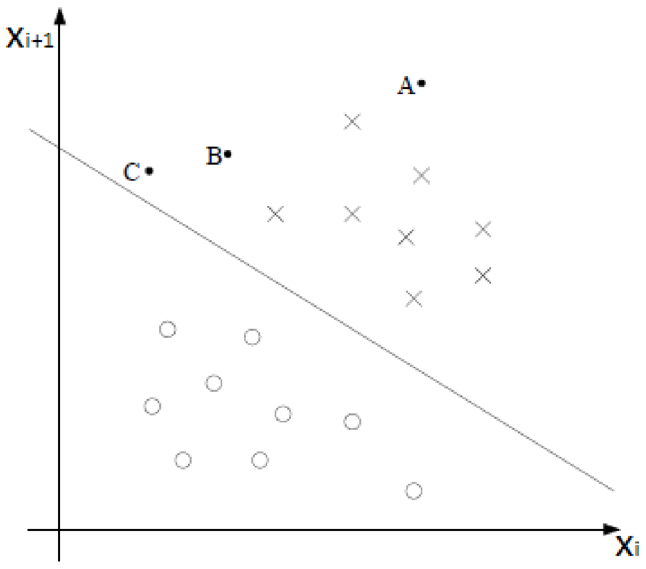


Figure 2.5: Building a boundary between classes using an SVM approach.

For example, in the case of using a binary SVM classifier, the boundary between the two classes, circles and crosses, is determined after training the classifier (see Figure 2.5). If a new n-dimensional point for classification is above the boundary, it is classified as a 'cross' and as a 'circle' otherwise. For example, A, B, and C are classified as 'crosses' since they are above the boundary.

Analogously to LR, an implementation of SVM exists in most common ML software tools. SVM maximises a margin between classes rather than the likelihood of a particular class is maximised as it is performed in the case of LR. However, there is no evidence as to whether or not SVM can achieve better accuracy of prediction than LR can achieve.

k Nearest Neighbours (k-NN)

The idea of k-NN algorithm is to choose k neighbouring vectors for an input vector x as the most similar to the input vector. For choosing a neighbouring vector of the input vector, a distance metrics between the data vectors has to be defined. For example Euclidean metric can be used. The next step is to define the similarity measure for comparison of the vectors. For finding the nearest neighbours, this similarity measure is used. In case of Euclidean space, a similarity measure $S(p, q)$

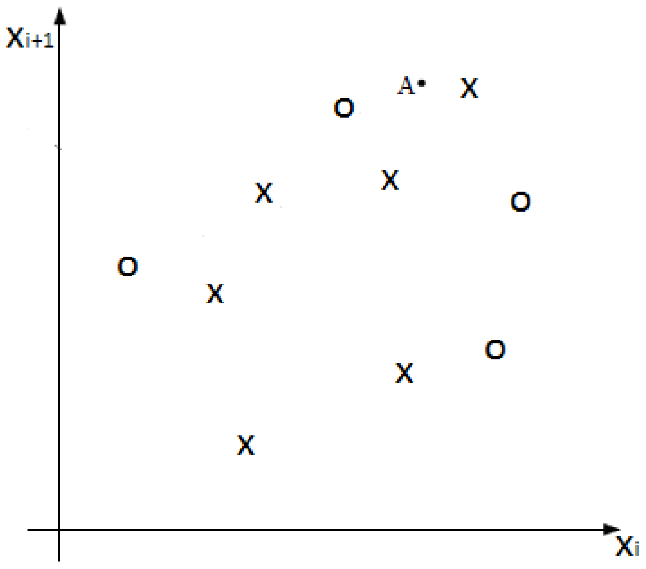


Figure 2.6: Classification using a k-NN approach.

between two vectors p and q is considered to be an Euclidean distance:

$$S(p,q) = \sum_{i=1}^{n} \sqrt{\omega_i (p_i - q_i)^2} \tag{2.3}$$

where p and q are two vectors, p_i and q_i are the i^{th} entries of the vectors, n is the number of entries; $\omega = \{\omega_i\}$ is a vector of weights corresponding to the importance of each entry in the prediction. Each vector p_i can be considered as a point in n-dimensional Euclidean space.

The value of the variable for prediction Y is determined by the majority class of the nearest k neighbours to the n-dimensional point of interest. For example, if $k = 5$ and three the nearest neighbours are of a class 'crosses', or '1', whether two others are of a class 'circles', or '0', then the variable $Y = 1$ since the majority of the neighbours are of the class 'crosses' (see Figure 2.6).

Decision Trees (DT)

Decision trees (DT) are tree-like structures which can be used for classification. In these structures, *leaves* represent class labels and *branches* represent conjunctions of features that lead to these class labels. DT with continuous target variables are called regression trees.

An example of a decision tree is shown in Figure 2.7. This DT has four

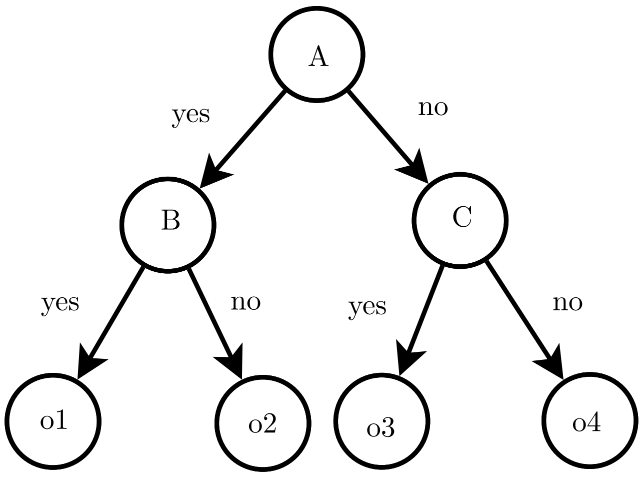


Figure 2.7: An example of a decision tree. If the condition A is true then the condition B is checked, the condition C is checked otherwise. Finally, if condition B holds then the outcome o_1 comes out, otherwise the outcome o_2. For the condition C there are the outcomes o_3 and o_4 correspondingly.

classes $\{o_1, o_2, o_3, o_4\}$ and three binary conjunctions $\{A, B, C\}$.

Again, most modern ML software tools have an implementation of an algorithm to construct a DT. There are a few such implementations including ID3 [Quinlan, 1986], C4.5 [Quinlan, 1993], and J48, an open source implementation of C4.5 algorithm.

Extreme Gradient Boosting (XGB)

Ensemble ML methods is a family of ML algorithms where a combination of ML models is used to build a meta model. One of such algorithms which has shown promising results, for example at Kaggle[7], a popular platform for ML competitions, is an ensemble regression trees proposed in [Friedman, 2000]. Tree boosting is a highly effective and widely used ML method which is a state-of-the-art algorithm due to its performance and success in various ML challenges. More details on extreme tree boosting using gradient descent are in the paper [Chen and Guestrin, 2016]. The authors of this paper developed one of the most popular library using such tree boosting concept, Extreme Gradient Boosting (XGB).

2.6.2 Online and Offline (Batch) Learning Modes

Online prediction is a hot research topic due to the need of the Big Data world to provide fast and accurate results. For example, predicting human activity for the

[7]www.Kaggle.com

next few seconds or minutes leaves no time for long batch training unless a model is very simple. In [Choi et al., 2013], the authors compared the accuracy of prediction using online, mini-batch, and batch training. However, they did not evaluate the time taken for training their models and then the prediction time.

There is no common ground whether online learning is as fast or faster than offline learning. Even more, there is a debate whether batch training provides more accurate results than online training [Wilson and Martinez, 2003]. Comparing these two learning modes will facilitate choosing which learning mode as well which learning algorithm is advantageous for deployment in a real world system. The fact that comparing online learning is challenging and depends on a particular prediction task, motivates the development of a method for comparing online and offline learning in this thesis.

2.6.3 Deep Learning

Deep learning is an automated approach in learning *data representations* and *features.* Theoretically, deep learning networks can learn a powerful representation of a problem space. Figure 2.8 shows the difference between a *shallow* and a *deep* neural network. A deep learning network consists of an input layer, an output layer, and a few hidden layers whereas a shallow learning network has an input layer, an output layer, and only one hidden layer. Ideally, every farther hidden layer in a deep neural network learns how to represent more complex features of the data. Indeed, theoretically the more the layers a neural network has, the more complicated functions it can represent. Thus, deep learning networks are advantageous for learning data representation in comparison to shallow learning networks.

There are three the most popular types of deep learning networks which showed promising results in several application:

- *Deep Belief Networks (DBN)*, a type of deep neural network, where multiple layers are interconnected but there is no connection between units within each layer [Hinton et al., 2006]. These networks are generative models of deep neural networks and have been used, for example, for predicting user behaviour in smart home environment [Choi et al., 2013];
- *Convolutional Neural Networks (CNN)*, a type of deep neural network where the connectivity architecture across its units is inspired by the organisation of the animal visual cortex. Currently state-of-the art approaches for image classification and recognition and Natural Language Processing (NLP) are based on CNNs [Ciresan et al., 2012; Gu et al., 2015];

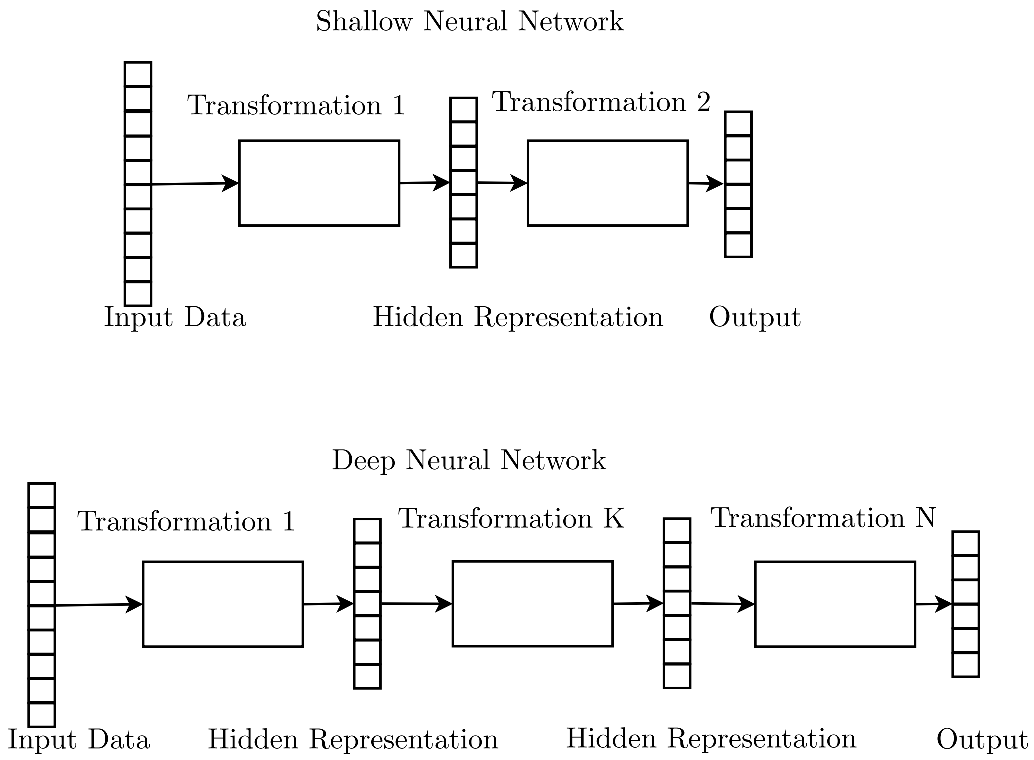


Figure 2.8: The difference between a shallow neural network and a deep neural network.

- *Recurrent Neural Networks (RNN)*, a type of deep neural network where units are interconnected and form a directed cycle. These networks showed promising results, for example, in learning sequences in images and speech [Lipton, 2015].

Traditionally, back-propagation was used to train such learning networks. However, training neural networks is computationally challenging thus, making them not useful for many practical purposes. Nevertheless, recently there was a break through in deep learning caused, first, by technological advances in parallelising these algorithms across multiple machines, CPUs and GPUs, and, second, due to proposals of a few computationally more efficient methods for training such networks. For example, a fast, greedy learning algorithm for training a DBN was proposed in [Hinton et al., 2006]. As a result, training multi-layer neural networks in several applications has been revisited.

Nevertheless, it is still challenging to apply deep learning networks to large scale applications since training and testing a deep learning network is a time-consuming and computationally expensive task. In [Ruangkanokmas et al., 2016] the authors propose using a DBN with a feature selection approach in combination. In their approach a chi-squared based feature selection allows to decrease the complexity of the feature input by eliminating irrelevant features. As a result, the learning phase of DBN becomes more efficient. The experimental results showed that such feature selection allows one to train a DBN-based model which is capable to provide higher accuracy compared to a LR-based model or a DBN-based model without a feature selection stage. These results motivate the investigation of how to train and use a DBN-based model in combination with a feature selection approach, for example a wrapper method, for predicting user behaviour on the Web.

Finally, after building a predictive model, the performance of the model has to be evaluated. The next Subsection addresses this problem.

2.6.4 Evaluating the Efficiency of Prediction

In order to evaluate the performance of a model $M(X, Y)$, a test set D_{td} and an evaluation function E_p are introduced:

Definition 2. *Let a test set be defined as $D_{td} = \{(x_j, y_j)\}$ where $x_j \in X$, $y_j \in Y$, $1 \leq j \leq m$ where X is a variable, Y is a variable for prediction, and m is the number of data entries. There are no common elements in sets D_{tr} and D_{td}: $D_{tr} \cap D_{td} = \emptyset$ where D_{tr} is a training set. The set D_{td} is used to evaluate the performance of the model $M(X, Y)$. Such evaluation can be performed by comparing the values of variables $y_j \in Y$ against predicted variables $\hat{y_j}$ for x_j for all $(x_j, y_j) \in D_{td}$. The evaluation function E_p summarises the performance of the built model $M(X, Y)$ on the test set D_{td} numerically.*

Performance measures are typically specialised to the class of ML problem. In the case of *classification*, accuracy measures are widely used for evaluation of model performance.

Performance Measures

The total number of correct classifications N_c divided by the total number of predictions N is one performance measure:

$$E_p = \frac{N_c}{N} \tag{2.4}$$

Each possible outcome of the classification are the following:

- *False positives (FP)*: the number of examples predicted as positive, which are from the negative class.

- *False negatives (FN)*: the number of examples predicted as negative, whose true class is positive.
- *True positives (TP)*: the number of examples correctly predicted as pertaining to the positive class.
- *True negatives (TN)*: the number of examples correctly predicted as belonging to the negative class.

Some standard accuracy measures calculated using the aforementioned four outcomes are $Precision$, $Recall$, and $F1$. $Precision$ can be calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

$Recall$ is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

$F1$ Score is:

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

Equation 2.4 can be rewritten as:

$$E_p = \frac{TN + TP}{TP + TN + FP + FN} \tag{2.8}$$

Cross-Validation

Cross-validation can be employed for evaluation of the accuracy of a prediction. For example, in n-fold cross-validation, first, the data is divided in n parts (see Figure 2.9 for $n = 5$) and then n models are built on $n - 1$ parts of the data consequently. Finally each model is tested on the left, n^{th}, part of the data. For example, for 5-fold validation, the first model is trained on the $1 - 4$ parts of the data and then tested on the 5^{th}, the second is trained on the parts $2 - 5$ and tested on the 1^{st} part, the third - $3, 4, 5, 1$, tested on the 2^{nd} part, the fourth is trained on the $4, 5, 1, 2$ parts, tested on the 3^{rd} part, the fifth - $5, 1, 2, 3$, is tested on the 4^{th}.

2.7 Summary

Predicting user behaviour on the Web is a broad field with a variety of methods and approaches. Predicting user behaviour on the Web involves performing a set of consequent steps. These steps are, first, modelling user behaviour on the Web (see Chapter 3), then, extracting features influencing user behaviour (see Chapter

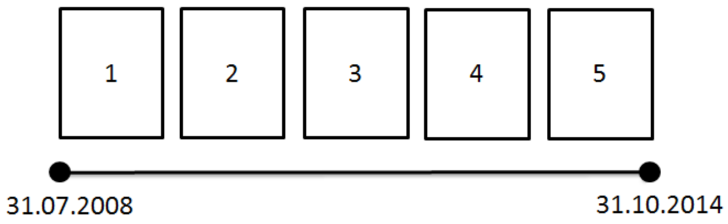


Figure 2.9: 5-fold cross-validation.

4). Third, choosing an efficient ML set-up to build a predictive model using these features (see Chapter 5). Also, choosing visualisations for model exploration is important (see Chapter 6).

This Chapter summarised related works as well as provided necessary background for each step of such a methodology to predict user behaviour on the Web. The next Chapter introduces the first step in this methodology, modelling user behaviour, namely modelling user interaction and their relationship development on the Web.

Chapter 3

Model of User Interaction and Relationship Development on the Web

A model of user behaviour, more precisely a model of user interaction and relationship development on the Web, is proposed in this Chapter. First, a Time-Varying Attributed Graph (TVAG) is proposed. In this graph both nodes and edges are *objects* with time-varying *attributes*. Second, two graphs, an *interaction graph*, representing how people interact with each other, and a *relationship graph*, representing social ties that people form with each other on the Web are constructed using a TVAG. The resulting TVAG-based model combines these two graphs to model temporal dynamics of human interaction and relationship development on the Web.

First, an introduction on modelling user behaviour is presented in Section 3.1. The proposed model of user behaviour is introduced in Section 3.2. A toy modelling example using the proposed model is shown in Section 3.3. Modelling real world platforms, such as Stack Exchange, Twitter, and Facebook using the proposed TVAG-based model is demonstrated in Section 3.4. Finally, Section 3.5 and Section 3.6 provide a conclusion and a summary of the Chapter.

3.1 Introduction

A few graph-based models describing relationships between people, such as social connections [Pfaltz, 2013], and interaction between people have been proposed [Cattuto et al., 2013; Kim and Leskovec, 2011]. In these graph-based models, nodes represent people and edges represent a social connection between these people or an interaction between them. However, there are several shortcomings in such models. First, most models miss temporal properties of communication [Lattanzi, 2010]. Second, these models capture only social properties or only interaction properties of social media [Pfaltz, 2013; Cattuto et al., 2013; Kim and Leskovec, 2011].

The first shortcoming, how to represent temporal properties of communication in a graph model, has been considered from a theoretical point of view [Casteigts et al., 2010; Campbell et al., 2013; Cattuto et al., 2013; Semertzidis and Pitoura, 2016]. For example, a few attempts of expanding graphs into a temporal domain to model dynamic networks were proposed [Casteigts et al., 2010; Cattuto et al., 2013]. However, it is not straightforward to use such abstract models for practical tasks since it is ambiguous how to map a real world network into an abstract graph model [Campbell et al., 2013; Semertzidis and Pitoura, 2016]. In this Chapter, the proposed model is based on Time-Varying Graphs (TVGs) where both nodes and edges have a rich set of time-varying attributes. These attributes allow the use of the proposed Time-Varying Attributed Graph (TVAG) model to encode information from real-world social media networks. On the other hand, the time-varying graph structure enables the performance of a statistically sound analysis of human interaction and social connections on the Web. For example, such analysis includes calculating centrality metrics, distance metrics related to both nodes and edges. As a result, it allows one to analytically estimate the importance of nodes or edges in the network [Cai and Chakravarthy, 2013; Hu et al., 2013; Teinemaa et al., 2015].

The second shortcoming of modern social network models is that of capturing only one particular aspect of user behaviour on the Web. An attempt to overcome this shortcoming was approached, for instance, by combining several graphs in the paper [Lattanzi, 2010]. Also, for example, in [Lattanzi, 2010] the authors combined a graph of an affiliation network and a social network graph to make their model capable of capturing both social ties and interests of people. The affiliation graph is a simple bipartite graph where people are associated with nodes and interests are associated with edges of the graph. The social network graph is a multi-graph where people are associated with nodes and social ties between these people are associated with edges. In [Zhang and Zhang, 2013] the authors introduced a model of a social network as a combination of two graphs, an uncertain graph and an attributed graph. The authors called their model the 'uncertain attribute graph' which is capable of capturing rich information of social networks as well as uncertainty of this information. In this Chapter, the proposed TVAG-based model captures both social and interaction properties of social media networks. This is achieved by combining a social, or relationship, graph and an interaction graph. As a result, the proposed

TVAG-based model allows one to get a wider insight into user behaviour on the Web.

The specificity of the addressed problem can be formulated as the following research question:

- How can one model human interaction and relationship development in order to capture, explore, and facilitate understanding of user behaviour on the Web?

In order to answer this question, the following goals are set:

1. To provide an infrastructure and methodology to capture, explore, and facilitate understanding of user interaction and relationship development on the Web;

2. To help in the identification and exploration of features influencing user behaviour on the Web;

3. To facilitate an efficient and automatic feature extraction for prediction tasks of machine learning.

The first and the second goals are achieved by defining the TVAG-based model. All entities of the proposed model are introduced, and an example of constructing the model is given. Then the proposed model is used on three large social media platforms, namely Twitter, Facebook, and Stack Exchange to demonstrate how to use the proposed TVAG-based model. It is shown how the model captures temporal properties of user behaviour on the Web as well as how the interaction and relationship development is captured by the model.

The third goal is achieved by introducing the temporality in the attributes of the proposed model. As a result, the temporal attributes allow one to sample the graph into subgraphs. Sampling the graph into subgraphs leads to an efficient feature extraction since the features are calculated only for subgraphs spanning a limited time interval. This approach for feature extraction scales well for large temporal networks with millions of nodes and multiple attributes while preserving the structural characteristics of such networks.

3.2 Proposed Model of User Behaviour

The model of user behaviour on the Web is built with elements adopted from communication theory [Shannon, 1948], graph theory [Casteigts et al., 2010], and social network analysis [Kim and Leskovec, 2011; Cattuto et al., 2013; Semertzidis and Pitoura, 2016].

In information theory, a model of communication has an information source that produces a message and a destination, or a target, for whom the message is sent [Shannon, 1948]. The message is sent via a channel which is the medium of communication. A transmitter encodes the message to create a signal which is then decoded by a receiver. Human communication and interaction on the Web can be considered as an exchange of information between people who serve as information sources and as receivers and the channel of communication is the Web.

In graph theory, mathematical structures consisting of objects, or nodes connected by edges are studied. Recently, Time Aggregated Graphs (TAGs), graphs which change over time were proposed [Casteigts et al., 2010]. Nevertheless, TAGs is a relatively new concept and has not been explored much, especially real world applications of TAGs.

In social network theory, social networks consist of individuals, organisations, or other actors interacting with each others inside their network. The actors are connected by ties, for example, friendship or interaction ties. A social network can be considered as a graph in which nodes are associated with actors of the social network and edges are associated with the social ties the actors form. Thus, graph theory can be employed for studying the properties of social networks and for identification of important actors in the networks.

In this Chapter, a model of human interaction and relationship development is proposed. This model consists of two graphs, an *interaction graph* and a *relationship graph.* Both graphs are instances of TVAGs which are introduced as TAGs in which both nodes and edges have attributes. The nodes and edges of TVAGs are defined as time-varying *objects* which are described below.

3.2.1 Objects

The core of the model of human interaction and relationship development are connected *objects* interacting with each other. Each *object* has time-varying *attributes.*

Definition 3. *An attribute $a(t)$ is a time-varying function defined for the discrete time t in $\mathbb{Z}^+$. An object $o(A_o(t), t)$ is a time-varying entity with a set of time-varying attributes $a_o(t) \in A_o(t)$. Each object $o(A_o(t), t)$ as well as the attributes of this object $A_o(t)$ are defined for the discrete time t in $\mathbb{Z}^+$. Each object has a time span $\mathbb{T} \subset \mathbb{Z}^+$.*

An object $o(A_o(t), t)$ can have no attributes at all, $A_o(t) = \{\emptyset\}$ or an in-finite number of attributes $A_o(t) = \{a_{o1}(t), ..\}$. The attributes $A_o(t)$ of an object $o(A_o(t), t)$ hold the information on the object and can be divided in two groups: numerical and categorical attributes. Numeric attributes have values that describe a measurable quantity as a number whereas categorical attributes have values that describe a 'quality' or 'characteristic' of the object. Numeric attributes can be continuous or discrete and categorical attributes can be nominal or ordinal. Thus, each attribute $a_o(t) \in A_o(t)$ can be continuous **C**, discrete **D**, nominal **N**, or ordinal **O**.

Definition 4. *The attribute $a_o(t)$ is called continuous **C** iff this attribute can take any value between a certain set of real numbers $\mathbb{R}$. The attribute is called discrete **D** iff it can only take a value from a set of distinct whole values. The attribute is called nominal **N** iff this attribute cannot be organised in a logical sequence. The attribute is called ordinal **O** iff this attribute can only take a value that can be logically ordered or ranked.*

An example of a *continuous* attribute is the height of a person measured with a ruler. On contrary to a *continuous* attribute, a *discrete* attribute cannot take the value of a fraction between one value and the next closest value. For example, the number of children in a family is a *discrete* attribute. The clothing sizes such as small, medium, large, and extra large is an *ordinal* attribute since it is possible to logically order these attributes. On contrary, gender or eye colour cannot be logically ordered, thus, gender and eye colour are *nominal* attributes.

Next, these objects $O(A_o(t), t)$ are used to derive the proposed TVAG model and then the model of human interaction and relationship development based on TVAGs is introduced.

3.2.2 Time-Varying Attributed Graph (TVAG)

A TVAG is a graph in which all nodes and edges have time-varying attributes. Both nodes and edges are assumed to take place over a time span $T \subseteq \mathbb{T}$ called the lifetime of the TVAG. The temporal domain $\mathbb{T}$ is $\mathbb{N}$ in this model since discrete time is appropriate for the way time stamped events are collected on the Web. Shortly, a TVAG is defined as follows:

Definition 5. *A Time-Varying Attributed Graph (TVAG) is a graph $G(t)$ such that $G(t) = (N(A_n(t), t), E(A_e(t), t))$ where the nodes $N(A_n(t), t)$ are connected by edges $E(A_e(t), t)$. The nodes $N(A_n(t), t)$ and the edges $E(A_e(t), t)$, are instances of the objects $O(A_o(t), t)$, and $A_n(t)$ and $A_e(t)$ are the attributes of the nodes and the edges respectively.*

Next, an *interaction* and a *relationship* graphs are introduced.

3.2.3 Relationship Graph

Social media consists of connected *users* who interact with each other. While communicating, the *users* establish *relationships*, for example friendship *relationships*. *Users* and *relationships* are instances of the *objects* and they form a TVAG, a *relationship graph* $G_r(t)$, where the nodes of the graph are associated with *users* and the edges are associated with *relationships*.

Definition 6. *The users $U(A_u(t), t)$ are connected by relationships $R(A_r(t), t)$ where $U(A_u(t), t)$ and $R(A_r(t), t)$, are instances of the objects $O(A_o(t), t)$, and $A_u(t)$ and $A_r(t)$ are the attributes of the users and the relationships respectively. The relationship graph $G_r(t) = (U(A_u(t), t), R(A_r(t), t))$ is a graph representing the relationships development between the users over the time t.*

Since the users $U(A_u(t), t)$ are connected by the relationships $R(A_r(t), t)$, each relationship $r(A_r(t), t) \in R(A_r(t), t)$ has at least two attributes, a source user $a_{rSource}(t)$ and a target user $a_{rTarget}(t)$ such that $a_{rSource}(t)$ uniquely identifies a user $u_{source}(A_{uSource}(t), t)$ and $a_{rTarget}(t)$ uniquely identifies $u_{target}(A_{uTarget}(t), t)$, where $a_{rSource}(t), a_{rTarget}(t) \in A_r(t)$ and $u_{source}(A_{uSource}(t), t), u_{target}(A_{uTarget}(t), t) \in U(A_u(t), t)$.

3.2.4 Interaction Graph

While communicating, the *users* interact with each other by exchanging *messages*. Thus, *users* and *messages* which are instances of the *objects* form a TVAG, an *interaction graph* $G_{int}(t)$, where the nodes of the graph are associated with *users* and the edges are associated with *messages*.

Definition 7. *The users $U(A_u(t), t)$ interact with each other via sending and receiving messages $M(A_m(t), t)$, where $U(A_u(t), t)$ and $M(A_m(t), t)$, are instances of the objects $O(A_o(t), t)$, and $A_u(t)$ and $A_m(t)$ are the attributes of the users and the messages respectively. The interaction graph $G_{int}(t) = (U(A_u(t), t), M(A_m(t), t))$ is a graph representing the communication between the users.*

These two TVAGs, namely the *interaction graph* $G_{int}(t)$ and the *relationship graph* $G_r(\mathrm{t})$, describe the dynamics of communication and relationships of the users on the Web. An example of such graphs is shown in Figure 3.1 and a step-by-step construction of these two graphs for this example is given further in Section 3.3.

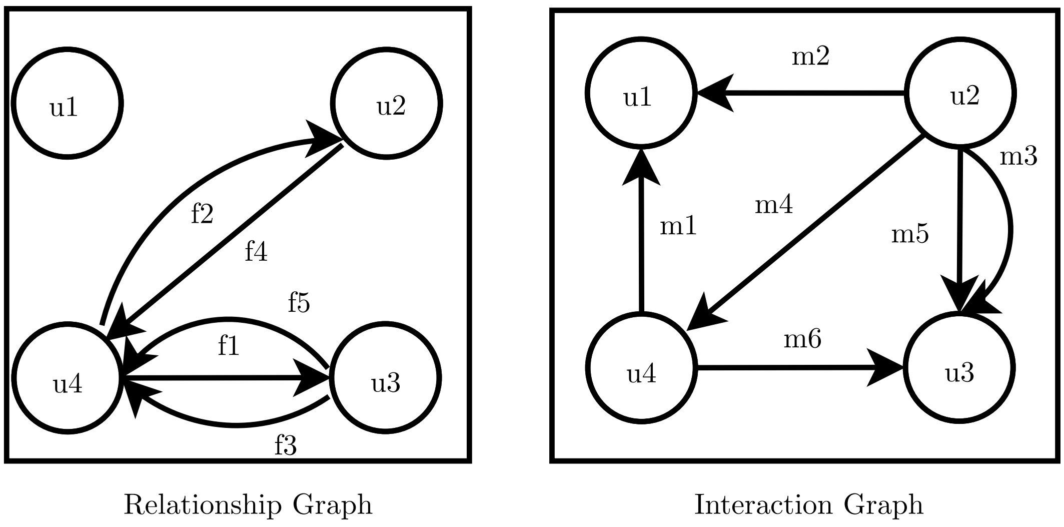


Figure 3.1: An example of a *relationship graph* and an *interaction graph* representing communication of four users other a fixed time period T.

Notation Shorthand

Further in this thesis, users $U(A_u(t), t)$ sometimes are denoted as $U(t)$ or just U for a shorter notation. The same notation shorthand applies for relationships $R(A_r(t), t)$, messages $M(A_m(t), t)$, and a single user u, a relationship r, or a message m. Also, the *relationship graph* $G_r(t) = (U(A_u(t), t), R(A_r(t), t))$ sometimes is denoted as $G_r(t) = (U(t), R(t))$ or just G_r for a shorter notation. The same applies for the *interaction graph* $G_{int}(t)$.

3.2.5 User Attributes

Users U are represented by the nodes in both relationship and interaction graphs G_r and G_{int}. A user $u_i \in U$ appears in the graphs G_r and G_{int} once he/she registers in a social media network or he/she is registered by another user. Every user u_i has a 'birth' time stamp t_b when a user u_i was registered in social media. Once the node was created, the attributes A_u of the node u_i can be created and initiated. Later on, these attributes can be changed, updated, or new attributes can be introduced. A set of possible user node attributes is presented below:

- aId: Unique Id (nominal, mandatory)
- $aName$: Name (categorical, optional)
- $aLocation$: Location (categorical, optional)
- $aPopularity$: Popularity (continuous, optional)
- $aAge$: Age (continuous, optional)

- *aGender*: Gender (discrete, optional)
- *aTimeRegistered*: Time Registered (discrete, optional)

For example, if Nick, a male user located in the US, registered on October 10th then a node $u(t) = u(a_u(t), t)$ is created, where $a_u(t)$ are the node attributes and t is the time:

$$\begin{aligned} u(a_u(t), t) = \{&aName(t) = \{(Nick, t_b)\}, \\ &aLocation(t) = \{(US, t_b)\}, \\ &aPopularity(t) = \{(0, t_b)\}, \\ &aAge(t) = \{(null, t_b)\}, \\ &aGender(t) = \{(male, t_b)\}, \\ &aTimeRegistered(t) = \{(t_b, t_b)\} \\ &\} \end{aligned} \tag{3.1}$$

As the reader can notice, all the attributes are temporal.

3.2.6 Relationship Attributes

Every user $u_i(t)$ can establish a relationship $r(t)$ to another user. The established relationships form the set of edges $R(A_r(t), t)$ with two mandatory attributes, namely $aSource(t)$ and $aTarget(t)$ which represent the source and the target users who form the relationship $r(t)$. The user can establish a relationship to himself, thus $aSource(t) = aTarget(t)$. The user can have any number of relationships with other users or none at all. The relationship $r(t)$ can have the following attributes $A_r(t)$:

- $aSource = \{UserId1\}$ (nominal, mandatory)
- $aTarget = \{UserId2\}$ (nominal, mandatory)
- $aType = \{\}$ (nominal, optional)

A relationship $r(t)$ can be of different types $aType(t)$, for example, friendship, blood relations, work relations, a status of following a person. Also, the relationship type can change over time from one nominal value to another. For example, let assume that Nick and Jack register on the Web with their user accounts $u_{Nick}(t)$ and $u_{Jack}(t)$ respectively. Then Nick starts following Jack on July 21st. As a result, a relationship

$r(a_r(t), t)$ is created:

$$\begin{aligned} r(a_r(t), t) = \{&aSource(t) = \{(Nick, t_1)\}, \\ &aTarget(t) = \{(Jack, t_1)\}, \\ &aType(t) = \{(following, t_1)\}, \\ &\} \end{aligned} \tag{3.2}$$

Where $Nick$ is a unique id which identifies $u_{Nick}(t)$, $Jack$ is a unique id which identifies $u_{Jack}(t)$, $following$ is the type of the relationship they establish, and t_1 is the date when Nick started following Jack which is July 21^{st}.

3.2.7 Message Attributes

Once a user u is 'born', he/she can interact with other users by sending messages M. Users $U(t)$ and messages $M(t)$ form an interaction graph $G_{int}(t)$ where the users are the nodes and the messages are the edges of the graph. The attributes $A_m(t)$ of the edges $M(t)$ can be the following:

- $aSource = \{UserId1\}$ (nominal, mandatory)
- $aTarget = \{UserId2\}$ (nominal, mandatory)
- $aType = \{\}$ (nominal, optional)
- Message Body (nominal, optional)

A message m must have a source $aSource$ and a target $aTarget$. The message itself can be empty or contain some information, for example, text, images, url links. For example, if Nick sends a message 'how are you?' to Jack on July 22^{nd} then a message $m(a_m(t), t)$ is created:

$$\begin{aligned} m(a_m(t), t) = \{&aSource(t) = \{(Nick, t_2)\}, \\ &aTarget(t) = \{(Jack, t_2)\}, \\ &aType(t) = \{(text\,message, t_2)\}, \\ &aMessageBody(t) = \{('how\,are\,you?', t_2)\} \\ &\} \end{aligned} \tag{3.3}$$

Where $Nick$ is a unique id which identifies $u_{Nick}(t)$, $Jack$ is a unique id which identifies $u_{Jack}(t)$, $text\,message$ is the type of the message, and t_2 is the date when Nick sent the message to Jack which is July 22^{nd}.

Next, a step-by-step example describes how the proposed model captures temporal dynamics of human interaction and relationship development.

3.3 A Toy Modelling Example

A toy example of temporal dynamics for four users registering and then interacting and establishing relationships is introduced. As a result, the process of forming an interaction and relationship graph is presented.

3.3.1 User Attributes

Assume that at the time t_{01} a user u_1 registers on the Web and then the user enters some information about himself, for example, his name, age, and location. Thus, the attributes $A_{u1}(t)$ of the user u_1 are updated with the attributes at time t_{01}:

$$\begin{aligned} A_{u1}(t) = \{&aName_{u1}(t) = \{(name_1, t_{01})\}, \\ &aAge_{u1}(t) = \{(age_1, t_{01})\}, \\ &aLocation_{u1}(t) = \{(location_1, t_{01})\} \\ &\} \end{aligned} \tag{3.4}$$

Then at the time t_{02} the user u_1 updates his location to $location_2$ and another user u_2 registers but enters only his name $name_2$:

$$\begin{aligned} A_{u1}(t_{02}) &= \{aLocation_{u1}(t_{02}) = \{(location_2, t_{02})\}\} \\ A_{u2}(t_{02}) &= \{aName_{u2}(t_{02}) = \{(name_2, t_{02})\}\} \end{aligned} \tag{3.5}$$

Thus, the users' information becomes:

$$\begin{aligned} u1(A_{u1}(t), t) = \{&aName_{u1}(t) = \{(name_1, t_{01})\}, \\ &aAge_{u1}(t) = \{(age_1, t_{01})\}, \\ &aLocation_{u1}(t) = \{(location_1, t_{01}), (location_2, t_{02})\} \\ &\} \\ u2(A_{u2}(t), t) = \{&aName_{u2}(t) = \{(name_2, t_{02})\}, \\ &\} \end{aligned} \tag{3.6}$$

Finally, at the time t_{03} two users, u_3 and u_4 register on the Web and the user u_2 changes his name to $name_3$. That means that the attributes for the users u_3 and u_4 are assigned to an empty set $\emptyset$ since they did not enter any information whereas

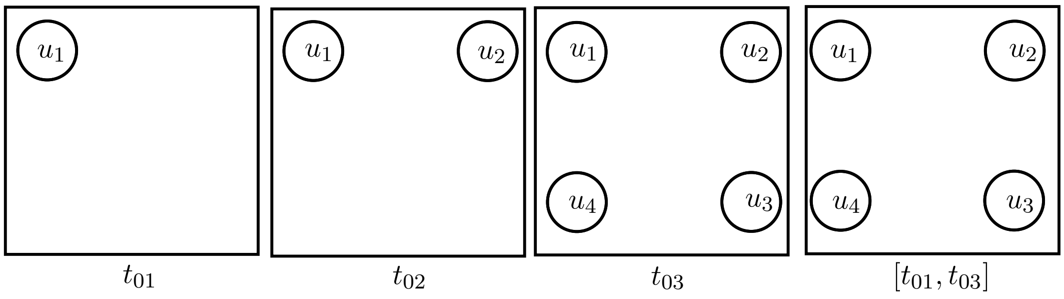


Figure 3.2: Users' 'births' and updates for the three time points t_{01}, t_{02}, t_{03} and then for the interval $[t01, t03]$.

the attribute 'name' of the user u_2 is updated with $name_3$:

$$\begin{aligned} A_{u2}(t_{03}) &= \{name_{u2}(t_{03}) = (name_3, t_{03})\} \\ A_{u3}(t_{03}) &= \{\emptyset\} \\ A_{u4}(t_{03}) &= \{\emptyset\} \end{aligned} \tag{3.7}$$

This scenario of users' births and their information updates is depicted in Figure 3.2.

3.3.2 Interaction Graph

An example of forming an *interaction graph* is shown in Figure 3.3. At the time moment t_{11} the user u_4 sends a message m_1 to the user u_1:

$$\begin{aligned} A_{m1}(t_{11}) = \{&aSource_{m1}(t_{11}) = (u_4, t_{11}), \\ &aTarget_{m1}(t_{11}) = (u_1, t_{11})\} \end{aligned} \tag{3.8}$$

Then at the time t_{12} the user u_2 sends the messages m_2 and m_3 to the users u_1 and u_3:

$$\begin{aligned} A_{m2}(t_{12}) = \{&aSource_{m2}(t_{12}) = (u_2, t_{12}), \\ &aTarget_{m2}(t_{12}) = (u_1, t_{12})\} \\ A_{m3}(t_{12}) = \{&aSource_{m3}(t_{12}) = (u_2, t_{12}), \\ &aTarget_{m3}(t_{12}) = (u_3, t_{12})\} \end{aligned} \tag{3.9}$$

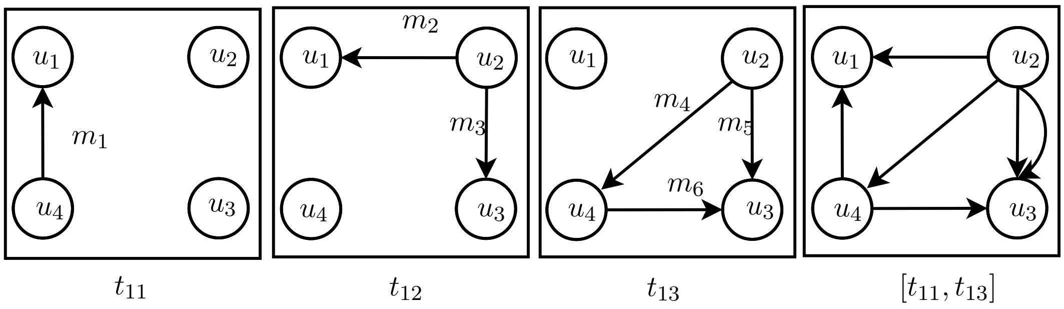


Figure 3.3: An *interaction graph* representing communication of four users for the three time points t_{11}, t_{12}, t_{13} and then for the interval $[t11, t13]$.

Finally, at the moment t_{13} the user u_2 sends the messages m_4 and m_5 to the users u_3 and u_4; also, the user u_4 sends the message m_6 to the user u_3:

$$\begin{aligned} A_{m4}(t_{13}) &= \{aSource_{m4}(t_{12}) = (u_2, t_{12}), \\ &\quad aTarget_{m4}(t_{12}) = (u_3, t_{12})\} \\ A_{m5}(t_{13}) &= \{aSource_{m5}(t_{12}) = (u_2, t_{12}), \\ &\quad aTarget_{m5}(t_{12}) = (u_4, t_{12})\} \\ A_{m6}(t_{13}) &= \{aSource_{m6}(t_{12}) = (u_4, t_{12}), \\ &\quad aTarget_{m6}(t_{12}) = (u_3, t_{12})\} \end{aligned} \tag{3.10}$$

The result of such communication is the following *interaction graph*:

$$\begin{aligned} &G_{int} = (U(A_u(t), t), M(A_m(t), t)) \quad where \\ &U(A_u(t), t) = \{u_i(A_{ui}(t), t)\},\ i = 1, .., 4 \quad and \\ &M(A_m(t), t) = \{m_j(A_{mj}(t), t)\},\ j = 1, .., 6 \end{aligned} \tag{3.11}$$

This scenario of users interaction and the resulting interaction graph is depicted in Figure 3.3.

3.3.3 Relationship Graph

An example of forming a *relationship graph* is shown in Figure 3.4. At the time moment t_{21} the user u_4 establishes a relationship with the user u_3 and the relationship r_1 is created:

$$\begin{aligned} A_{r1}(t_{21}) &= \{aSource_{r1}(t_{21}) = u_4, \\ &\quad aTarget_{r1}(t_{21}) = u_3\} \end{aligned} \tag{3.12}$$

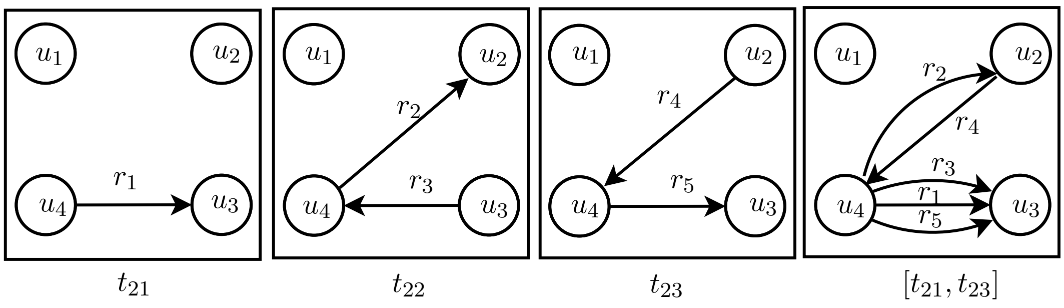


Figure 3.4: A *relationship graph* representing development of relationships between four users for the three time points t_{21}, t_{22}, t_{23} and then for the interval $[t21, t23]$.

Then at the time t_{22} the user u_3 confirms the relationship with the user u_4. Also, the user u_4 establishes a relationship with the user u_2 which leads to creation of the link r_2:

$$\begin{aligned} A_{r2}(t_{22}) = \{&aSource_{r2}(t_{22}) = u_4, \\ &aTarget_{r2}(t_{22}) = u_2\} \\ A_{r3}(t_{22}) = \{&aSource_{r3}(t_{22}) = u_4, \\ &aTarget_{r3}(t_{22}) = u_3\} \end{aligned} \tag{3.13}$$

Finally, at the moment t_{23} the user u_2 confirms a relationship with the user u_4 and the user u_4 establishes another relationship with the user u_3:

$$\begin{aligned} A_{r4}(t_{23}) = \{&aSource_{r4}(t_{23}) = u_2, \\ &aTarget_{r4}(t_{23}) = u_4\} \\ A_{r5}(t_{23}) = \{&aSource_{r5}(t_{23}) = u_4, \\ &aTarget_{r5}(t_{23}) = u_3\} \end{aligned} \tag{3.14}$$

The result of such relationship development is the following *relationship graph*:

$$\begin{aligned} &G_r = (U(A_u(t), t), R(A_r(t), t)) \quad where \\ &U(A_u(t), t) = \{u_i(A_{ui}(t), t)\}\, i = 1, .., 4 \quad and \\ &R(A_r(t), t) = \{r_k(A_{rk}(t), t)\},\ k = 1, .., 5 \end{aligned} \tag{3.15}$$

3.4 Modelling Examples

Real world social media platforms can be very diverse and be represented, for example, by a social network, a blogging platform or a Q&A forum. In this Section, the proposed TVAG-based model is instantiated for three diverse platforms, namely

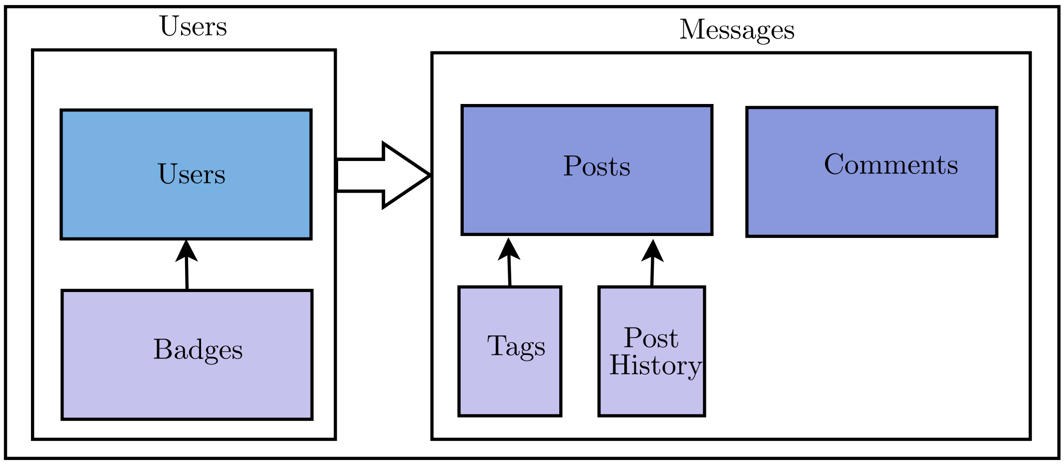


Figure 3.5: An overview of the data model of *Stack Exchange* websites.

Stack Exchange, *Twitter*, and *Facebook* communities.

3.4.1 Stack Exchange

Stack Exchange websites facilitate users to ask and answer questions by posting messages on the platform. Each message can be one of tree types - a question, an answer, or a comment. An *interaction graph* is formed by the users and their messages. The type of a message is encoded by an attribute.

Similar to *Twitter*, questions on *Stack Exchange* are tagged with predefined *tags*. A tag is a keyword reflecting a topic or a domain of the question. Each question can have up to six tags. Analogously to a *Twitter* hashtag, a *Stack Exchange* tag is an attribute of a message. The message in *Stack Exchange* can be a question, an answer, or a comment whereas on *Twitter* there is a tweet, a retweet, a mention, a reply.

Stack Exchange websites are uniform in their structure. In total there 128 *Stack Exchange* websites on different topics. The data model is the same for all of the websites and consists of six tables:

- Users;
- Posts;
- Tags;
- Badges;
- Comments;
- PostHistory.

The relationships between these six tables are shown in Figure 3.5. The transformation of the existing data model into the proposed TVAG model results in an

interaction graph $G_{int}(U, M)$ with the following attributes:
User attributes U_a:

- Registration Date $U_{aRegistrationDate}$: Time
- Display Name $U_{aDisplayName}$: Text
- Reputation $U_{aReputation}$: Integer
- Email Hash $U_{aEmailHash}$: Text
- Website Url $U_{aWebsiteUrl}$: Text
- Location $U_{aLocation}$: Text
- Age U_{aAge}: Integer
- About Me $U_{aAboutMe}$: Text
- Views U_{aViews}: Integer
- Upvotes $U_{aUpvotes}$: Integer
- Downvotes $U_{aDownvotes}$: Integer

Stack Exchange records some temporal changes in the messages, for example changes in titles and bodies of questions. Also, changes in the tags a question is tagged with are recorded as well. This fact allows one to construct $M_{aTitle}(t)$, $M_{aBody}(t)$, and $M_{aTags}(t)$. For all messages M without any temporal information, the only temporal attribute associated is the time when a message m was created which is reflected in $CreationDate$ attribute of Stack Exchange data model.
Message attributes M_a:

- Source $M_{aSource}$: UserId1
- Target $M_{aTarget}$: UserId2
- Type M_{aType}: $\{question, answer, comment, badge\}$
- Action $M_{aAction}$: $\{Posted, Liked, Upvoted, Downvoted, Received\}$
- Score: M_{aScore}: Integer
- Title: M_{aTitle}: Text
- Body: M_{aBody}: Text
- Tags: M_{aTags}: Text
- Text M_{aText}: textual message
- Link M_{aLink}: Url link

Stack Exchange does not have a functionality for users to establish any relationships. This leads to the fact there is no explicit *relationship graph.* Thus, only a *interaction graph* can be constructed from the question answering activities.

An example of constructing an interaction graph is shown in Figure 3.6. First, a question q_1 is asked by a user u_1. Second, the question receives an answer a_1 posted by a user u_2. Third, a user u_3 posts another answer a_2. Then a user u_4 writes a comment c_1 for the answer a_2. Finally, the user u_2 writes another comment c_2 to the answer a_2. The resulting interaction graph for such a scenario is shown in the last, fifth, box in Figure 3.6.

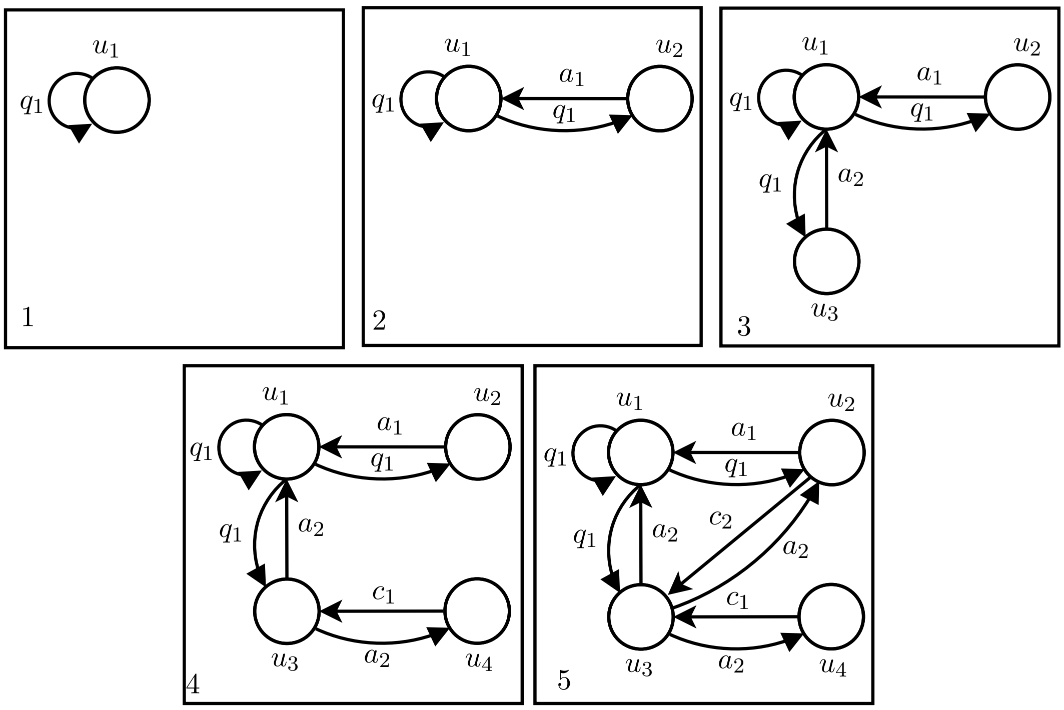


Figure 3.6: A step-by-step example of constructing an interaction graph for *Stack Exchange* websites. The graph consists of four nodes u_1, u_2, u_3, u_4 associated with four users, an edge q_1 associated to a question asked by the user u_1, two edges a_1 and a_2 associated to two answers for this question, and two comments c_1 and c_2 associated to two comments to the second answer a_2.

Model Construction for a Stack Exchange Dataset

Stack Exchange provides data dumps of their data for all of their 128 websites. The largest Q&A forum Stack Overflow has $7,990,488$ questions and $13,683,746$ answers with $3,472,204$ users for the time period from July 31, 2008 to October 23, 2014. An interaction graph for 1,000 the most recent questions is shown in Figure 3.7. The total number of answered questions for the time interval from July 31, 2008 to October 23, 2014 is shown in Figure 3.8. Answered questions are defined as questions which have at least one answer accepted by the asker.

3.4.2 Twitter

Twitter was established as a micro blogging platform where users can post messages or so-called 'tweets' and establish relationships with other users by following each other. Due to the limitation on the length of a tweet before 2015, users used to actively attach urls to their tweets to share more information. Another important feature of *Twitter* is the term *hashtag* which is a string starting with the character '#'. A *hashtag* represents a topic, event, a keyword.

According to the official documentation provided at *Twitter* website[1], the data model of *Twitter* is as follows:

- Object: Users
- Object: Tweets
- Object: Entities
- Object: Entities in Objects
- Object: Places

The documentation says that *"Users can be anyone or anything. They tweet, follow, create lists, have a home timeline, can be mentioned, and can be looked up in bulk."*, and about tweets: *"Tweets are the basic atomic building block of all things [in] Twitter. Tweets, also known more generically as "status updates." Tweets can be embedded, replied to, liked, unliked and deleted."*. Thus, there is mapping between users, tweet activity and follower-following relationships on Twitter to users, messages and relationships of the proposed TVAG-based model. For example, if a user u_i follows a user u_j then a directed link from the user u_i to the user u_j is established. The graph where the users are the nodes and the following relationships are the links, is the *relationship graph.* However, *Twitter* does not provide temporal information on relationship graph at all, thus, it is assumed that all relationships as old as the corresponding user's registration date.

[1]https://dev.twitter.com/overview/api

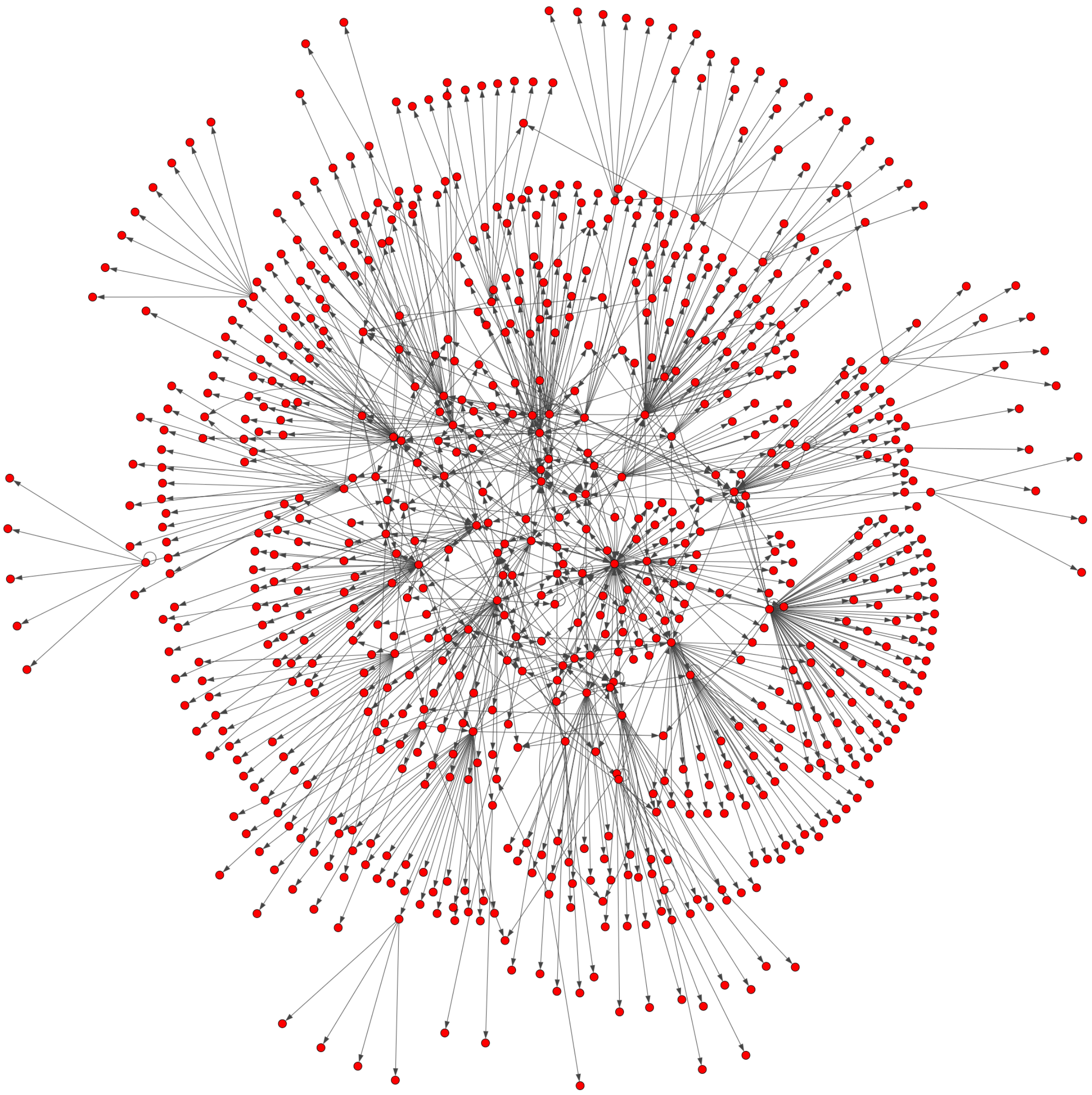


Figure 3.7: An *interaction graph* for 1,000 of the most recent questions on Stack Overflow. The nodes represent users and the edges represent the flow of answering questions of other users.

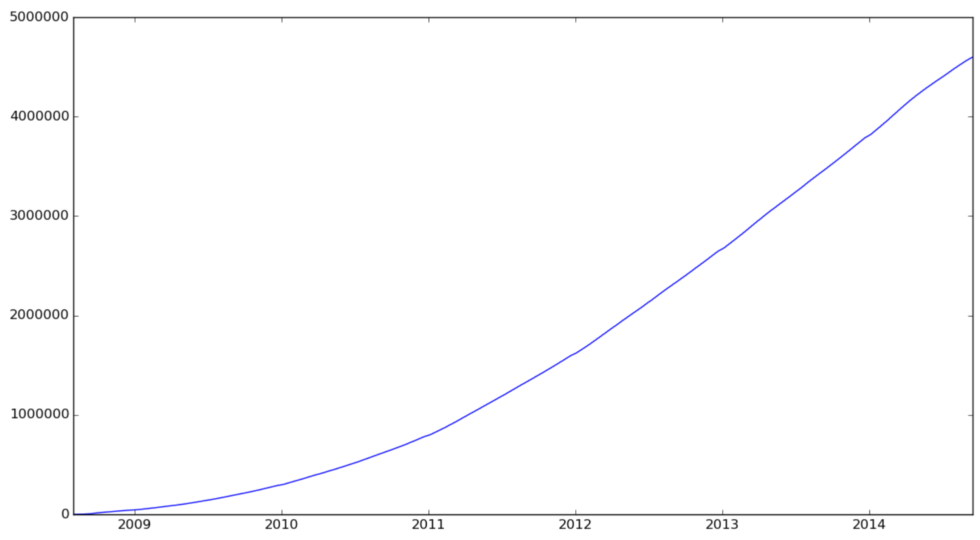


Figure 3.8: The total number of answered questions for the time from 31 July 2008 to 23 October 2014.

The communication and interaction between users in *Twitter* is organised by *tweets*, the messages sent by the users. Technically, all tweets are public and can be seen by anyone. However, only users who follow a tweeted person will see his tweet. Thus, a user's *tweet* m is broadcasted only to all his $followers$. Also, if a tweet starts with "@username1" then this tweet will not be automatically broadcasted to the timeline of the follower @username2 in cases when @username2 is not following @username1.

Users can interact with each other via *retweet* and *mention* in Twitter. A *retweet* is identified by the text "RT @username" or "via @username" in a tweet m. A *mention* is a *tweet* with @username if it is not a *retweet*. Thus, the *interaction graph* is formed by the users and their tweets where each tweet can be a post, a retweet, a reply or a mention. The interaction graph on Twitter is timestamped.

Model Construction for a Twitter Dataset

Twitter has an API for searching and downloading their data. Below is an example of using hashtags for searching and extracting Twitter data and then fitting the extracted data into the proposed TVAG-based model.

For searching tweets by a hashtag #*hashtag* the following query must be executed:

$$http://search.twitter.com/search.json?q = \%23hashtag \tag{3.16}$$

The query returns timestamped tweets as well as unique identification numbers (ids) of users who posted the tweets. These timestamped tweets allow to form an *interaction graph*. Next, the unique users' ids can be used for extracting the *relationship graph* of the proposed model. The relationship graph consists of following-follower relationships between the users. As a result, the two extracted graphs form the proposed model. However, only the interaction graph has temporal features, thus, it is assumed that all relationship ties were formed instantaneously at the earliest interaction between the people in the network.

In [De Domenico et al., 2013] the authors extracted tweets by hashtags *lhc*, *cern*, *boson*, *higgs*. The tweets were filtered by date so only the tweets from $00:00$ AM 1st July 2012 to $11:59$ PM 7th July 2012 were considered. As a result $985,692$ tweets were extracted. Each tweet has a timestamp, unique id of a user who posted it, information on the tweet such as whether the tweet is a reply (RE), retweet (RT), or a mention(MT). These tweets form an interaction graph where each node corresponds to a user and directed edges represent the flow of tweets from that user. Each edge has two attributes, namely the time a tweet was posted and the type of a tweet - RE, RT, or MT. The interaction graph for 1,000 of the most active users is shown in Figure 3.9. The relationship graph for 5,000 of the most active users is shown in Figure 3.10.

All the extracted $985,692$ tweets correspond to $456,631$ authors. The relationship graph has $456,631$ nodes corresponding to the authors connected by $14,855,875$ directed edges associated with follower-following relationships.

As a result, the two graphs, interaction and relationship graphs, form the proposed TVAG-based model. The interaction graph is a timestamped graph where each edge has the time attribute but the relationship graph is timestamped with the time of the first tweet in the networks since Twitter API does not provide the time when a relationship between the users was established, terminated, and established again. Below is the list of attributes for users, relationships, and messages:
User attributes A_u:

- Anonymised User's Name U_{aName}: UserId

Relationship attributes A_r:

- Source $R_{aSource}$: UserId1

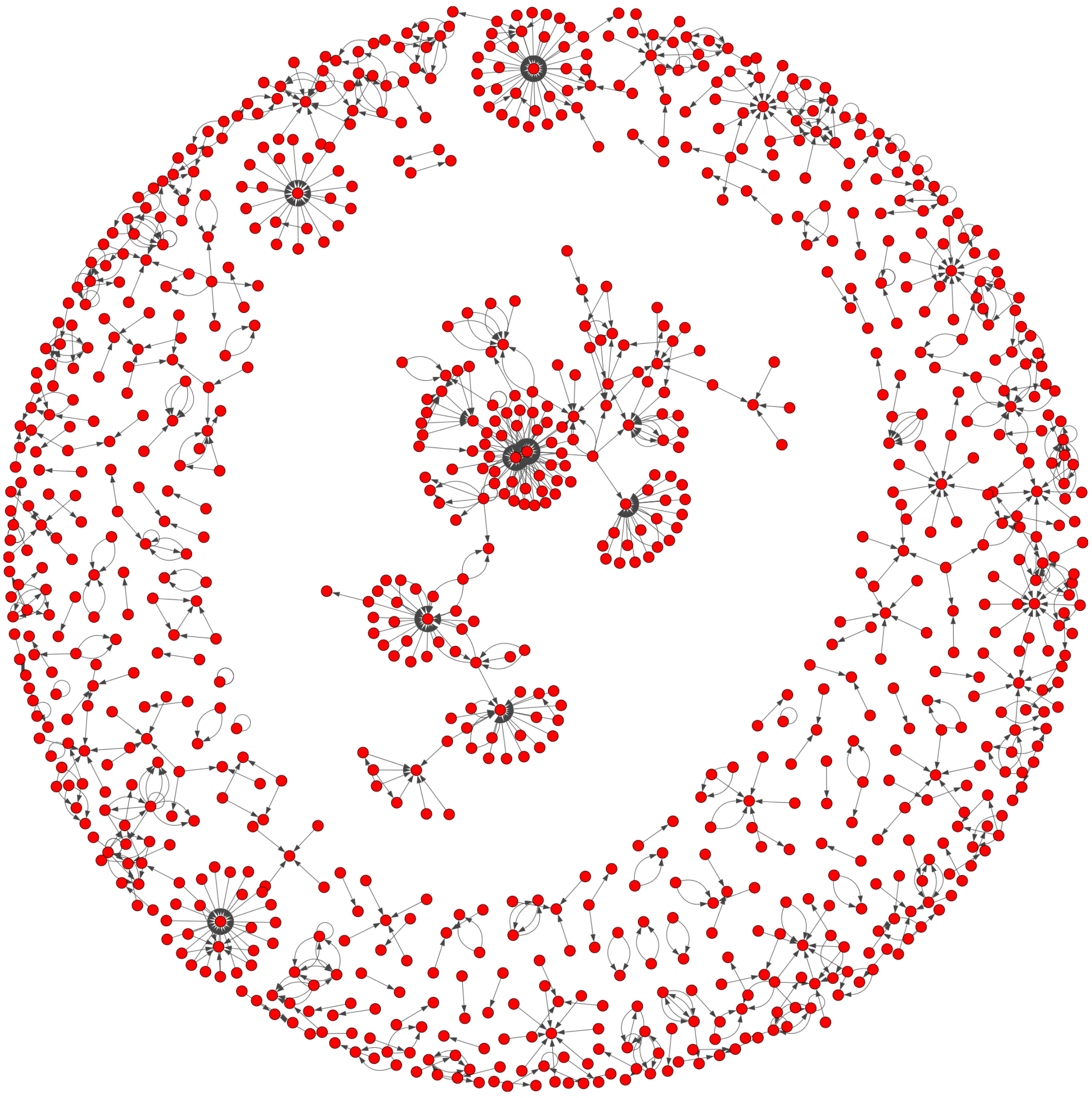


Figure 3.9: An *interaction graph* for 1,000 of the most active users on Twitter. The nodes represent people and the edges represent replies, retweets and mentions by users. The most active users are the users who sent the largest number of tweets.

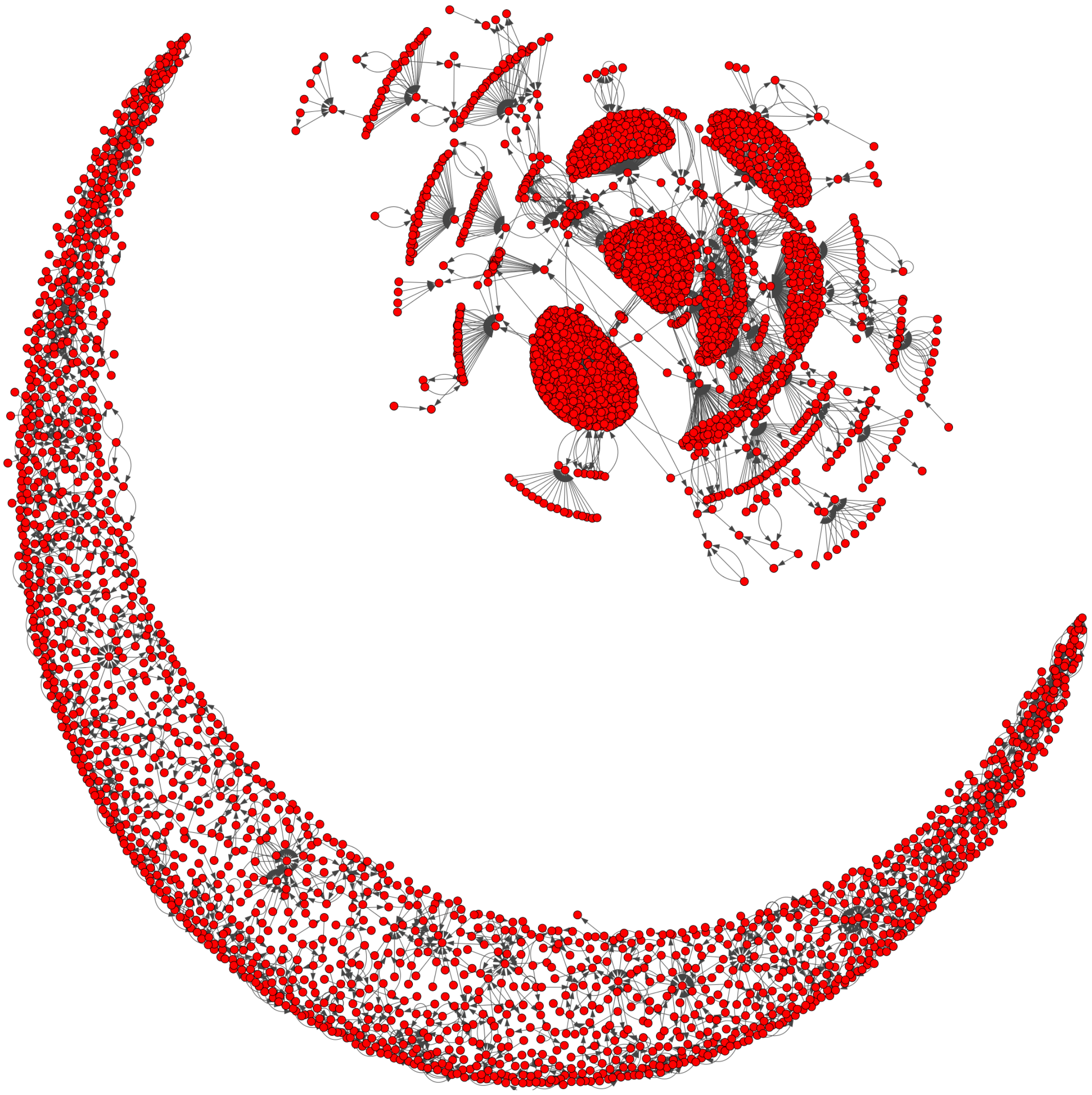


Figure 3.10: A *relationship graph* for 5,000 of the most active users on Twitter. The nodes represent users and the edges represent following-follower relationships between the users.

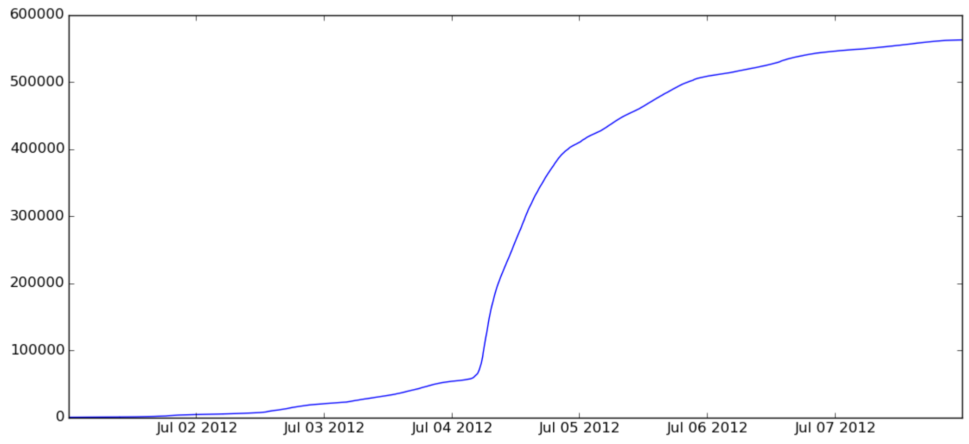


Figure 3.11: The total number of tweets during the time from $00:00$ AM 1^{st} July 2012 to $11:59$ PM 7^{th} July 2012.

- Target $R_{aTarget}$: UserId2
- Type R_{aType}: $\{following\}$

Message attributes A_m:

- Source $M_{aSource}$: UserId1
- Target $M_{aTarget}$: UserId2
- Type M_{aType}: $\{retweeting(RT), replying(RE), mentioning(MT)\}$

As a result, an interaction graph $G_{int} = (U, M)$ and a relationship graph $G_r = (U, R)$ are constructed.

3.4.3 Facebook

Facebook provides *people* with means to communicate, take *actions* and establish *relationships* with other people. Facebook developed a data model, called TAO, for modelling these entities and connections as a graph. According to Facebook, TAO is optimised for reads, and explicitly favours efficiency and availability over consistency [Bronson et al., 2013]. TAO *objects* are typed nodes, and TAO *associations* are typed directed edges between *objects*:

$$\begin{aligned} Object &: (id) \rightarrow (otype, (key \rightarrow value)*) \\ Association &: (id1, atype, id2) \rightarrow (time, (key \rightarrow value)*) \end{aligned} \tag{3.17}$$

Every *object*, regardless of its object type *otype*, can be identified by a unique id which is a 64-bit integer. *Associations* can be identified by the source object $id1$, association type *atype*, and destination object $id2$. At most one association of a given type can exist between any two objects. All objects and associations may contain data represented as a *key* $\rightarrow$ *value* pair [Bronson et al., 2013]. *Actions* may be encoded either as *objects* or *associations*. However, repeatable *actions* are usually encoded as *objects*.

For example, in TAO, users are *objects* and the relationships between the users, for example the status of being a friend, are *associations*. Messages, such as private messages or messages to the feed of a user can be considered as *associations* as well. The difference with the proposed TVAG-based model is that in TVAG users, relationships, and messages are timestamped objects and relationships as well as messages have two time-varying attributes uniquely identifying the source and the destination objects analogously to $id1$ and $id2$ of the associations in TAO.

Facebook allows users to interact in several ways, for example by messaging each other, through applications, uploading pictures, sharing videos, chatting, posting to each other walls.

Next, an example of constructing the proposed model from a Facebook dataset is introduced.

Model Construction for a Facebook Dataset

Facebook website provides with means for extracting user data from a Facebook user page. All the information available at Facebook can be classified as information on Users, Messages, and Relationships [Facebook, 2016]. By analysing the available data, one can see that Facebook does not provide historical values for every single attribute. For the most of the attributes only the latest value is provided. To convert Facebook data in the proposed TVAG-based model, the attributes without any historical information were treated as attributes with a single timestamp. The value of that timestamp was assigned to the date of creation of the attribute or to the date when the object was created. For example, since there is no information on the time when a user updated their address information $U_{aAddress}$, the assumption is that the address was entered at the time $t = t_{uregistered}$ when the user registered at Facebook. That means that the address attribute $U_{aAddress}(t)$ consists of a single timestamped value:

$$U_{aAddress}(t) = \{(U_{aAddress}, t_{uregistered})\} \tag{3.18}$$

Nevertheless, some attributes are timestamped and even historical information on those attributes is provided. For example, Account Status History can be used to recover $U_{aAccountStatus}(t)$ as a discrete function of time when the status of a user was changed:

$$\begin{aligned} U_{aAccountStatus}(t) = \{ & (U_{aAccountStatus1}, t_1), \\ & ..., \\ & (U_{aAccountStatusn}, t_n)\} \end{aligned} \tag{3.19}$$

The same procedure can be repeated for building $U_{aNames}(t)$ from Name Changes, and for constructing $U_{aIPAddress}(t)$ from Logins and Logouts. As a result of such transformation and recovery of temporal information where possible, the proposed model based on TVAG with the following attributes is constructed:
User attributes U_a:

- Registration Date $U_{aRegistrationDate}$: Time
- Account Status $U_{aAccountStatus}$: Account Status
- About Me $U_{aAboutMe}$: Text
- Favourite Quote $R_{aFavouriteQuote}$: Text
- Hometown $U_{aAboutMe}$: Text
- Current City $U_{aCurrentCity}$: Text
- Address $U_{aAddress}$: Text
- Name U_{aName}: Text
- Alternate Names $U_{aAlternateNames}$: Text
- Screen Names $U_{aScreenNames}$: Text
- Date of Birth $U_{aAlternateNames}$: Time
- Birthday Visibility $U_{aScreenNames}$: Boolean
- Spoken Languages $U_{aSpokenLanguages}$: Text
- Work U_{aWork}: Text
- Vanity URL $U_{aVanityURL}$: Text
- Education $U_{aEducation}$: Text
- Emails U_{aWork}: Text
- Work U_{aWork}: Text
- Linked Accounts $U_{aLinkedAccounts}$: Text
- Gender $U_{aGender}$: Text
- Phone Numbers $U_{aPhoneNumbers}$: Text
- Political Views $U_{aPoliticalViews}$: Text
- Religious Views $U_{aReligiousViews}$: Text
- Status $U_{aStatus}$: Text
- Privacy Settings $U_{aPrivacySettings}$: Text
- Notification Settings $U_{aNotificationSettings}$: Text
- Locale $U_{aLocale}$: Text

- Currency $U_{aCurrency}$: Text
- Credit Cards $U_{aCreditCards}$: List
- Locale $U_{aLocale}$: Text
- Pages You Admin $U_{aPagesYouAdmin}$: List
- Physical Tokens $U_{aPhysicalTokens}$: List
- Networks $U_{aNetworks}$: List
- IP address $U_{aIPAddress}$: Integer
- Location $U_{aLocation}$: Text
- CheckIn $U_{aCheckIn}$: Text
- Search $U_{aSearch}$: Text
- Apps U_{aApps}: Text
- Ad Clicked $U_{aAdClicked}$: Text
- Ad Topics $U_{aAdTopics}$: Text
- Sites Liked $U_{aSitesLiked}$: Text

Relationship attributes R_a:

- Source $R_{aSource}$: UserId1
- Target $R_{aTarget}$: UserId2
- Type R_{aType}: $\{connection, family, following, friends, hiding\}$
- Action $R_{aAction}$: $\{Requested, Accepted, Removed, Deleted, Hidden\}$

Message attributes M_a:

- Source $M_{aSource}$: UserId1
- Target $M_{aTarget}$: UserId2
- Type M_{aType}: $\{chat, message, post, poke, share, note, photo, video, event, group\}$
- Action $M_{aAction}$: $\{Liked, Posted, Poked, Shared, Noted, Joined, Created\}$
- Text M_{aText}: textual message
- Link M_{aLink}: Url link

Below is an example of interaction and relationship graph generation using extracted data for one user. The user registered in June 2009 and has regularly used his Facebook account. As a result, he/she has had $1,212$ conversations. In total, the user has sent $19,549$ messages and received $24,352$ messages. The user has 440 friends, all the friendships are timestamped. The dynamics of sent and received messages for that user is shown in Figure 3.12. The interaction graph for that user is shown in Figure 3.13.

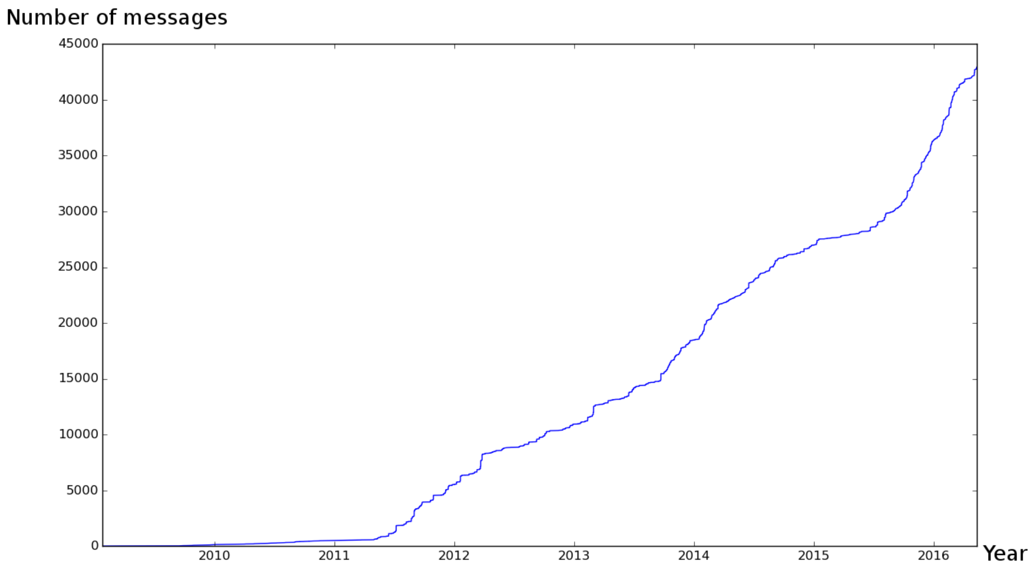


Figure 3.12: The total number of sent and received messages on Facebook for the time from June 2009 to May 2016.

3.5 Conclusion and Discussion

Graphs are a powerful tool for the representation and exploration of social media networks. However, the importance of temporal properties of such graphs is often underestimated. In this Chapter, first, a TVAG was introduced and then a TVAG-based model for modelling human interaction and development of relationships between people was proposed. The time-varying attributes of TVAG allow one to capture the temporal properties of human interaction and relationship development. Existing social networks and platforms can be easily transformed into the proposed model which was demonstrated for three of the most popular social platforms such as Twitter, Facebook, and Stack Exchange. This transformation does not lose any data but allows one to capture and explore temporal user behaviour on the Web. On the other hand, the three platforms of interest ignore the temporal nature of a substantial number of attributes. Nevertheless, regardless of data models and data structures used by a social platform, the TVAG-based model is useful to capture user behaviour on the Web. The main contribution of the proposed TVAG model is the time varying component of features of the graph nodes and edges.

A limitation of the proposed TVAG-based model is the fact that the precision of its mathematical formulation must be improved. However, making the TVAG-based model too formal mathematically may lead to limited flexibility of the model

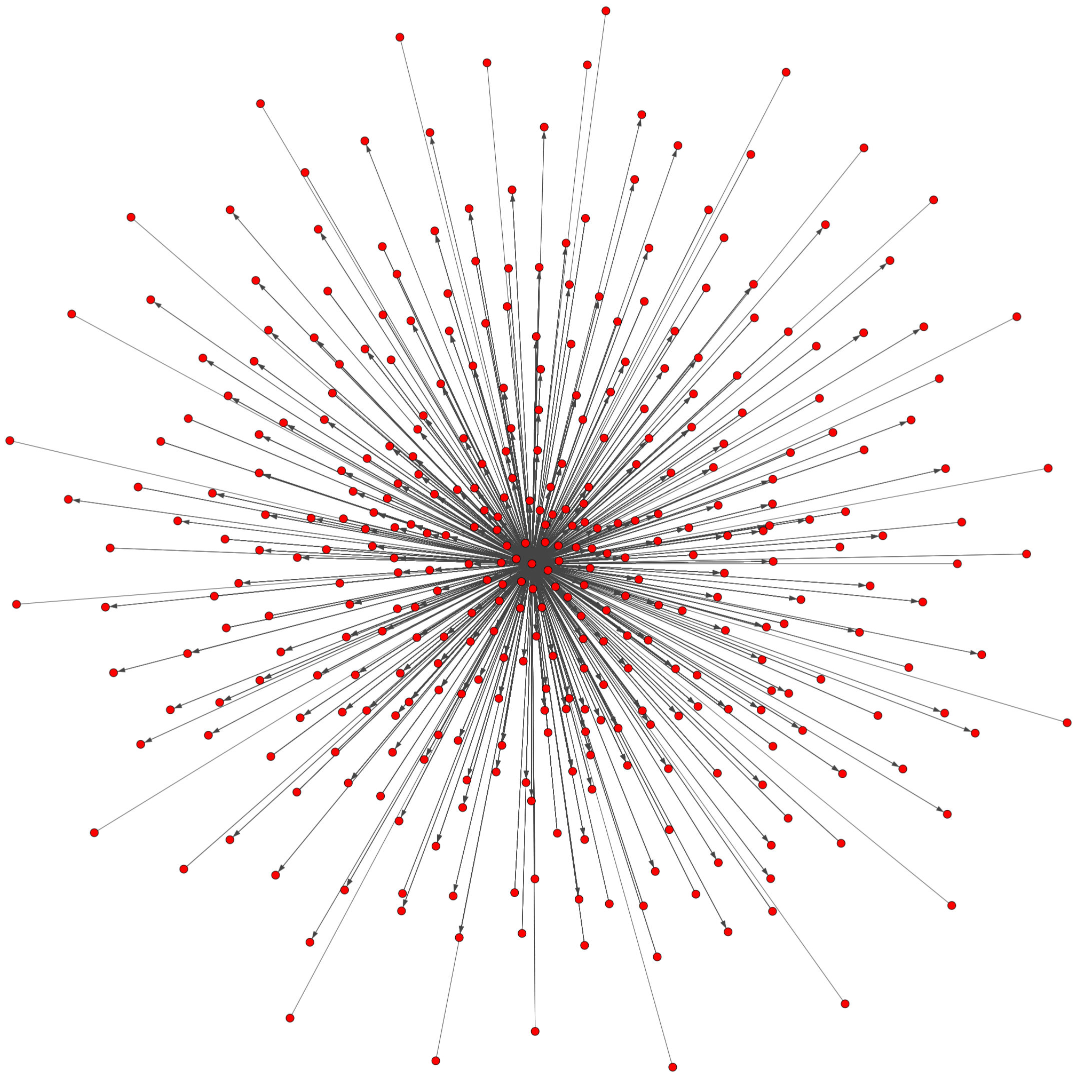


Figure 3.13: The interaction graph for a single user for the time from June 2009 to May 2016. The user is represented as the node in the centre and the people the user interacted with are represented as the nodes around the centre. The distance between the central node and the other nodes is related to the number of messages the users exchanged: the closer are the central node to another node the more messages these two people exchanged.

and also to decreased applicability of the model to real world social networks. The goal of introducing a TVAG-based model is to provide a flexible tool for modeling dynamic and diverse user behaviour on the Web.

First, the proposed model can facilitate visual exploration of user behaviour. Indeed, graph-based models facilitate the exploration of user behaviour on the Web by visualisation due to the fact that a large number of tools for visualisation has been proposed and developed. Even more, visualisation is an important step in data mining which precedes extracting features and then building a predictive model.

Second, the proposed model can facilitate feature extraction for various ML tasks. The proposed model of human interaction and relationship development consists of two TVAGs, an interaction graph $G_{int} = (U, M)$, and a relationship graph $G_r = (U, R)$. Both graphs capture interaction and relationship properties and structure of user behaviour on the Web. In ML, a feature extraction step is an important step which often determines the success of the whole ML process. It was shown in the literature that structural features of human communication are of high impact on many prediction tasks [Cai and Chakravarthy, 2013; Hu et al., 2013; Teinemaa et al., 2015]. However, in real world applications the number of nodes and edges in the graphs G_r and G_{int} may reach millions and it is computationally expensive to calculate features for the whole graph G defined on time interval $\mathbb{T}$. Nevertheless, explicit temporal properties of both nodes and edges allow one to find a smaller subgraph G_T on a time interval $T \in \mathbb{T}$ and calculate the features for this subgraph rather than for the whole graph G. Thus, temporal features of the graph allow one to calculate and then use the structural features even where the graph G is large.

To sum up, this Section introduced a TVAG, then it proposed a model of user behaviour on the Web based on two TVAGs, and, finally, demonstrated how the proposed model can be used for three real world social platforms.

3.6 Summary

A novel graph model, TVAG, and then a TVAG-based model of user behaviour, more specifically a model of human interaction and relationship development on the Web, were proposed in this Chapter. The model captures temporal properties of human communication and relationship development on the Web which facilitates to explore and understand user behaviour. The applicability of the model was demonstrated for three popular social platforms, namely *Facebook*, *Twitter*, and *Stack Exchange.*

In the next Chapter, a procedure for an efficient feature extraction to build predictive models of user behaviour on the Web is discussed.

Chapter 4

Feature Extraction for an Efficient Prediction

In this Chapter the problem of efficient extraction of features for different prediction tasks in the context of user behaviour on the Web is discussed. The methodology of extracting features from the proposed TVAG-based model of human communication is introduced and evaluated. The efficiency of feature extraction is considered in terms of how the extracted features improve the accuracy of a prediction as well as how computationally efficient it is to extract these features.

First, a brief introduction of feature extraction for predicting user behaviour is introduced in Section 4.1. The feature extraction problem is summarised in Section 4.2. Then classification of features semantically and in accordance with their complexity is explored in Section 4.3 and Section 4.4. An experiment investigating how these features affect the performance of predicting user behaviour is presented in Section 4.5. The results of the experiment are discussed in Section 4.6. The results are compared to related studies in Section 4.7. Finally, Section 4.8 and Section 4.9 provide a conclusion and a summary of the Chapter.

4.1 Introduction

Ideally, a predictive model is built using a small dataset without much feature extraction effort. However, in order to improve the accuracy of a model, sometimes one needs to build and extract additional features. From another side, using all the available features is computationally prohibitive for most real world high-dimensional problems.

In this thesis, prediction is considered as a process where the feature extraction step precedes the step of choosing a ML set-up for building a model for predictions. Usually these so-called features are presented in a matrix or in a graph form. Representing a problem space as a graph introduces an additional level of complexity for calculating features based on the graph model [Schenker et al., 2005]. Introducing a hybrid representation can facilitate capturing both structural, graph-related features and other features extracted from the vector model [Markov et al., 2008]. However the computational complexity for extracting features from such a hybrid model can be unacceptable in some practical scenarios. As a result, the following question arises:

1. How can one construct a set of features for an *accurate* but *fast* prediction of user behaviour on the Web?

A list of features potentially affecting the accuracy of predicting user behaviour has been identified in the literature [Yang et al., 2011; Asaduzzaman et al., 2013; Dror et al., 2013; Anderson et al., 2012; Lezina and Kuznetsov, 2012; Burlutskiy et al., 2015; Bhat et al., 2014]. In order to systematise the features, they are classified semantically in this thesis - the features are grouped into users, messages, relationships, temporal, and structural features. Such a classification allows one to systematically analyse the computational complexity of the feature sets as well as their contribution to the accuracy of predictions. However, not much research on time complexity of calculating such features has been conducted. An attempt to estimate the complexity of features was performed in [Page et al., 2014]. Nevertheless, there are many possible combinations of features and extracting the features often involves expensive calculations.

A comparison of training and testing times for different ML algorithms was performed in [Lim et al., 2000]. However, the authors did not address the question of how the features influence the accuracy and time of predictions. Even though many works have shown high accuracy for different predicting tasks, the time and complexity of the trained models are usually not of main concern. In order to address this issue, the time and space complexity of features influencing user behaviour is estimated in this thesis. As a result, a classification based on semantic and complexity properties of the features is proposed.

In order to answer the first research question, a few experiments on three different datasets are conducted. First, the complexity of feature extraction for a predictive model is estimated. Second, the accuracy contribution of the classified feature sets is evaluated. As a result, the usefulness of the proposed feature classification into semantic and computational complexity groups is discussed and a guideline for feature extraction is proposed.

The last question arises from the fact that temporal properties of human communication are often missing in many social platforms for communication [Bronson et al., 2013]. However, using temporal properties can facilitate to decrease the size of the data for extraction as well as extract temporal features to improve the accuracy of a prediction. This, as a result, can provide an accurate prediction as well as it can help to reduce computational efforts significantly. The second question is formulated as follows:

2. How important are temporal features for prediction in terms of accuracy and time efficiency?

To answer this question, a set of temporal features is selected and then a model is trained using all features and then all features except temporal features. Also, the time required to extract those features is captured and compared to the time required to extract other features.

4.2 Feature Engineering

In this thesis, the features for prediction are represented as a feature space F. Having a model of human communication and relationship development in the graph form $G = (U, E)$ proposed in Chapter 3, the transformation of the graph G into the feature space $F = \{F_1, .., F_n\}$, where n is the number of features, can be written as:

$$F = f(G) \tag{4.1}$$

where f is a function which maps the graph G into the feature space F.

Selection of the features F for predictions is a challenging task. The choice of features used for the prediction is mainly influenced by the availability of data in the graph model as well as by the features successfully used in other predictions of user behaviour [Yang et al., 2011; Dror et al., 2013; Asaduzzaman et al., 2013; Bhat et al., 2014].

4.3 Feature Semantics

The features F can be divided in several groups depending on the semantics and the elements of the graph G the features are associated with. The graph G has nodes representing users U, edges E representing messages M and relationships R between the users U, time t associated to users U, messages M, and relationships R. Finally, the graph model G allows one to calculate structural features (see Figure 4.1). Thus, the transformation of the graph G into the feature space F can be rewritten as a union of subspaces of these features:

$$F = f_1(U) \cup f_2(M) \cup f_3(R) \cup f_4(t) \cup f_5(G) \tag{4.2}$$

Rewriting the transformations $\{f_1, f_2, f_3, f_4, f_5\}$ of the graph G, namely *Users* features as F_u, *Messages* as F_m, *Relationships* as F_r, *Temporal* as F_t features, and *Structural* as F_s leads to five types of features F_i. The features F_u are extracted from the nodes U of the *graph* G, features X_m are extracted from the edges M of the *interaction graph* G_{int}, the features F_r are derived from the edges of the *relationship graph* G_r, and the features F_t are the temporal properties of the graph G. Finally, F_s are the features representing structural properties of the graph G:

$$\begin{aligned} F_u &= f_1(U) \\ F_m &= f_2(M) \\ F_r &= f_3(R) \\ F_t &= f_4(t) \\ F_s &= f_5(G(t)) \end{aligned} \tag{4.3}$$

The feature vector F is a union of all these features,

$$F = F_u \cup F_m \cup F_r \cup F_t \cup F_s \tag{4.4}$$

Subsections 4.3.1-4.3.5 provide a list of these user F_u, message F_m, relationship F_m, structural F_s, and temporal F_t features which are then used for training a predictive model.

4.3.1 User Features

The user features F_u are derived from the attributes of the nodes U of the graph G:

1. *User Id*: A unique number associated with every user;
2. *User Personal Info*: A feature representing information on a user. A user u can have a multitude of user personal info features, for example:
   (a) *User's Name*: The name of a user;
   (b) *User's Gender*: The gender of a user;
   (c) *User's Location*: User's location $L(u_i)$, it can be the name of his/her hometown or his/her country;
   (d) *User's Longitude*: The longitude of user's location;

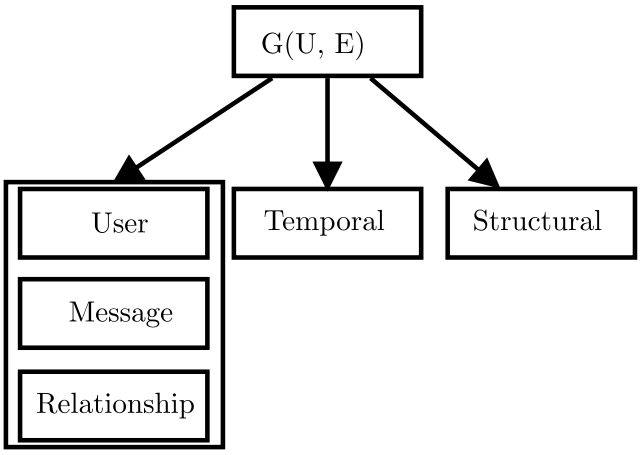


Figure 4.1: Extracting features from a graph model.

(e) *User's Latitude*: The latitude of user's location;
(f) *User's Time zone*: User's time zone can be derived from their location or by identifying the place from where users post messages: $T(u_i) = f(L(u_i))$ where $L(u_i)$ is the location of the user u_i;

4.3.2 Message Features

The message features F_m are derived from the attributes of the edges M of the interaction graph G_{int}. There are three attributes of these edges, namely *source, destination*, and *message.* The first two attributes:

1. *Source User*: The ID of a user who sent the message;
2. *Destination User*: The ID of a user who received the message;

A possible list of features F_m extracted from messages M is below:

3. *Type*: The type of the message;

4. *The number of sent messages*: The number of sent messages for a particular user u_i at a time moment t_e. $N_m = outdeg(u_i, t_e)$;

5. *The number of messages received by a user*: The number of received messages for a particular user u_i at a time moment t_e. $N_m = indeg(u_i, t_e)$;

6. *The number of answered messages*: The number of sent messages which received at least one answer for a particular user u_i at a time moment t_e is $N_a = outdeg(u_i, t_e)$;

7. *Textual features*: Features related to a textual body of a message. Such features may include various NLP features, for instance:
   (a) *Text length*: The number of letters in the message m;
   (b) *Occurrence of a word*: For example, the number of 'who' words in the text;
   (c) *Occurrence of a set of words*: For example, the number of 'wh' words in the text;

A common formalisation for a textual message is a vector space model, for example, the bag of words and phrases [Le and Mikolov, 2014]. However, the bag of words and phrases representation is limited to frequency features of these words and phrases. Nevertheless, structural and semantic features can be extracted from the text as well. One of the main issues is the complexity of such computations. For example, the occurrence of a word or a set of words can include the following:

- *The number of active verbs*: the total number of verbs in the message;
- *The number of times of self referencing*: the number of times a user mentioned themselves, for example, 'we', 'I', 'me';
- *The number of url links*: the number of url links in the text;
- *The number of images*: the number of images in the text;
- *The number of tags*: The number of tags the message is tagged with. For example, Twitter and Stack Exchange allow users to tag their messages, tweets or questions correspondingly.

4.3.3 Relationship Features

The relationship features F_r are derived from the nodes R of the relationship graph G_r. There are three attributes of these nodes, namely *source, destination, relationship.* The first two attributes:

1. *Source User*: The ID of a source user;
2. *Destination User*: The ID of a target user;

Relationships formed between the users U can be friendship, family, work ties. For example, the following features can be extracted:

3. *Type*: The type of a relationship;
4. *The number of outcoming relationships*: The number of users who follow the user u;

5. *The number of incoming relationships*: The number of following users by the user u;

The relationship graph G can have several types of relationships. For example, Facebook allows one to have friendship relationships and blood relationships as well.

4.3.4 Structural Features

Structural features F_s are the features which are related to the topology of the interaction graph G_{int} and the relationship graph G_r. These features include centrality measures that identify the importance of nodes in the graphs. Below is a list of the measures investigated in this thesis:

1. *Degree centrality* $C_D(u_i)$ of a node u_i is the number of edges e this node has:

$$C_D(u_i) = deg(u_i) \tag{4.5}$$

   Where $deg(u_i) = outdeg(u_i) + indeg(u_i)$ since the graph G is a directed graph.

2. *Eigenvector centrality* $C_E(u_i)$ of a node u_i can be calculated as following:

$$C_E(u_i) = \frac{1}{\lambda} \sum_k a_{k,i} C_E(u_k) \tag{4.6}$$

   Where λ is a constant, $a_{k,i}$ is the $(k,i)^{th}$ entry of the adjacency matrix A of the graph G.

3. *PageRank*, a form of eigenvector centrality, was introduced by Sergey Brin for use in their search engine [Page et al., 1999]. The PageRank $y(u_i)$ of a node u_i can be calculated as:

$$y(u_i) = \alpha \sum_{j \in U} \frac{\omega_{ji}}{k_j} y(u_j) + \beta \tag{4.7}$$

   Where $\alpha \geq 0$ and $\beta \geq 0$, $\alpha + \beta = 1$ are the parameters for calculating the importance of the node i; ω_{ji} is a weight coefficient associated with the j, i^{th} node, and k_j is a coefficient reflecting the importance of linkage.

4. *Closeness centrality* $C(x)$ of a node is the average distance from the node to all other nodes:

$$C(x) = \frac{1}{\sum_{y \neq x} d(y,x)} \tag{4.8}$$

Where $d(y, x)$ is the distance between a node x and y.

5. *Betweenness centrality* $C_B(u)$ of a node reflects the frequency of the node to lie on short paths between other pairs of nodes. There is a path between nodes u_i and u_j if there exists a sequence of nodes connected by edges from the node u_i to the node u_j. A shortest path σ_{jk} is a path consisting of the smallest number of nodes in such a sequence. There may be zero or more than one shortest paths between two nodes u_i and u_j. Finally, *betweenness centrality* can be defined as:
$$C_B(u) = \sum_{j,k \in U} \frac{\sigma_{jk}(i)}{\sigma_{jk}} \tag{4.9}$$
Where σ_{jk} is the number of (j, k)-shortest paths and $\sigma_{jk}(i)$ is the number of the shortest paths laying via the node i.

These structural features F_s can be calculated for both the interaction graph G_{int} and the relationship graph G_r.

4.3.5 Temporal Features

In the proposed TVAG-based model G, time is explicit which allows one to extract various temporal features F_t. Below is a list of temporal features which can be used for predictions:

1. *Timestamped features* F_{ts}: these features represent the time-varying properties of the graph G;
    (a) F_{ts1} is the time when a message m was posted can include the year, month, week, day, minute, second. This is a timestamped feature;
2. *Duration* F_{td} is the difference between two timestamped events, for example the length of time a friendship lasts, the time taken for a message to be sent and delivered, how much time has passed since two people met each other.
    (a) F_{td1} is the time since registration on the Web: $T_{onsite}(u_i) = T_{now} - T_{registered}(u_i)$ where T_{now} is the current time and $T_{registered}$ is the moment in time when the user u_i registered on the Web. This is a duration feature;
3. *Frequency* F_{tf} can be constructed from a list of timestamps for an edge or a node. This feature can uncover patterns in edge or node occurrences. Indeed, periodicity is present in certain datasets. For example, the following feature is an instance of the frequency feature F_{tf}:

(a) F_{tf1} is an average messaging activity. An example of such activity can be the average response time $T_{av}(u_i)$ for a user u_i which can be calculated as an arithmetic sum of all message response times $T_j(u_i)$ for this user u_i divided by the total number of messages n: $T_{av}(u_i) = \frac{1}{n}\sum_{j=1}^{n} T_j(u_i)$;

4. *Local Frequency* F_{tl}: similar to F_{tf} with the difference that the frequency is calculated for a time interval T.

   (a) F_{tl1} *the number of messages in the period* T: instead of calculating the number of messages for the whole period T_w, the number is calculated only for the period $T \subset T_w$. This is a local frequency feature;

   (b) F_{tl2} *the number of established relationships in the last period* T: instead of calculating the number of relationships for the whole period T_w, the number is calculated only for a period $T \subset T_w$. This is a local frequency feature;

   (c) F_{tl3} *a pattern of temporal messaging activity by a time period* T: an example of such a feature can be a vector with m entries where each entry represents the number of messages for each time period T. This is a local frequency feature;

Temporal features can represent the properties of the users U, messages M, and relationships R between the users.

4.4 Feature Complexity

Feature engineering can significantly improve accuracy and time efficiency of a predictive model. However, calculating features can be computationally expensive. The complexity of feature extraction depends on the data structure used for storing data. The complexity of the features can be evaluated in big O notation. Thus, the features can be divided into a few groups according to their big O complexity. In this thesis, the features are classified into three groups, namely raw, easy, and hard features (see Figure 4.2).

Raw features are all input features which can be used without any computations. These can include, for example, the time when a user posted a message, a photo they posted, the text describing the photo. Sometimes these features can be fed into a ML algorithm directly. However, in some scenarios there is a need to modify the features through, for example, dimension reduction, normalisation, sampling, binarisation (see Subsection 2.5.2).

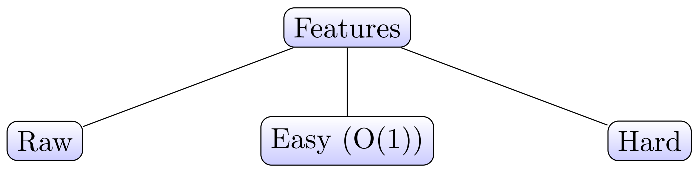


Figure 4.2: Classification of the features. Raw features - no calculations, easy features - requires one to calculate features with complexity of $O(1)$, hard features - computational complexity more than $O(1)$.

Table 4.1: Computational complexity of user features.

F_n	**Description**	**Comments**	**Computational Complexity**
F_1	User Id	A unique number associated with every user	Raw
F_2	User Personal Info	This may include user's name, gender, location, latitude, longitude, time zone	Raw or $O(1)$ for a HashTable implementation

4.4.1 Computational Complexity

Raw features are used for calculating easy and hard features. Easy features are the features which can be calculated from the raw features in constant time, thus, their complexity is $O(1)$. Hard features are the features which complexity is higher than constant time. For example, easy features involve calculating the number of messages m_i for a user u_i. If each message has a unique id and all messages are sorted then calculating such a feature takes constant time $O(1)$. An example of a hard feature is calculating the occurrence of a phrase in all messages or in a subset of messages. The time complexity of calculating this feature is linear, $O(kn)$, where n is the number of messages and k is the number of words in the phrase.

User features include a raw feature *User Id* and a set of features providing information about the user. These features can be raw or, in case of being stored in a hashtable, can be calculated in constant time (see Table 4.1).

Message features include two mandatory raw features, *Source User* and *Destination User* (see Table 4.2). Also, message features have a raw feature F_3, *Type* of a message. The features $F_4 - F_8$ are NLP features and require processing text messages and calculations. The features F_4 - F_8 can be calculated for a single message m or for a set of messages m. In case when these features are for a set of k messages then the complexity of calculating these features increases by the number of messages k. The features $F_9 - F_{11}$ are reflecting the qualitative characteristics of users messaging each other.

Relationship features include two mandatory raw features *Source User* and *Destination User* (see Table 4.3). Also, relationship features have a raw feature F_3, *Type* of a relationship, as well as easy features $F_4 - F_5$ reflecting the qualitative characteristics of relationships between users.

Structural features are the most expensive to calculate (see Table 4.4). For example, the complexity of calculating the feature F_1, *degree centrality* for all nodes U, is $\Theta(U^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation, where E is the number of edges. The complexity for calculating betweenness centralities is $\Theta(U^3)$ or $O(U^2 \log U + UE)$ for sparse representations [Brandes, 2001].

Finally, the computational complexity of temporal features are presented in Table 4.5. These features include timestamped features (F_1), duration and frequency of events (features $F_2 - F_4$), as well as local frequency features (feature F_5).

The computational complexities in Tables 4.1 - 4.5 include constants in front of their Big O notations. These constants do not really matter for a formal analysis of asymptotic behaviour of such functions. However, these constants are shown for practicians interested in running a particular implementation in the code. As a result, these constants can lead to a large overhead or, for example, affect how the functions perform for smaller input sizes.

In this thesis, human communication and relationship development is modelled as a TVAG-based graph in order to capture dynamics as well as the topological properties of such communication and relationship development. For extracting features, first, querying a TVAG-based data structure is performed and then calculating the features is executed. The complexity of such operations depends on the data structure used for storing the data. Currently the problem of representing data as a TVG is an area of active research [Cattuto et al., 2013; Semertzidis and Pitoura, 2016]. The proposed TVAG model is based on TVG. Thus, the computational complexity of TVAG operations are corresponding to the complexity of TVG [Cattuto et al., 2013; Semertzidis and Pitoura, 2016].

4.4.2 Space Complexity

All *raw* features are also classified into *light* and *heavy* (see Figure 4.3). If a feature is less than M bytes, for example, M is 128 bytes and the feature is 64 bytes, then it is defined as a *light* feature [1]. Thus, if a feature requires $m < M$ bytes to be

[1]It is important to mention that this classification is applied across all samples: if there exists a feature greater or equal to M bytes even for a single sample, then this feature is treated as a *heavy* feature for all samples.

Table 4.2: Computational complexity of message features.

F_n	**Description**	**Comments**	**Computational Complexity**
F_1	Source User	The ID of a user who sent the message	Raw
F_2	Destination User	The ID of a user who received the message	Raw
F_3	Type	For example, a message can be a question, a tweet, or a reply	Raw
F_4	Text length	The number of letters in the message m	kO(1) where k is the number of letters in a word
F_5	Occurrence of a word	For example, the number of 'who' word in the text	O(1) for a HashTable implementation
F_6	The most frequent word	For example, finding the most popular word in a text	$O(n \log(n))$ for sorting and then O(1) for a HashTable implementation
F_7	The occurrence of a set of words	For example, the number of 'wh' words in the text	mO(1) for a HashTable implementation, where m is the number of words
F_8	The most frequent set of words	For example, the most popular words in a text	$O(n \log(n))$ for sorting and then kO(1) for a HashTable implementation, where k is the number of words in the set
F_9	The number of messages sent by a user	The number of messages for a particular user u_i at a time moment t_s. $N_m = outdeg(u_i, t_s)$	$O(\|E\|)$ where $\|E\|$ is the number of edges-messages
F_{10}	The number of messages received by a user	The number of messages for a particular user u_i at a time moment t_s. $N_m = indeg(u_i, t_s)$	$O(\|E\|)$ where $\|E\|$ is the number of edges-messages
F_{11}	The number of messages answered by a user	The number of received messages with a feedback for a particular user u_i at a time moment t_s. $N_a = outdeg(u_i, t_s)$	$O(\|E\|)$ where $\|E\|$ is the number of edges-messages

Table 4.3: Computational complexity of relationship features.

F_n	**Description**	**Comments**	**Computational Complexity**
F_1	Source User	The ID of the source user	Raw
F_2	Destination User	The ID of the destination user	Raw
F_3	Relationship Type	For example, a relationship can be friendship or blood relation	Raw
F_4	The number of incoming relationships	The number of users who is connected by a relationship to the user u	$O(\|E\|)$ where $\|E\|$ is the number of edges-relationships
F_5	The number of outcoming relationships	The number of users to whom the user u is connected by a relationship	$O(\|E\|)$ where $\|E\|$ is the number of edges-relationships

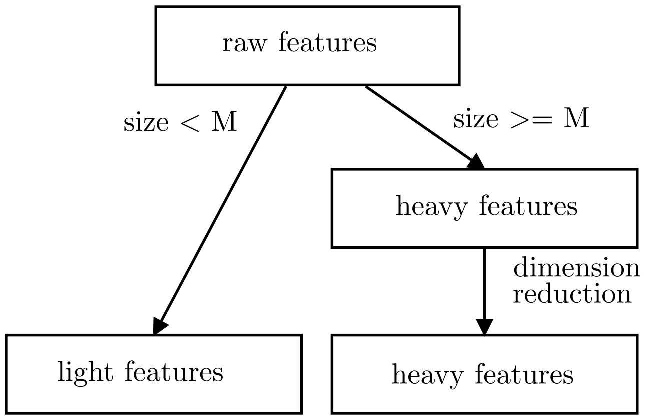


Figure 4.3: Processing raw features into two groups, light and heavy features with dimension reduction.

stored then it is a *light* feature. For example, if a feature is a text or a picture of $m \geq M$ bytes then it is considered to be *heavy*. In that case some sort of dimension reduction can be applied. For instance, in case of textual data, a *'bag of words'*, term frequency, and phrase extraction can be used for dimension reduction.

In order to demonstrate a usage of the proposed classification for feature extraction, an experiment is conducted. This experiment is described below.

4.5 Experiment Outline

In this experiment, prediction tasks are formulated as a classification task where the fact of an event e to happen is predicted. The fact of the event e to happen is predicted using the features F, where $F = f(G(t))$, and $G(t) = (U(t), E(t))$ is the graph representing user behaviour, $U(t)$ are the nodes representing users, $E(t)$

Table 4.4: Computational complexity of structural features.

F_n	**Description**	**Comments**	**Computational Complexity**
F_1	Degree centrality $C_D(u_i)$	The number of users who follow the user u	$\Theta(U^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_2	Weighted degree centrality	The number of following users by the user u;	$\Theta(U^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_3	Eigenvector centrality $C_E(u_i)$	The number of users who follow the user u	$\Theta(U^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_4	PageRank	The number of following users by the user u;	$\Theta(U^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_5	Closeness centrality	The number of users who follow the user u	$\Theta(U^3)$ or $O(U^2 \log U + UE)$ for sparse representations.
F_6	Betweenness centrality	The number of following users by the user u;	$\Theta(U^3)$ or $O(U^2 \log U + UE)$ for sparse representations.

Table 4.5: Computational complexity of temporal features.

F_n	**Description**	**Comments**	**Computational Complexity**
F_1	Message timestamp	The time when a message m was sent by a user u	O(1) for each message
F_2	Time passed since a user u registered on the Web	$T_{onsite}(u_i) = T_{now} - T_{registered}(u_i)$	O(1) for each message
F_3	The average messaging activity of a user u	$T_{av}(u_i) = \frac{1}{n}\sum\limits_{j=1}^{n} T_j(u_i)$ where $T_j(u_i) = t_0(u_i(M_{aj})) - t_0(u_i(M_{qj}))$ is the time for the i^{th} message by a user	O(n) where n is the number of messages
F_4	The number of messages by a user during the period T	instead of calculating the number of messages for the whole period T_w, the number is calculated only for a period $T \subset T_w$	Assuming that the messages are sorted by time, O(n) where n is the number of messages during the period $T - T_{now} \leq t \leq T_{now}$
F_5	The frequency of temporal activity by a time period T	a vector with n_{act} entries where each entry represents the number of messages for different time periods $T_i \subset T$	n_{act}O(n) where n is an average number of activities during a time period T_i

are the edges representing messages M and relationships R between the users. The information on the graph $G(t)$ spans from T_s to T_f and the event e can happen within the interval $[T_s, T_f]$ or outside the interval. First, a set of semantically grouped features F is extracted and then their complexity is evaluated. Second, the time required to extract the features is analysed, and the contribution of these features to the accuracy of predictions is also estimated. This process is executed for three different prediction tasks across different datasets, two of the largest datasets from *Stack Exchange* websites and a dataset from *Twitter* platform:

1. *Stack Exchange*: Predicting users' response time on the Web, particularly in the context of the largest Q&A community, Stack Exchange;
2. *Twitter*: Predicting whether a tweet will be retweeted in 24 hours or not;

Four ML algorithms, namely k-NN, LR, DT, and XGB are chosen for predictions. The choice is dictated by the fact that, first, all four ML algorithms are different in their nature. Second, these algorithms showed promising results for predicting user behaviour in the literature [Lezina and Kuznetsov, 2012; Burlutskiy et al., 2015; Bhat et al., 2014]. These algorithms are discussed in details in Section 2.6.

First, the proposed methodology is tested for predicting users' response time on the Web, particularly in the context of the largest Q&A community, Stack Exchange.

4.5.1 Stack Exchange: Stack Overflow and Stack Maths

Online Q&A websites are popular platforms where users can ask questions and then receive answers. However, many users are left frustrated since their questions remain unanswered or answered after a long time. Thus, a prediction task of predicting users' response time is formulated.

Prediction Task

To predict whether a question q will receive an answer a within the time T or whether it will take longer than this time T:

$$t(a) \in [t(q), t(q) + T] \text{ or } t(a) \in (t(q) + T, \infty), \tag{4.10}$$

where $t(q)$ is the time when the question q was asked, $t(a)$ is the time when the answer a for this question q will be received, and T is the duration of a period of time, for example, 1 hour. In the experiment $T = T_{median}$ where T_{median} is the median time of answering questions. Choosing the median time leads to balanced data.

Dataset

Stack Exchange datasets are available on-line and represent raw archived *xml* files [StackExchange, 2014]. The dataset consists of 127 websites on different topics. The timespan for these websites is from July 31, 2008 to October 23, 2014. The whole list of available attributes of the dataset is presented in Appendix A. At the moment of writing this thesis, *Stack Exchange* was one of the most popular and fastest growing Q&A platform which was why it was chosen for the experiments. The total amount of raw data in *xml* format was ∼156 GB.

For this experiment, the two largest datasets from Stack Exchange are chosen, Stack Overflow (SO) website and Stack Maths (SM).

Feature Engineering

The raw archived *xml* data is transformed into a TVAG model and then 41 features including 9 user features (see Table B.1), 19 message features (see Table B.2), 11 temporal features (see Table B.3), and 2 structural features (see Table B.4) are extracted. The feature F_{11}, answerer ID, is excluded since it provides the information about an answerer in the future which is a part of the prediction task. Stack Exchange websites do not have a social network, thus, there is no relationship features to extract. The time required for extracting each group of features is recorded and the complexity for each feature is estimated. As a result, the features are classified into semantic and computational complexity groups. Finally, four different ML algorithms are executed for all features and then for all features except one group in order to determine the contribution of a feature group to the overall accuracy.

4.5.2 Twitter

Twitter is the largest micro-blogging platform where users tweet and retweet others. Some tweets become viral and thousands of people retweet them, however, some tweets are never retweeted. In the second experiment, a problem of predicting whether a tweet will be retweeted within 24 hours or not is formulated. To predict whether a tweet q will be retweeted within the time T or whether it will take longer than this time T:

$$t(a) \in [t(q), t(q) + T] \text{ or } t(a) \in (t(q) + T, \infty) \tag{4.11}$$

Where $t(q)$ is the time when the tweet q was asked, $t(a)$ is the time when the tweet t is retweeted, and T is the duration of a period of time; in this experiment the duration T is equal 24 hours.

Dataset

For the second experiment, a Twitter dataset is used. The dataset is taken from [De Domenico et al., 2013] where the authors extracted tweets by hashtags *lhc*, *cern*, *boson*, *higgs*. The choice of these hashtags was dictated by the fact that the researchers wanted to extract tweets about the discovery of a boson at CERN in July 2012. The tweets are filtered by date so only the tweets from $00:00$ AM 1^{st} July 2012 to $11:59$ PM 7^{th} July 2012 are considered. As a result $985,692$ tweets are extracted. This data allows one to build both relationship and interaction graphs since the data have information on users, messages (tweets), and relationships between users (who follows whom).

Feature Engineering

Analogically to Stack Exchange datasets, 1 user feature, 9 message features, 5 relationship features, 4 structural features, and 9 temporal features are extracted from relationship and interaction graphs. The list of extracted features with their corresponding complexities is represented in Tables B.5, B.6, B.7, B.8, B.9. The feature F_4, the type of a message, is excluded from the training set since it is the feature for prediction. As a result, there are 27 features in the training set. The feature F_4 is transformed into a binary label where 1 is for retweets and 0 is for all other kind of tweets.

As a result, all features are classified into semantic groups, namely user, message, relationship, temporal, and structural features, and computational complexity groups including raw, easy, and hard features. Finally, four different ML algorithms are executed using all extracted features and then for all features except a single group, a semantic group or a computational complexity group, in order to determine the contribution of that group to overall accuracy.

4.5.3 Varying Class Balance

The label for prediction for Stack Exchange datasets was derived from users' response time, all times less than median time formed '0' and more than median time lead to '1'. Thus, the classes are balanced, 50%/50%. However, in order to understand how the balance between classes influences the accuracy of the prediction, the proportion between the two classes for Stack Overflow dataset was varied. Variation in the proportion between classes was achieved by varying the response time T. The first class ('1') consists of all questions answered faster than the time T, and the second class ('0') consists of the questions answered slower than the time T. The proportions between the classes were chosen to comply with the following five scenarios:

1. The second class prevails: $T = 5$ minutes or 15%/85% - only 15% of the questions were answered within 5 minutes;
2. The classes are balanced: 26 minutes or 50%/50%;
3. The classes are nearly balanced: 1 hour or 61%/39%;
4. The first class prevails: 1 day or 84%/16%;
5. The first class significantly prevails: 1 month or 96%/4%.

4.6 Results and Discussions

The resulted contributions to the accuracy for different semantic groups of features are shown in Table 4.6. The resulted contributions to the F1 score for different semantic groups of features are shown in Table 4.7. The resulted contributions to the accuracy for different computational complexity groups of features are shown in Table 4.8 and Figure 4.4. The resulted contributions to the F1 score for different computational complexity groups of features are shown in Table 4.9 and Figure 4.5. Tables 4.6, 4.7, 4.8, 4.9 also summarise information on the quantity of features in each group, the time required to extract these features, and the accuracy or F1 score gain of each group to the overall accuracy. The accuracy and F1 gain are calculated in comparison to a naive approach where prediction for each class results in 50% accuracy. Finally the results for varying class balance for Stack Overflow data are shown in Figure 4.6.

First, the results for varying semantic groups of features and the performance of predictions are discussed.

Table 4.6: Accuracy for semantic groups of features for Stack Overflow (SO), Stack Maths (SM), and Twitter.

Dataset	**Features**	**Qty**	**Time, s**	**Accuracy Gain, %**			
				k**-NN**	**LR**	**DT**	**XGB**
SO	User	9	~788	+1.1	+0.3	-0.4	+2.0
	Message	18	~232,000	+4.2	+3.1	+4.5	+2.2
	Relationship	-	-	-	-	-	-
	Temporal	11	~65,300	**+8.1**	**+9.1**	**+7.4**	**+11.1**
	Structural	2	~82,300	+4.2	+3.1	+2.5	+1.3
	All features	41	~380,400	**+10.5**	**+14.2**	**+15.2**	**+16.1**
SM	User	9	~83	+0.7	+1.2	+1.0	-0.6
	Message	18	~35,100	+3.7	+4.1	+3.7	+2.8
	Relationship	-	-	-	-	-	-
	Temporal	11	~9,800	**+9.1**	**+7.3**	**+6.9**	**+9.4**
	Structural	2	~11,200	+3.1	+4.0	+3.2	+2.4
	All features	41	~56,200	**+11.2**	**+10.6**	**+13.1**	**+15.0**
Twitter	User	1	-	+0.2	+0.1	-0.7	+1.1
	Message	8	~45	+1.5	+3.2	+1.2	+4.0
	Relationship	5	~321	+2.3	+2.5	+3.1	+4.0
	Temporal	9	~135	+4.2	+5.3	+6.3	+8.7
	Structural	4	~21,100	**+18.1**	**+18.5**	**+22.3**	**+37.2**
	All features	27	~21,600	+12.5	+10.1	**+20.1**	**+38.5**

4.6.1 Semantic Variations of Features and Performance

First, it can be noticed from Table 4.6 and Table 4.7 that calculating features, especially structural features, takes significant time. For example, for Stack Overflow with nearly two million entries, calculating structural features took almost a day (23 hours). Nevertheless, calculating message features for Stack Overflow and Stack Maths also took significant time, almost three times more than for structural features. However, calculating message features can be parallelised with help, for example MapReduce techniques. On the contrary, parallelising calculation of structural features is problematic. Calculating temporal features took a relatively short time compared to structural and message features for Twitter but comparable time

Table 4.7: F1 score for semantic groups of features for Stack Overflow (SO), Stack Maths (SM), and Twitter.

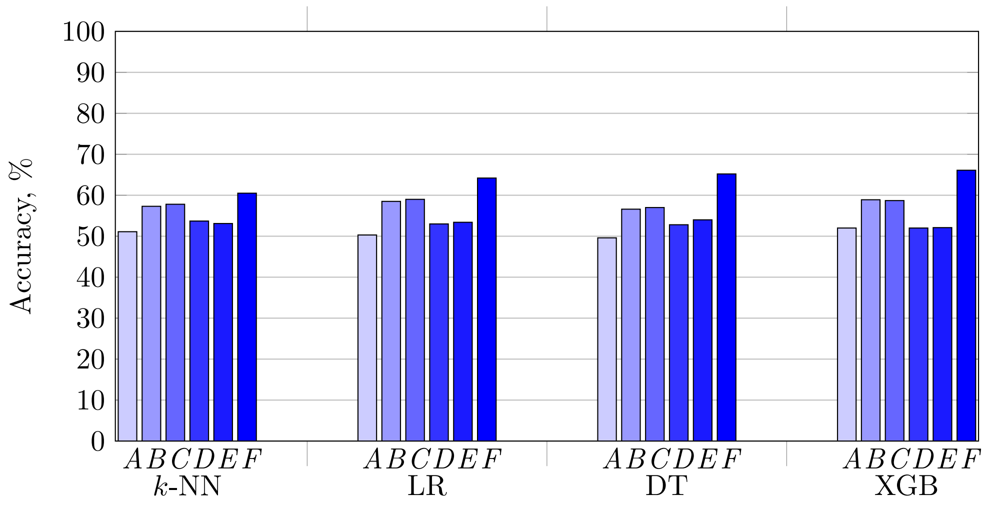
Dataset	**Features**	**Qty**	**Time, s**	**F1 score Gain, %**			
				***k*-NN**	**LR**	**DT**	**XGB**
SO	User	9	~788	+0.1	-0.6	+1.1	+1.2
	Message	18	~232,000	+5.3	+4.2	+3.8	+2.4
	Relationship	-	-	-	-	-	-
	Temporal	11	~65,300	**+7.8**	**+8.3**	**+8.7**	**+10.8**
	Structural	2	~82,300	+1.4	+2.7	+2.6	+2.1
	All features	40	~380,400	**+9.5**	**+12.2**	**+14.7**	**+15.6**
SM	User	9	~83	+0.5	+1.0	-0.3	+0.1
	Message	18	~35,100	+3.3	+2.8	+3.8	+2.3
	Relationship	-	-	-	-	-	-
	Temporal	11	~9,800	**+8.7**	**+8.1**	**+7.0**	**+8.2**
	Structural	2	~11,200	+1.9	+3.7	+3.5	+3.0
	All features	40	~56,200	**+10.0**	**+11.1**	**+12.8**	**+15.4**
Twitter	User	1	-	-0.5	-0.9	+1.0	+1.1
	Message	8	~45	+1.2	+2.7	+1.2	+3.2
	Relationship	5	~321	+1.9	+2.2	+2.8	+3.6
	Temporal	9	~135	+5.7	+5.1	+2.7	+7.9
	Structural	4	~21,100	**+17.2**	**+16.8**	**+21.0**	**+38.0**
	All features	27	~21,600	+13.1	+11.0	**+21.2**	**+38.7**

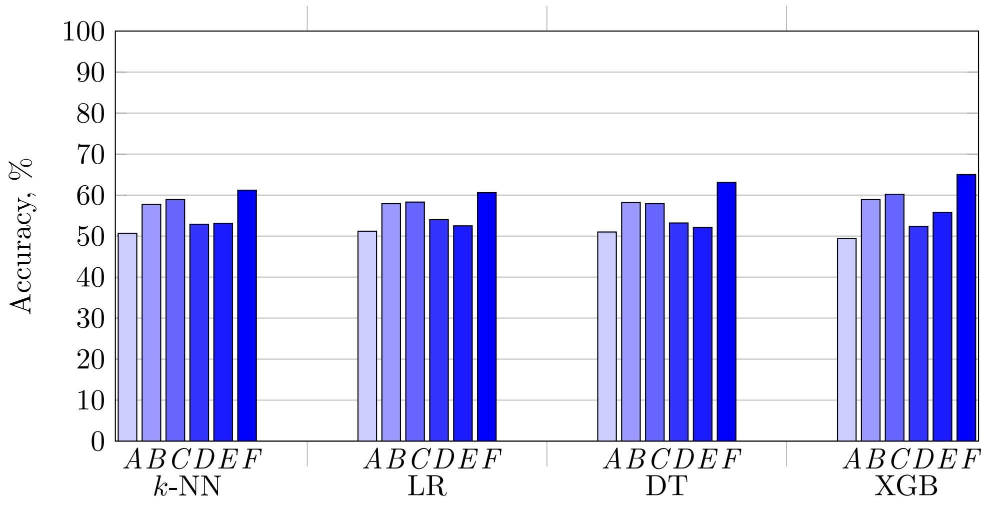
Table 4.8: Accuracy for computational complexity groups of features for Stack Overflow (SO), Stack Maths (SM), and Twitter.

Dataset	**Features**	**Qty**	**Time, s**	**Accuracy Gain, %**			
				k-**NN**	**LR**	**DT**	**XGB**
SO	All raw	9	-	+1.1	+0.3	-0.4	+2.0
	All easy	27	~66,900	+7.3	+8.5	+6.6	+8.9
	All raw + easy	36	~66,900	+7.8	+9.0	+7.0	+8.7
	All hard	4	~313,500	+3.7	+3.0	+2.8	+2.0
	All raw + hard	13	~313,500	+3.1	+3.4	+4.0	+2.1
	All features	40	~380,400	**+10.5**	**+14.2**	**+15.2**	**+16.1**
SM	All raw	9	-	+0.7	+1.2	+1.0	-0.6
	All easy	27	~11,400	+7.7	+7.9	+8.2	+8.9
	All raw + easy	36	~11,400	+8.9	+8.3	+7.9	+10.2
	All hard	4	~44,800	+2.9	+4.0	+3.2	+2.4
	All raw + hard	13	~44,800	+3.1	+2.5	+2.1	+5.8
	All features	40	~56,200	**+11.2**	**+10.6**	**+13.1**	**+15.0**
Twitter	All raw	7	-	+0.2	+0.1	-0.7	+1.1
	All easy	15	~180	+1.4	+2.0	+1.1	+4.2
	All raw + easy	22	~180	+1.5	+1.9	+0.4	+3.2
	All hard	6	~21,400	+18.1	+18.2	+19.3	+37.2
	All raw + hard	13	~21,400	**+18.5**	+18.5	+18.2	+37.1
	All features	27	~21,580	+12.5	+10.1	**+20.1**	**+38.5**

Table 4.9: F1 score for computational complexity groups of features for Stack Overflow (SO), Stack Maths (SM), and Twitter.

Dataset	**Features**	**Qty**	**Time, s**	**F1 score Gain, %**			
				k-**NN**	**LR**	**DT**	**XGB**
SO	All raw	9	-	+1.1	+1.3	+0.8	+1.2
	All easy	27	~66,900	+8.9	+7.8	+8.3	+9.2
	All raw + easy	36	~66,900	+9.5	+8.6	+9.1	+11.6
	All hard	4	~313,500	+2.5	+3.2	+2.9	+1.7
	All raw + hard	13	~313,500	+3.1	+3.4	+4.0	+2.1
	All features	40	~380,400	**+9.5**	**+12.2**	**+14.7**	**+15.6**
SM	All raw	9	-	+0.6	+1.1	+1.2	-0.2
	All easy	27	~11,400	+7.7	+7.9	+8.2	+8.9
	All raw + easy	36	~11,400	+8.9	+8.3	+7.9	+10.2
	All hard	4	~44,800	+2.1	+2.2	+1.9	+2.9
	All raw + hard	13	~44,800	+3.3	+3.0	+2.0	+6.9
	All features	40	~56,200	**+10.0**	**+11.1**	**+12.8**	**+15.4**
Twitter	All raw	7	-	+0.2	+0.1	-0.7	+1.1
	All easy	15	~180	+1.5	+1.2	+1.3	+2.9
	All raw + easy	22	~180	+1.1	+1.5	+0.8	+3.1
	All hard	6	~21,400	**+17.4**	+16.5	+20.1	+34.1
	All raw + hard	13	~21,400	+15.4	**+18.0**	+18.2	+34.2
	All features	27	~21,580	+13.2	+11.0	**+21.2**	**+35.5**





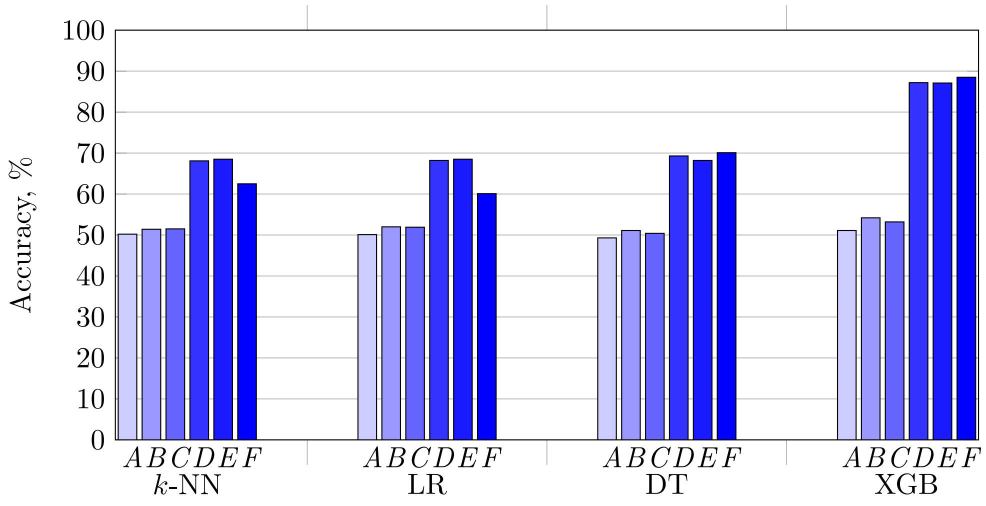


Figure 4.4: Accuracy for *Stack Overflow* (top), *Stack Maths* (middle), and *Twitter* (bottom); (A) - raw features, (B) - easy, (C) - raw and easy, (D) - hard, (E) - raw and hard, and (F) - all features.

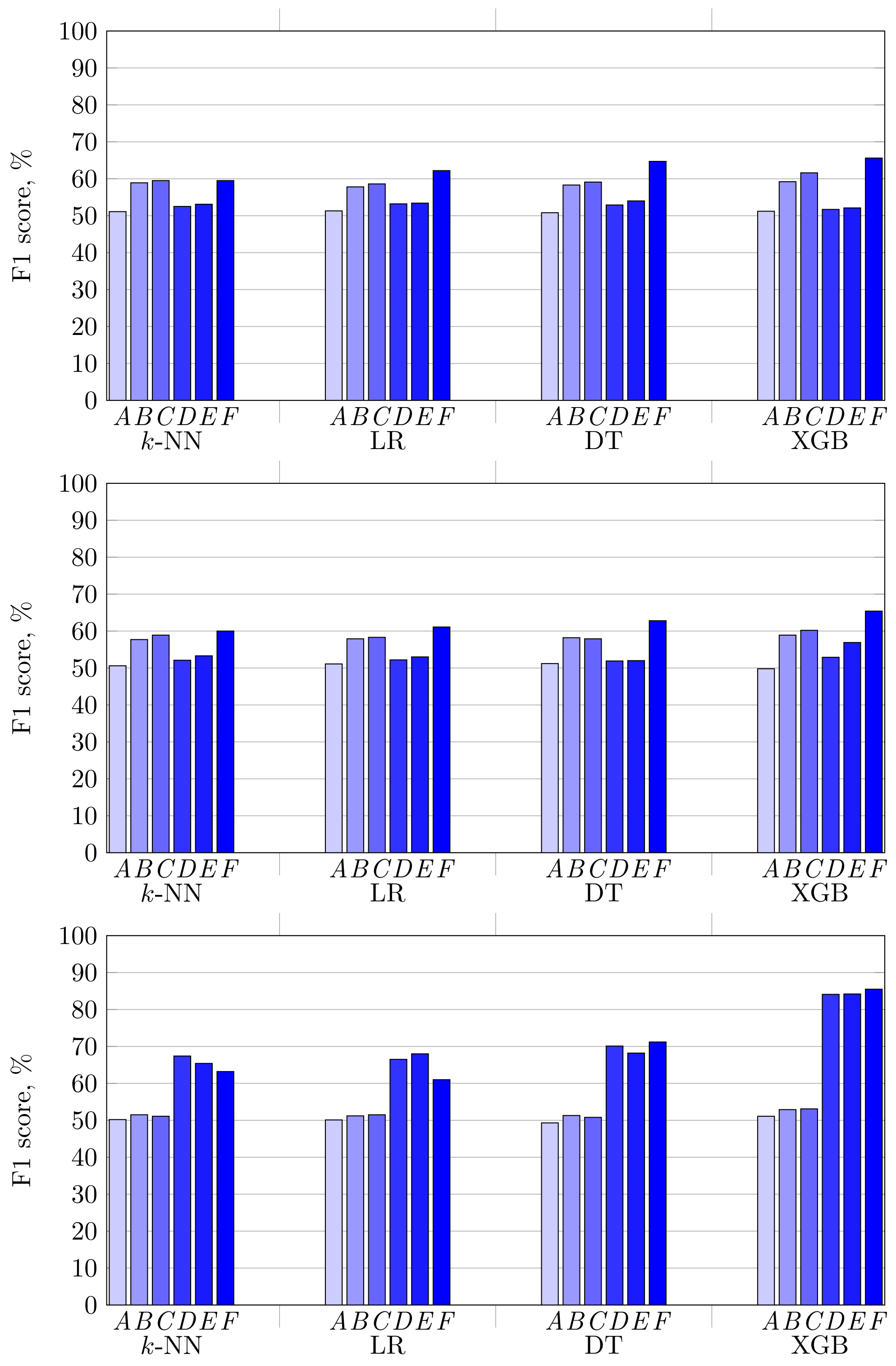
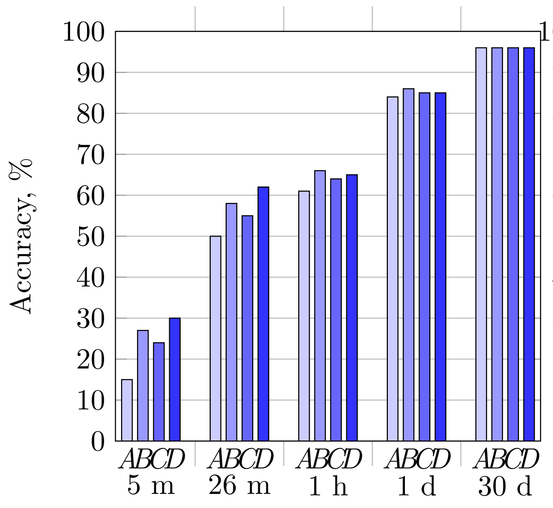
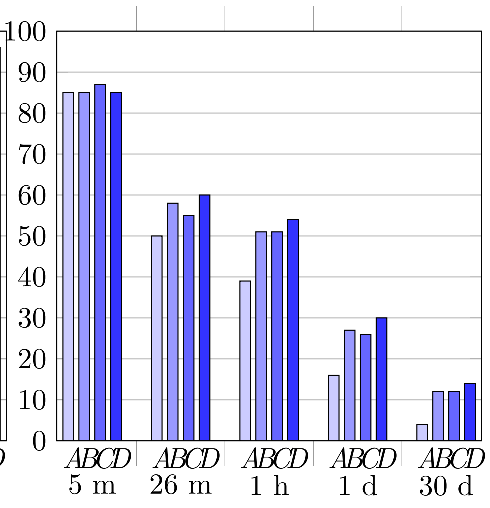


Figure 4.5: F1 score for *Stack Overflow* (top), *Stack Maths* (middle), and *Twitter* (bottom); (A) - raw features, (B) - easy, (C) - raw and easy, (D) - hard, (E) - raw and hard, and (F) - all features.



(a) Answered **faster** than the time T.



(b) Answered **slower** than the time T.

Figure 4.6: The results for the accuracy of prediction for *Stack Overflow* where the five different response times T were predicted using different ML algorithms; where (A) - Naive approach, (B) - DT, (C) - k-NN, and (D) - XGB.

for Stack Overflow and Stack Maths. User features for Twitter were raw that meant no calculations were involved. For the two other datasets, Stack Overflow and Stack Maths, calculating user features took the least time since the only features calculated were spatial features from a lookup table. Introducing spatial features such as users' location, latitude, longitude, and time zone showed importance for predicting users' response time. Indeed, users in different locations and time zones are less likely to be involved in a prompt discussion compared to users in the same location or time zone.

Second, it can be noticed that temporal features contribute the most to overall accuracy for both Stack Overflow and Stack Maths datasets whereas structural features contributed the most to the accuracy of Twitter prediction. Using all features together showed the best performance in terms of accuracy for both Stack Overflow and Stack Maths as well as for Twitter prediction in two experiments where DT and XGB ML algorithms were used. Nevertheless, for the other two experiments with a Twitter dataset with k-NN and LR algorithms, using only structural features showed the highest accuracy. Temporal features showed a 4.2% - 8.7% accuracy gain for the Twitter dataset but were three-four time less accurate compared to structural features. Nevertheless, structural features are very slow to calculate since the computational complexity of calculating these features is proportional to the number of nodes cubed (nodes represent the people in the network).

To summarise, the highest accuracy for ten out of twelve experiments, four for each dataset, were achieved by using the whole set of features. Thus, it might be possible to improve the accuracy of predictions by extending the list of the features further. However, for a fast prediction, the time complexity of these features must be taken into consideration since calculating these features can take several days which may not be acceptable for practical purposes.

4.6.2 Computational Variations of Features and Performance

Table 4.8 and Table 4.9 show the contributions of the groups of features to the resulted accuracy and F1 score of the predictions. One can notice that the time required to calculate the features is proportional to the complexity of the features. For the experiments in this thesis, calculating easy features required four - five time less time for Stack Overflow and Stack Maths datasets whereas for the Twitter dataset the time difference was more than two magnitudes.

For all three datasets, Stack Overflow, Stack Maths and Twitter datasets, raw features did not provide any significant accuracy or F1 score gain. In contrast, easy features provided a 6.6% - 8.9% accuracy gain for Stack Overflow and Stack Maths datasets. However, easy features were not useful for Twitter prediction. Finally, hard features slightly improved the accuracy for both Stack Overflow and Stack Maths and significantly improved the accuracy for Twitter prediction.

As a result, using only hard features for Twitter dataset would give the highest accuracy but calculating easy features would require a negligible effort compared to calculating hard features. Thus, it may be beneficial to use all features at least for DT and XGB algorithms. For Stack Overflow and Stack Maths datasets, using all features including hard features is preferential since using all features leads to the highest accuracy. Excluding hard features leads to a 2.7% - 7.4% decrease in accuracy.

4.6.3 Machine Learning Algorithms and Performance

For this experiment, four different ML algorithms were chosen, namely k-NN, LR, DT, and XGB algorithms. All the algorithms performed quite similar for all three datasets (see Figures 4.4 and 4.5). XGB showed the highest accuracy across all the three datasets, DT was the second, and k-NN with LR showed the worst accuracy. Also, all four algorithms performed quite consistently across various sets of features (see Tables 4.6 and 4.7). For example, all algorithms performed better with temporal and structural features. However, two exceptions were noticed where k-NN and LR performed worse using all features of Twitter dataset compared to using only structural features. The other two algorithms, DT and XGB, performed more consistently across various sets of features. On contrary, k-NN and DT showed consistent results for various set of features for Stack Overflow and Stack Maths datasets. One of the main reasons why k-NN and DT performed worse on all set of features rather than on structural features only, could be the fact that classes for prediction in Twitter were highly unbalanced. Thus, k-NN and DT seemed to struggle to learn under-represented class.

XGB showed higher accuracy with quite consistent results over a different set of features for all three datasets (see Figures 4.4 and 4.5). Indeed, this algorithm is using an ensemble of decision trees and showed high accuracy for a number of prediction tasks. Nevertheless, training models for all four algorithms should be taken into consideration. For example, k-NN is the slowest algorithm and XGB is quite slow compared to DT and LR. As a result, the question is whether it is possible to decrease the time for training and testing as well as to increase the accuracy of the prediction. The training time can be decreased by training models in mini-batches or in online modes and the accuracy can be improved by utilising an ensemble of models or using more complicated algorithms for training, for example, using deep learning algorithms.

Finally, Figure 4.6 demonstrates the results for varying balance between predicted classes for Stack Overflow dataset. The results showed that even varying the balance between classes allows one to build models which learn from the data since in all experiments the trained models outperformed a naive approach. For example, for the Stack Overflow website with unbalanced data the ML algorithms outperformed the naive approach by 5-20% for a smaller class.

4.6.4 A Guideline for Feature Extraction and Selection

The proposed classification of features into *semantic* and *computational* groups can be used for an efficient feature extraction and feature selection. The features F sorted in accordance with their complexity and their semantic group can be used for training predictive models in a few steps. The process of selecting the features for such a training involves choosing the least 'complex' semantic groups and then training a predictive model using these features.

The reason for selecting the features in groups rather than individually is due to the performance concerns. First, individual extensive selection is computationally very expensive. Second, there is a semantic relationship to features described in this Chapter.

Evaluating computational complexity groups of features is beneficial due to the fact that it takes less effort to extract simpler features than to extract more complex groups. However, there is a possibility that some of the included features in a group add little or no value to the resulting accuracy.

In the proposed classification, each feature is associated with both a semantic group and a complexity group. For example, a feature is an *easy user* feature. This classification is used to combine the features into groups to provide a fast feature extraction as well as to find a set of features which provides a fast and accurate model for prediction.

The classification into *light* and *heavy* as well as *easy* and *hard* features is arbitrary. The most important aspect of this classification is the fact that these features can potentially influence the accuracy of the prediction as well as it takes extra effort in terms of *space* (light or heavy) and *time* (easy or hard) to calculate the features. Also, the time and space complexity of those features depends on the data structures used. Thus, the same features can have different classifications for different implementations. However, the proposed classification can simplify finding a trade-off between the accuracy of the prediction and the time required to extract the features.

Finally, the proposed guideline of feature engineering involves the following steps:

1. Divide all *raw* features into *light* and *heavy* features;
2. Reduce the dimension for *heavy* raw features, for example, by creating a *'bag of words'* and then possible dimension reduction using Principal Component Analysis (PCA);
3. Calculate *easy* features (calculations are performed in constant time, computational complexity is $O(1)$);
4. Calculate *hard* features (computational complexity is harder than $O(1)$).

A predictive model can be trained on raw, easy, raw and easy, and then all features including the hard features. Also, a semantic grouping inside each of computational complexity group can be used. As a result, a set of features leading to the highest accuracy but with a reasonable time to extract and calculate these features can be derived.

4.7 Comparison to Related Studies

In [Gorodetsky et al., 2010] a feature space synthesis approach for "heavy" high dimensional learning tasks was proposed. It was reported that the important advantage of their proposed approach was the fact that the approach allows one to transform ontology-centred of heterogeneous possibly poor structured data into a homogeneous feature space. As a result, most ML learners can be easily trained on these features. On the contrary, the advantage of the proposed feature extraction in this thesis is the fact that the approach allows one to transform graph-based model of human communication into a homogeneous matrix-like feature space. Even more, the approach allows one to make this transformation efficiently by evaluating the time and space complexity of the transformation.

The purpose of many feature extraction methods is to remove redundant and irrelevant features so that classification of new instances will be more accurate [Hira and Gillies, 2015]. However, these approaches do not utilise the domain knowledge. The proposed approach for feature extraction is tailored for the domain of human communication where the features are associated with users, messages, and the relationships between people. Such association of the features with the model of human communication allows one to systematically extract features and build the feature space step by step in order to improve the accuracy of the predictive model but keep the feature space as simple as possible. The simplicity of the feature space is achieved by considering the cost of each feature which is associated to the time and space complexity of the transformation. This complexity characteristic of features can be used for removing redundant features or features that do not improve the performance of the feature space in training models.

The results correspond to the findings in [He et al., 2014] where the authors predicted ad clicks on Facebook. According to the results in their paper, in order to achieve an accurate and computationally efficient prediction of a user clicking on an ad, the most important is to choose the right features. In [He et al., 2014] these features included historical information about the user or ad and dominated other types of features. In this thesis, structural and temporal features showed high importance for predicting user behaviour which was shown for Stack Exchange and Twitter datasets. Also, it was shown that high importance features can be computationally involved.

The importance of temporal *tag* features was stated in [Bhat et al., 2014] which corresponds to the results for Stack Overflow and Stack Maths in this thesis. In this work, it was also shown that temporal features can not only significantly improve accuracy of prediction but these features are relatively efficient for calculation. Also, in [Bhat et al., 2014] the highest accuracy was achieved by using the whole set of features which matches the results in this thesis.

4.8 Conclusion

The results of the impact on accuracy and time performance for Stack Overflow, Stack Maths, and Twitter showed that temporal and structural features can significantly improve the accuracy of prediction. However, these features are computationally intensive features. Calculating such features requires computational efforts which can increase feature extraction time by more than 10-100 times.

As a result, a guideline for extracting features grouped into semantic and computational groups was proposed (see Subsection 4.6.4). This guideline suggests training models starting with the least complex computational features across various semantic groups and then extending the feature set with more complex features. The proposed approach for feature extraction allows one to extract features and then build prediction models efficiently. The guideline allows one to find a set of features which provides an accurate prediction but the time required for extracting the features is minimal.

4.9 Summary

In this Chapter, a feature extraction method for an efficient prediction was proposed. The approach involves the transformation of a TVAG-based model of human communication and relationship development into a homogeneous matrix-like feature space. The feature space is grouped both semantically and in accordance to the computational complexity of features into a few groups. Finally, it was shown how different semantic and computational groups of features influence the accuracy of user behaviour predictions across three different datasets. The results allowed to formulate a guideline for an efficient feature extraction.

In the next Chapter, the performance of two ML modes, online and offline (batch) modes, for predicting user behaviour on the Web is investigated. Also, a deep learning approach for predicting user behaviour is designed and then the performance of this approach is compared to state-of-the-art ML approaches.

Chapter 5

Choosing a ML Set-up for an Efficient Prediction

In this Chapter, the performance of a range of ML set-ups for building predictive models of user behaviour on the Web is explored. A method for comparing the performance of two conceptually different learning modes for training a model, online and offline learning modes is proposed and evaluated. The consistency of the results is verified for a few datasets, two datasets from *Stack Exchange*, namely *Stack Overflow* and *Stack Maths*, and a *Twitter* dataset. Finally, a Deep Learning (DL) algorithm based on a Deep Belief Network (DBN) with a feature selection approach is proposed and then the performance of this algorithm is compared to four other state-of-the-art ML algorithms. As a result, a guideline for choosing a ML set-up for an efficient prediction of user behaviour is proposed.

First, a brief introduction of choosing a ML set-up for a fast and accurate prediction of user behaviour is introduced in Section 5.1. Two learning modes, online and offline (batch) learning, are discussed in Section 5.2. The proposed method for comparing online and offline (batch) learning is introduced in Section 5.3. Section 5.4 presents a designed DBN-based DL set-up for predicting user behaviour on the Web. Section 5.5 describes computational complexity of a few popular online and offline (batch) learning algorithms. An experiment of comparing online and offline learning modes as well as the performance of the DBN-based DL algorithm to other state-of-the-art algorithms is presented in Section 5.6. The results of the experiments are discussed in Section 5.7. The results are compared to related studies in Section 5.8. Finally, Section 5.9 and Section 5.10 provide a conclusion and a summary of the Chapter.

5.1 Introduction

The era of the Internet and Big Data has provided us with access to a tremendous amount of digital data generated as a result of human activities on the web. This information allows one to apply statistical methods to the study of user behaviour. Due to the fast growth in the computational power of modern computers, as well as the advance in ML algorithms, many researchers have worked on developing ML approaches for predicting user behaviour on the Web [Burlutskiy et al., 2015; Nazerfard and Cook, 2013; Weerkamp and De Rijke, 2012; Choi et al., 2013]. As a result, there is a problem of choosing a ML set-up for a prediction.

There are two conceptually different modes of learning commonly used in ML, *online* and *offline/batch* learning. In most scenarios, *online* algorithms are computationally much faster and more space efficient [Liang et al., 2006]. Also, online learning scales very well for large amounts of data [Bifet et al., 2015]. Finally, *online* algorithms process data sample by sample which is natural for predicting user behaviour since the records of human activities are often in a chronological order. However, there is a question about how fast a prediction can be performed, and whether *online* learning can provide predictions which are as accurate as *offline* learning can provide. This question leads to the following research question [Burlutskiy et al., 2016]:

1. How do *online* and *offline* algorithms compare in terms of their time complexity and accuracy performance in predicting user behaviour on the Web?

There are two issues associated with comparing online and offline algorithms. First, due to the conceptual difference between online and offline learning, comparison of online and offline algorithms is not straightforward. In order to compare them, a method for comparing the accuracy and the time performance of training and testing the learners is proposed in this Chapter. Second, in many real world scenarios offline models are continuously retrained as more data is available and then these models are replaced with the updated ones. Another solution is to replace an old model with a new model trained only on the most recent data. Nevertheless, it is very hard to know in advance which solution is better for each particular case. This fact motivates conducting an experiment where the training batch size is varied and then prediction accuracy across three different datasets is evaluated.

In order to compare offline and online algorithms, five ML algorithms for offline training and three online ML algorithms are chosen and then their performance is compared using the proposed method for comparison of online and offline learning modes. The choice of ML algorithms for experiments in this Chapter is influenced by the papers where ML algorithms showed evidence of usefulness in predictions [Yang et al., 2011; Dror et al., 2013; Artzi et al., 2012; Anderson et al., 2012; Bhat et al., 2014; Lezina and Kuznetsov, 2012]. LR proves to be useful in solving predictive tasks due to scalability and transparency of results interpretation [Bhat et al., 2014]. Another algorithm, k-NN, is tested for comparison due to its simplicity. Another ML algorithm based on DT shows good performance for prediction in the context of Q&A communities predictions [Yang et al., 2011; Asaduzzaman et al., 2013; Bhat et al., 2014]. Deep Learning (DL) has shown high potential in producing accurate predictions [Hinton et al., 2006]. Thus, the second research question to answer is:

2. How does the performance of state-of-the-art ML algorithms compare? Is deep learning advantageous compared to other algorithms?

In order to answer this question, a DL approach, DBN, is chosen to train a predictive model of user behaviour on the Web. Then the performance of this model is compared to four other models trained using four other ML algorithms, namely LR, k-NN, DT, and SVM. The models are trained and evaluated across three different datasets, two from *Stack Exchange* and one dataset from *Twitter*.

First, an introduction on online and offline learning modes is discussed in the next Section.

5.2 Online and Offline (Batch) Learning

There are two conceptually different learning modes, namely *batch*, or *offline*, learning and *online* learning modes. The batch learning can also be divided in full-batch learning where weights of a model are updated over the entire training data and mini-batch learning when the weights are updated after some number m of training instances. On the contrary, in the online learning mode, the weights are updated using one training instance at a time. Both batch and online learning modes solve a learning problem defined by an input set X and a label set Y. The goal of these modes to train a model for predicting labels from Y for the instances in X.

5.2.1 Offline (Batch) Learning

In batch learning, there is an assumption on a probability distribution over the product space $X \times Y$, where X is an input set and Y is a label set. A batch learning algorithm uses a training set D_{tr} to generate an output hypothesis, which is a function F that maps instances in X to class labels in Y. The batch learning algorithm is expected to generalise, in the sense that its output hypothesis should accurately predict the labels of previously unseen examples, which are sampled from the distribution [Dekel, 2009].

Examples of ML algorithms for prediction of user behaviour which can be executed in *batch* mode are *Decision Trees* (DT), *Support Vector Machines* (SVM), *Logistic Regression* (LR), *k Nearest Neighbors* (*k*-NN), and *Deep Belief Network* (DBN) [Zheng et al., 2013; Nazerfard and Cook, 2013; Choi et al., 2013; Burlutskiy et al., 2015].

5.2.2 Online Learning

Online prediction is based on a training algorithm when a learner operates on a sequence of data entries. At each step t, the learner receives an example $x_t \in X$ in a d-dimensional feature space, that is, $X = R^d$. The learner predicts the class label for each example as soon as it receives it:

$$\hat{y}_t = sgn(f(x_t, w_t)) \in Y \tag{5.1}$$

Where $\hat{y}_t$ is the predicted class label, x_t is an example, w_t is the weight assigned to the example, $sgn()$ is the sign function, and $Y = \{0, 1\}$ for a binary classification task. Then the true label $y_t \in Y$ is revealed which allows one to calculate the loss $l(y_t, \hat{y}_t)$ which reflects the difference between the learner's prediction and the revealed true label y_t. The loss is used for updating the classification model at the end of each learning step.

A generalised online learning algorithm is shown in Algorithm 1 [Hoi et al., 2014]. This algorithm can be instantiated by substituting the prediction function f, loss function l, and update function Δ.

Stochastic Gradient Descent (SGD), *Perceptron*, and *Passive-Aggressive* (PA) are three instances of the algorithm which are of the interest in this thesis.

Algorithm 1 Online Learning

```
1: Initialise: $w_1 = 0$
2: for t=1,2,..,T do
3:     The learner receives an incoming instance: $x_t \in X$;
4:     The learner predicts the class label: $\hat{y}_t = sgn(f(x_t, w_t))$;
5:     The true class label is revealed from the environment: $y_t \in Y$;
6:     The learner calculates the suffered loss: $l(w_t, (x_t, y_t))$;
7:     if $l(w_t, (x_t, y_t)) > 0$ then
8:         The learner updates the classification model: $w_{t+1} \leftarrow w_t + \Delta(w_t, (x_t, y_t))$;
```

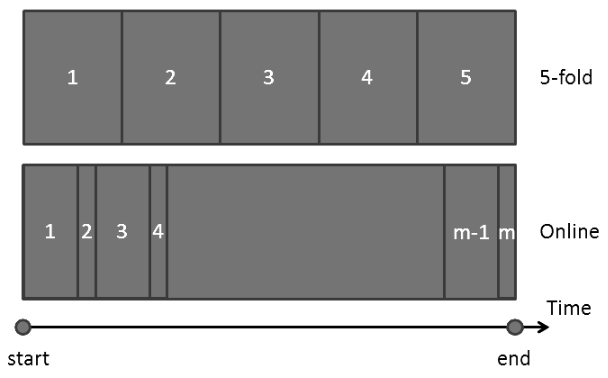


Figure 5.1: Proposed training and testing scheme for comparison of online and offline learning algorithms.

- *Stochastic Gradient Descent*: the function f in Algorithm 1 for this learner is the dot product: $f(x_t, w_t) = w_t \cdot x_t$ and the function Δ is calculated as follows: $\Delta = w_t + \eta(y_t - \hat{y}_t)x_t$ where η is the learning rate. For the loss function, we used hinge loss: $l(y_t) = \max(0, 1 - y_t(w_t \cdot x_t))$.
- *Perceptron*: the difference to SGD is in the way the learner's loss is calculated: $l(y_t) = \max(0, -y_t(w_t \cdot x_t))$;
- *Passive-Aggressive*: the family of these algorithms includes a regularization parameter C. The parameter C is a positive parameter which controls the influence of the slack term on the objective function. The update function is: $\Delta = w_t + \tau y_t x_t$ where $\tau = min(C, \frac{l_t}{||x_t||^2})$ and the loss function l is hinge loss. Larger values of C imply a more aggressive update step [Crammer et al., 2006].

In the next Section, the proposed method for comparing online and offline learning modes is presented.

5.3 Proposed Method to Compare Online and Offline Algorithms

Due to the conceptual difference in online and offline learning, a method for comparing the accuracy and the time performance of training and testing the learners is proposed. The standard performance measurements for offline learning are accuracy, precision, recall, and F1-measure [Mohri et al., 2012].

The performance of online learners is usually measured by the cumulative loss a learner suffers while observing a sequence of training samples. In order to compare online and offline learning algorithms, a method which is a modification of mini-batch training is introduced (see Figure 5.1).

First, Algorithm 2 is applied to online learners l_{on} and then the same al-

Algorithm 2 Calculating the accuracy of learners

```
 1: Initialise training and testing times: $T_{tr} = 0, T_{tt} = 0$
 2: Sort the data $D$ chronologically from the oldest to the latest;
 3: Divide the data $D$ into $m$ equal batches $D_i$
 4: for i=1,2,..,m do
 5:     Divide each batch $D_i$ into a training set $D_{itr}$ and test set $D_{itt}$;
 6:     Train a learner $l_i$ on $D_{itr}$;
 7:     Record the time taken for the training $T_{itr}$;
 8:     Update the training time $T_{tr} = T_{tr} + T_{itr}$;
 9:     Test $l_i$ on $D_{itr}$;
10:     Record the time taken for the testing $T_{itt}$;
11:     Update the testing time $T_{tt} = T_{tt} + T_{itt}$;
12:     Calculate the average accuracy $A_i$ over $D_{itr}$;
13: Calculate the average accuracy $A_{av}$ over all $A_i$
```

gorithm is repeated for offline learners l_{off}. The purpose of applying the same aforementioned algorithm to both online and offline learners is to compare the accuracy A_{av} and the performance of online and offline algorithms in terms of the training time T_{tr} and testing time T_{tt} on the same data D in the same set-up.

Second, a classical 5-fold validation for training and testing the models is used. The whole set is divided into five parts (Figure 5.1), a model is trained on any four parts and then it is tested on the fifth part. Then the parts for training and testing are changed and the process is repeated. Eventually all the five possible combinations of the parts for training and testing are used. Then the average accuracy A_{av} and the time performance for training T_{tr} and testing T_{tt} for for these five tests is calculated. This process is repeated for both online and offline algorithms.

A common way to convert online to batch learning is by presenting training examples one-by-one to the online algorithm, and then use the resulted model for testing. This technique is called the last-hypothesis online-to-batch conversion technique in [Dekel, 2009]. This technique is, first, for converting online to batch learning but not for comparison of the performance of online and offline learning modes. Second, the proposed method for comparison of online and offline learning methods in this thesis defines how to measure accuracy and time performance of these two conceptually different learning modes (see Algorithm 2).

The next Section introduces a DL approach for predicting user behaviour.

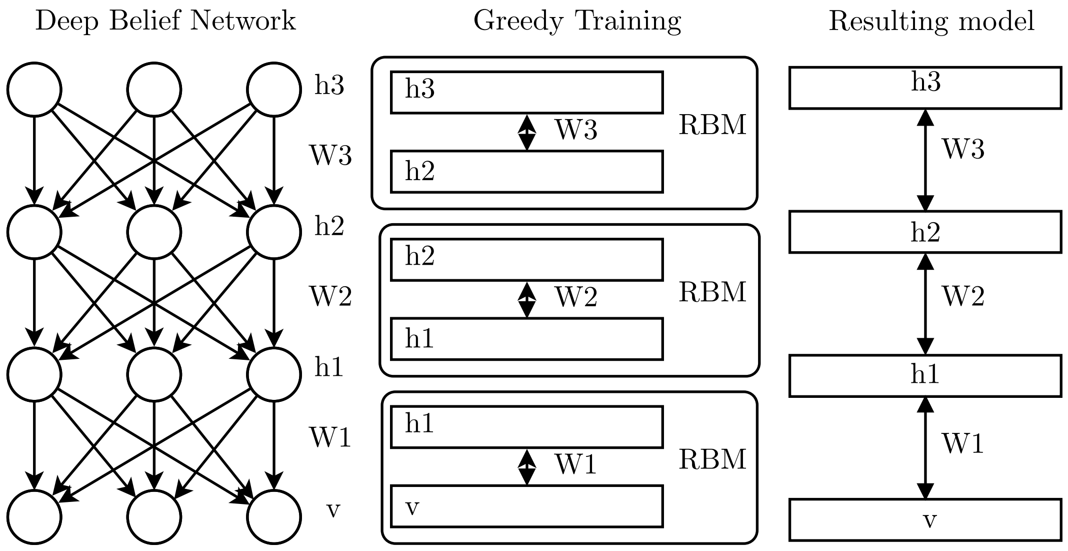


Figure 5.2: An example of the structure (left), training procedure (in the middle), and the resulting model after training (right) of a Deep Belief Network with three hidden layers.

5.4 Deep Learning Approach

A number of ML algorithms have been proposed, but there is no learning algorithm dominating the others. Nevertheless, several algorithms are widely used due to their ease of use in practice for large datasets and theoretical strength [Kotsiantis, 2007]. In this thesis, using both online and offline training modes, as well as different ML algorithms are proposed for prediction of user behaviour (see Section 2.6).

However, in order to achieve better accuracy, a DL approach, namely DBN-based approach, is employed for training a model and then using this model for predicting user behaviour on the Web.

5.4.1 Deep Belief Network (DBN)

A DBN is a type of neural network with multiple layers of hidden units. The layers are interconnected but there is no connections between units in a layer. Typically, these building units are Restricted Boltzmann Machines (RBMs). Since an RBM is non-linear, composing several RBMs together allows one to represent highly non-linear patterns in training data. Thus, the strength of DBN is in the fact that each layer can efficiently represent non-linearities in training data [Hinton et al., 2006]. In Figure 5.2 a graphical depiction of a DBN with three hidden and one visible layers is shown.

A RBM is a network of symmetrically coupled stochastic binary units. It contains a set of visible units $\mathbf{v} \in \{0,1\}^D$ and a set of hidden units $\mathbf{h} \in \{0,1\}^P$. In an RBM, on contrary to Boltzmann Machines (BMs), there are connections only between visible and hidden units and there are no connections within visible or hidden units. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{5.2}$$

Where v_i and h_j are the binary states of visible unit i and hidden unit j, a_i and b_j are their biases, and w_{ij} is the weight between these two states.

The probability of the visible vector $\mathbf{v}$ can be calculated as:

$$p(\mathbf{v}) = \frac{\sum_h \exp\left(-E(\mathbf{v}, \mathbf{h})\right)}{Z} \tag{5.3}$$

Where the normalising factor Z is called a partition function and can be determined as:

$$Z = \sum_v \sum_h \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{5.4}$$

The derivative of the log probability of a training vector with respect to a weight is as follows:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = E_{data}[v_i h_j] - E_{model}[v_i h_j] \tag{5.5}$$

Where $E_{data}[.]$ is an expectation with respect to the distribution defined by the data and $E_{model}[.]$ is an expectation with respect to the distribution defined by the model. Finally, the learning rule for stochastic steepest descent in the log probability of the training data becomes as follows:

$$\Delta w_{ij} = \alpha(E_{data}[v_i h_j] - E_{model}[v_i h_j]) \tag{5.6}$$

Where α is a learning rate.

The conditional distributions for hidden layers over the visible layer can be calculated as follows:

$$\begin{aligned} p(h_i = 1 | \mathbf{v}, h_i) &= \sigma(b_i + \sum_{j=1}^{D} w_{ij} v_j) \\ p(v_j = 1 | \mathbf{h}, v_j) &= \sigma(a_j + \sum_{i=1}^{P} w_{ij} h_i) \end{aligned} \tag{5.7}$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ is a logistic function.

5.4.2 Training Procedure

Training a DBN is a time consuming process. However, a greedy learning algorithm was proposed in [Hinton et al., 2006]. In this algorithm, each layer of a DBN is trained in a sequence. Starting from the bottom visible layer, each layer performs a non-linear transformation on its input vectors and produces output vectors. Then these output vectors are used as input vectors for the next layer in the sequence. Finally, the last top layer produces the prediction label y.

Each layer is trained for n_e sweeps (or so-called 'epochs') through the training set. During this training, the units in the 'visible' layer of each RBM has real-valued activities between 0 and 1. For training higher layers of weights, the real-valued activities of the visible units in the RBM are the activation probabilities of the hidden units in the RBMs of the lower layer. In the training of RBMs of the hidden layer, stochastic binary values are used.

Practically, the data is divided into m balanced mini-batches each containing k examples of each class. The weights of the model are updated after each mini-batch. Finally, the training procedure for a DBN in a mini-batch set-up is described in Algorithm 3.

Algorithm 3 Training a DBN

```
1: Sort the data $D$ chronologically from the oldest to the latest;
2: Divide the data $D$ into $m$ equal batches $D_i$;
3: for i=1,2,..,m do
4:     Divide each batch $D_i$ into a training set $D_{itr}$ and test set $D_{itt}$;
5:     Train a learner $l_i$ on a training set $D_{itr}$ starting with the bottom layer $n$:
6:     for j=1,2,..,n do
7:         Train a RBM on a training set $D_{itr}$ to obtain its weight matrix, $W_j$;
8:         Use $W_j$ as the weight matrix between the network layers $n$ and $n-1$;
9:         Transform $D_{itr}$ by the RBM to produce new data $D'_{itr}$ by computing the
10:         mean activation of the hidden units;
11:     Fine-tune all the parameters of $l_i$;
12:     Test $l_i$ on $D_{itt}$;
13:     Calculate the accuracy $A_i$ of the learner $l_i$ for $D_{itt}$;
```

5.5 Complexity of ML Algorithms

The complexity of ML algorithms for both training and testing times varies significantly across different families of algorithms and depends on the number of samples, number of features, and algorithm parameters. For some algorithms, a formal

Table 5.1: Time complexity of the algorithms.

Algorithm	**Implementation**	**Complexity**	**Reference**
DT	C4.5	$O(mn^2)$	[Su and Zhang, 2006]
SVM	LibSVM	$O(n^3)$	[Chapelle, 2007]
k-NN	kd-tree	$O(k \log(n))$	-
LR	LM BFGS	$O(mn)$	[Minka, 2003]
DBN	m hidden layers	high*	[Bianchini and Scarselli, 2014]
SGD, P, PA	Algorithm 1	$O(kn\bar{p})$	[Bottou, 2010]

* The complexity is linearly proportional to the number of layers.

time complexity analysis has been reported in the literature [Su and Zhang, 2006; Chapelle, 2007; Bianchini and Scarselli, 2014; Minka, 2003]. This analysis can help to choose an algorithm for a prediction. For example, if time performance is crucial then it might be advantageous to choose an algorithm with lesser time complexity. In contrast, in cases when accuracy is more important than time performance, then a more time-complex algorithm can be executed, for example, SVM or ANN. Nevertheless, the time complexity of ML algorithms depends on the implementation of the algorithms.

Time Complexity

In this Chapter, five offline algorithms were considered, namely *Decision Trees* (DT), *Logistic Regression* (LR), *Support Vector Machines* (SVM), *k Nearest Neighbours* (k-NN), and *Deep Belief Networks* (DBN), and three online algorithms, namely *Stochastic Gradient Descent* (SGD), *Perceptron*, and *Passive-Aggressive* (PA). The time complexity of training these models is presented in Table 5.1.

For DT-based learner C4.5, the time complexity is $O(mn^2)$ where m is the number of features and n is the number of samples [Su and Zhang, 2006]. The time complexity of LR is $O(mn)$ but might be worse depends on the implementation of the optimization method [Minka, 2003]. For SVM, the time complexity, again, depends on optimisation method but for one of the most popular implementation in LibSVM the time complexity is $O(n^3)$ [Chapelle, 2007]. For k-NN implemented as a kd-tree, training cost is $O(k \log(n))$ where k is the number of nearest neighbours and n is the number of instances. For more complicated models, such as DBN, a formal complexity analysis is hard since there is no measure to evaluate the complexity of functions implemented by deep networks [Bianchini and Scarselli, 2014].

The training cost for online learners used in this thesis is $O(kn\bar{p})$, where

n is the number of training samples, k is the number of iterations (epochs), $\bar{p}$ is the average number of non-zero attributes per sample [Bottou, 2010]. Although kernel-based online learners potentially can achieve better accuracy, they require more memory to store the data. Moreover, these learners are computationally more expensive what makes them unsuitable for large-scale applications. Thus, only linear online learners are considered in this thesis.

5.6 Experiment Outline

The experiments are conducted for three datasets, *Stack Overflow* dataset, *Stack Maths* dataset, and *Twitter* dataset. The extracted features are presented in Tables B.1, B.2, B.3, and B.4 for Stack Overflow and Stack Maths datasets, and in Tables B.5, B.6, B.7, B.8, and B.9 for Twitter dataset. As a result, the features are aggregated in a feature matrix of size $n \times m$, where n is the number of features F, and m is the number of instances Q. This matrix is used for training and then testing a classification model built by a ML algorithm.

In order to avoid features in greater numeric values to dominate others, the features are normalised to an interval $[0, 1]$. Also, the features are standardised by removing the mean value $\mu = 0$ and scaling variance to one $\sigma = 1$. This step is critical for regularised classification models such as k-NN and SVM. Also, this data transformation can improve the training time for SGD classifiers. Finally, categorical features are converted into numeric features.

The prediction tasks are formulated as binary classification tasks and then executed across three different datasets, the two largest datasets from Stack Exchange websites and a dataset from Twitter platform:

1. *Stack Exchange*: Predicting users' response time on the Web, particularly in the context of the largest Q&A community, Stack Exchange;
2. *Twitter*: Predicting whether a tweet will be retweeted within 24 hours or not.

Finally, two experiments are conducted. In the first experiment, three online (SGD, Perceptron, PA) and five offline (LR, k-NN, DT, SVM, DBN) algorithms are trained on the same dataset with all features. In the second experiment, the performance of the designed DL algorithm (see Section 5.4), DBN, is compared to the state-of-the-art algorithms in both batch and mini batch settings.

Offline vs Online Algorithms In the first experiment, three online (SGD, Perceptron, PA) and five offline (LR, k-NN, DT, SVM, DBN) algorithms are trained on the same dataset with all features. The goal of this experiment is to compare online and offline learning using the proposed method described in Section 5.3.

Online algorithms: The number of iterations is chosen to be equal to one since it was enough for the convergence of SGD. No regularisation methods are used [1]. The learning rate η is set to be equal '1'.

Offline algorithms: For k-NN classifier, k=3 is used. For DT, C4.5 algorithm is used. For SVM, LibSVM is used.

Deep Learning Approach In the second experiment, the performance of the designed DL algorithm (see Section 5.4), a DBN-based algorithm with a feature selection approach, is compared to the state-of-the-art ML algorithms.

Parameters of DBN: The number of epochs in the first greedy training step is set to 10 for all the hidden layers and the output layer. The learning rate is set to 0.1 with the decay rate set to 0.9 for each epoch. The number of neuron units in the output layer is two since all prediction tasks are defined as binary classification tasks. The number of input neuron units equals the number of features used for prediction. Finally, the number of neural units for the two hidden layers was set to 300 and 100. The choice of two hidden layers was dictated by the fact that more hidden layers did not improve the accuracy but worsen the time performance of the models. This choice of the architecture was dictated by the architectures showed high accuracy in the literature [Ruangkanokmas et al., 2016].

Each algorithm is executed ten times and then the average of these independent runs is recorded.

Finding the Batch Size In order to find out how the size of a batch influences the accuracy of a prediction, the size of a batch is varied. After k instances X_i are received, the features F_i are calculated based on these instances and then a model M_i is built (see Figure 5.3). Then this model M_i is employed for predicting using a test dataset. As a result, by varying the batch size k, it is possible to find such k_b which is small enough but provides with relatively high accuracy. Finally this k_b can be used for training a DBN in a mini-batch mode.

After the batch size k_b is determined, the experiment in the proposed mini-batch setting (see Algorithm 2 and Algorithm 3) with such k_b samples is run an then the same experiment is repeated using 5-fold validation.

[1]In the experiments, L_1 and L_2 regularisation were tried but no significant improvements in the results were achieved compared to the results without regularisation reported in this paper.

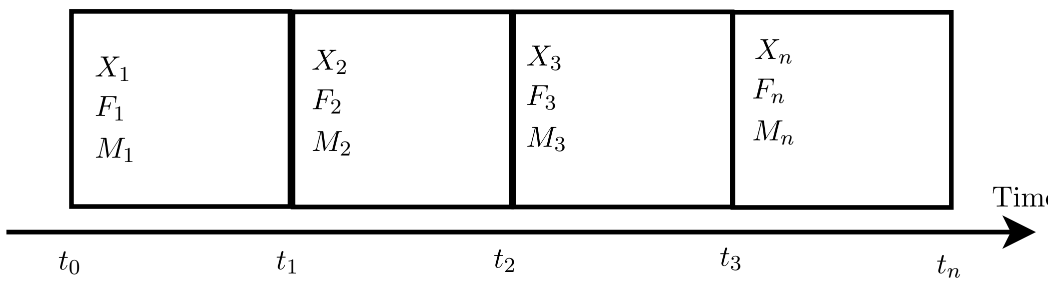


Figure 5.3: The proposed batch training of a model.

5.7 Results and Discussions

The results for the first and the second experiment, comparing offline and online learning modes across the three datasets, are presented in Table 5.2 and Table 5.3. The results for the change in the accuracy of predictions with respect to the number of training instances are in Figure 5.4, Figure 5.5, and Figure 5.6 for Stack Overflow, Stack Maths, and Twitter correspondingly.

5.7.1 Online vs Offline

There was no significant difference in the accuracy of the offline (batch) learning algorithms compared to online learning algorithms (see Table 5.2 for mini-batch training and the proposed method for evaluation and Table 5.3 for the 5-fold validation evaluation). However, the time spent on training and testing the batch models was significantly longer especially for DBN and SVM models compared to online learning methods such as PA, Perceptron, and SGD. The fastest batch learning algorithm, LR, showed almost $1.8 - 5.2\%$ worse accuracy than SGD and it was $7 - 13$ times slower in terms of the training time across all datasets. Nevertheless, LR performed slightly worse than SGD in one out of six experiments (see Table 5.3). SGD showed the best accuracy of 63.1% over the other online learners (see Table 5.2). Also, DBN slightly outperformed (by 3% – 9%) other ML algorithms in all set-ups.

SGD also outperformed other online algorithms. Relatively poor performance showed Perceptron (up to 6.1% worse than SGD), probably due to the fact that Perceptron is sensitive to badly labelled examples even after training on many instances, whereas the SGD and PA algorithms are more robust to badly labelled examples. Nevertheless, Perceptron slightly outperformed PA in two out of six experiments. Also, the fact that the online algorithms used different importance on each training instance over time might have influenced the results.

The time performance of batch algorithms corresponds to their time complexity evaluated in Section 5.5 earlier. SVM and DBN performed the worst and relatively similar in terms of the training time. The results are consistent across the three datasets used in the experiments. However, the time performance of DBN and SVM highly deteriorates with the number of training instances. Thus, training these algorithms require careful choice of the batch size.

It was expected that ML algorithms executed in the online learning mode should perform faster than the same algorithms executed in the batch learning modes in both the training and the prediction steps (see Table 5.2 for mini-batch training and the proposed method for evaluation and Table 5.3 for the 5-fold validation evaluation). The reason is in the nature of these algorithms - in online learning the weights of the model are updated at each step compared to the computationally expensive training in the batch mode. However, such significant differences were not expected. Even though the experimental results are hardware dependent and it is challenging to have a formal big O evaluation of the complexity of most of these algorithms, training and testing the models in the online learning mode took $13 - 3000$ times less time.

From another side, it can be seen that there is almost no difference in training and predicting time for all the online learning algorithms. Nevertheless, the training time for Perceptron was slightly longer than the training time of the other algorithms used for the datasets.

All learners showed worse accuracy in the results for 5-fold validation (see Table 5.3) for Stack Exchange datasets but better accuracy for Twitter dataset. One possible cause of such discrepancy in the accuracy can be the nature of user behaviour in these datasets, Stack Exchange dataset cover several years of user activity whereas Twitter dataset covers only four days of tweeting activity. This fact shows that the choice of a method for comparison of online and offline learning modes can significantly influence the results. Nevertheless, the proposed method for comparison seem to be more suitable due to the sequential nature of temporal user activity (Section 5.3). However, relative performance of ML algorithms using both the proposed method and 5-fold validation is similar where DBN performed the best, k-NN the worst, and DT, LR, and SVM quite similar to each other. Also, for online learners, SGD slightly outperformed the other online classifiers.

5.7.2 Deep Learning vs State-of-the-Art Algorithms

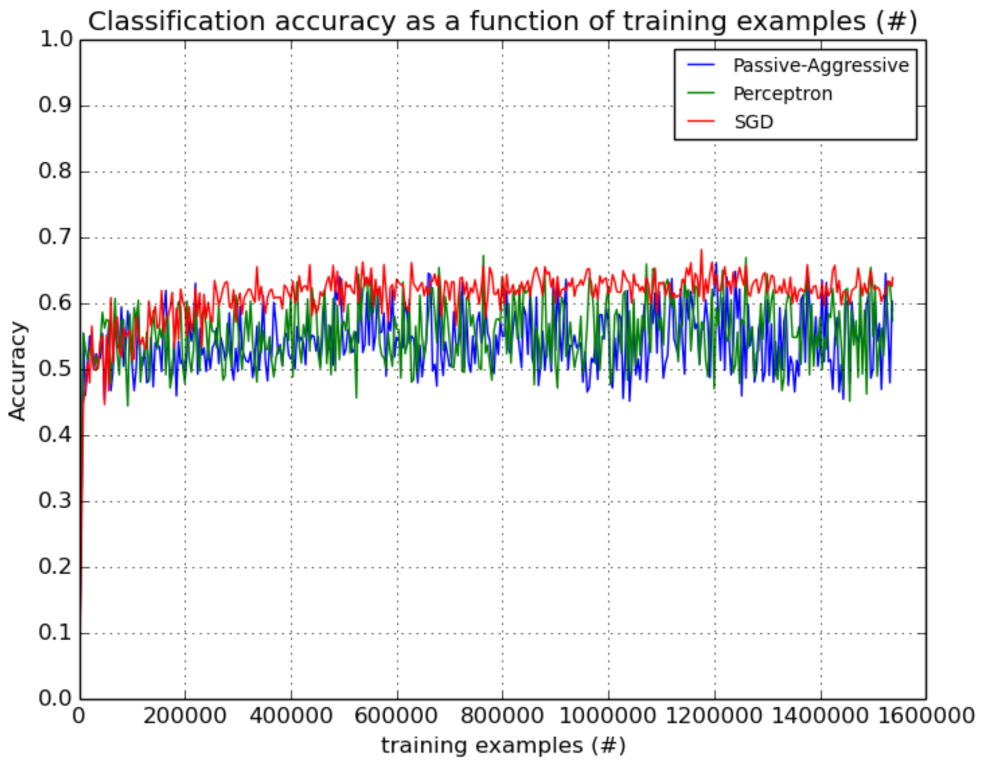
DBN slightly outperformed other ML algorithms in most set-ups. The worst accuracy was shown by k-NN whereas LR and DT showed almost no difference in

Table 5.2: The Results for offline and online learning in comparison (each batch of 5,000 samples).

Dataset	**Algorithm**	**Accuracy, %**	**Training, s**	**Testing, s**
SO	*k*-NN	57.8 ± 0.73	98.1	940.1
	DT	60.1 ± 0.43	60.8	0.985
	LR	61.4 ± 0.66	**18.6**	**0.362**
	SVM	62.0 ± 0.51	2,320.1	350.5
	DBN	**66.5 ± 0.51**	3,034.0	720.2
	PA	62.8 ± 0.62	1.3	**0.002**
	Perceptron	57.9 ± 0.55	**0.7**	0.003
	SGD	**63.7 ± 0.80**	1.2	**0.002**
SM	*k*-NN	60.0 ± 0.62	5.1	8.722
	DT	62.8 ± 0.52	5.3	0.055
	LR	61.1 ± 0.52	**2.0**	**0.044**
	SVM	62.1 ± 0.47	210.5	46.80
	DBN	**64.2 ± 0.58**	380.1	92.51
	PA	61.4 ± 0.46	0.2	**0.001**
	Perceptron	62.1 ± 0.55	**0.3**	**0.001**
	SGD	**63.1 ± 0.65**	**0.3**	**0.001**
Twitter	*k*-NN	57.6 ± 0.45	22.4	74.55
	DT	58.2 ± 0.63	40.9	0.829
	LR	57.7 ± 0.57	**12.1**	**0.412**
	SVM	58.4 ± 0.42	1,254.3	210.0
	DBN	**63.4 ± 0.52**	2,021.2	428.4
	PA	59.7 ± 0.49	1.0	**0.002**
	Perceptron	59.4 ± 0.54	**0.9**	0.003
	SGD	**62.9 ± 0.62**	1.1	0.003

Table 5.3: The Results for batch learning with 5-fold validation and online learning in comparison.

Dataset	**Algorithm**	**Accuracy, %**	**Training, s**	**Testing, s**
SO	*k*-NN	60.5 ± 0.70	811.5	2799
	DT	65.2 ± 0.20	190.1	0.499
	LR	64.2 ± 0.35	**82.5**	**0.065**
	SVM	62.4 ± 0.46	2,900.1	382.0
	DBN	**67.2 ± 0.82**	2,715.2	711.2
	PA	59.8 ± 0.61	4.5	0.068
	Perceptron	55.7 ± 0.52	**4.1**	**0.050**
	SGD	**61.2 ± 0.42**	4.5	0.053
SM	*k*-NN	61.2 ± 0.52	6.0	7.932
	DT	63.1 ± 0.46	7.0	0.051
	LR	60.6 ± 0.73	**4.4**	**0.042**
	SVM	62.0 ± 0.50	322.2	45.68
	DBN	**64.3 ± 0.53**	560.1	91.72
	PA	62.1 ± 0.53	0.4	0.002
	Perceptron	61.7 ± 0.57	0.5	0.002
	SGD	**63.1 ± 0.60**	**0.3**	**0.001**
Twitter	*k*-NN	62.5 ± 0.52	56.4	85.43
	DT	70.1 ± 0.56	150.9	0.932
	LR	60.1 ± 0.55	**21.2**	**0.515**
	SVM	59.2 ± 0.52	1,104.3	308.0
	DBN	**71.4 ± 0.62**	1,987.2	368.4
	PA	59.9 ± 0.52	1.4	0.004
	Perceptron	60.2 ± 0.51	**1.3**	**0.003**
	SGD	**62.6 ± 0.51**	1.5	**0.003**



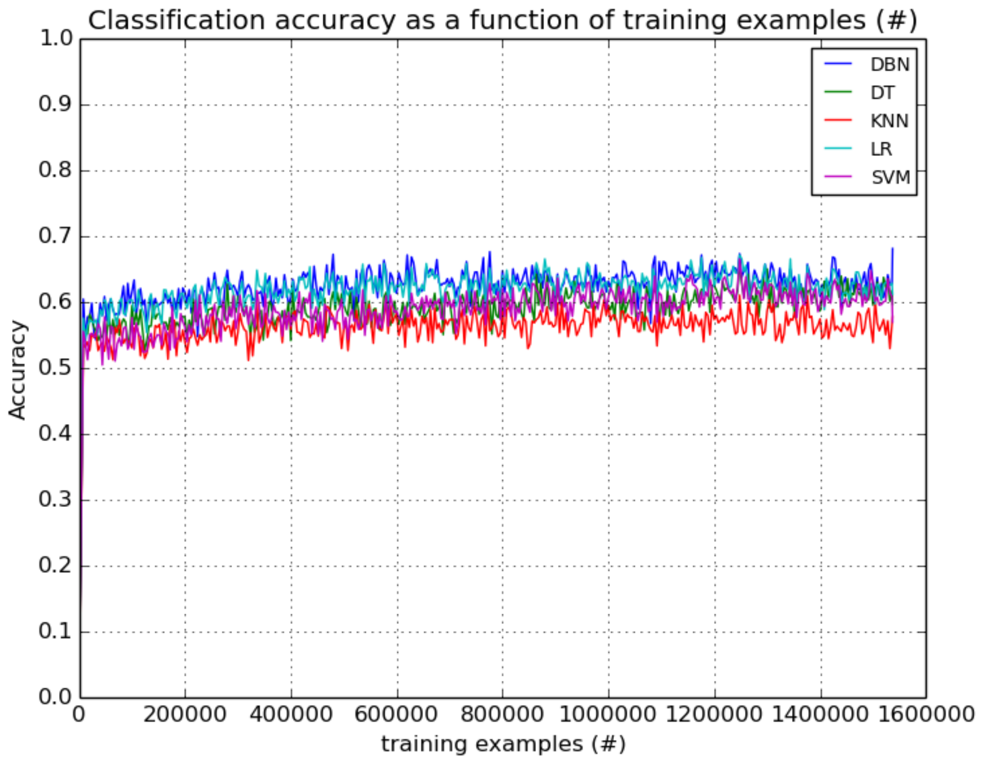
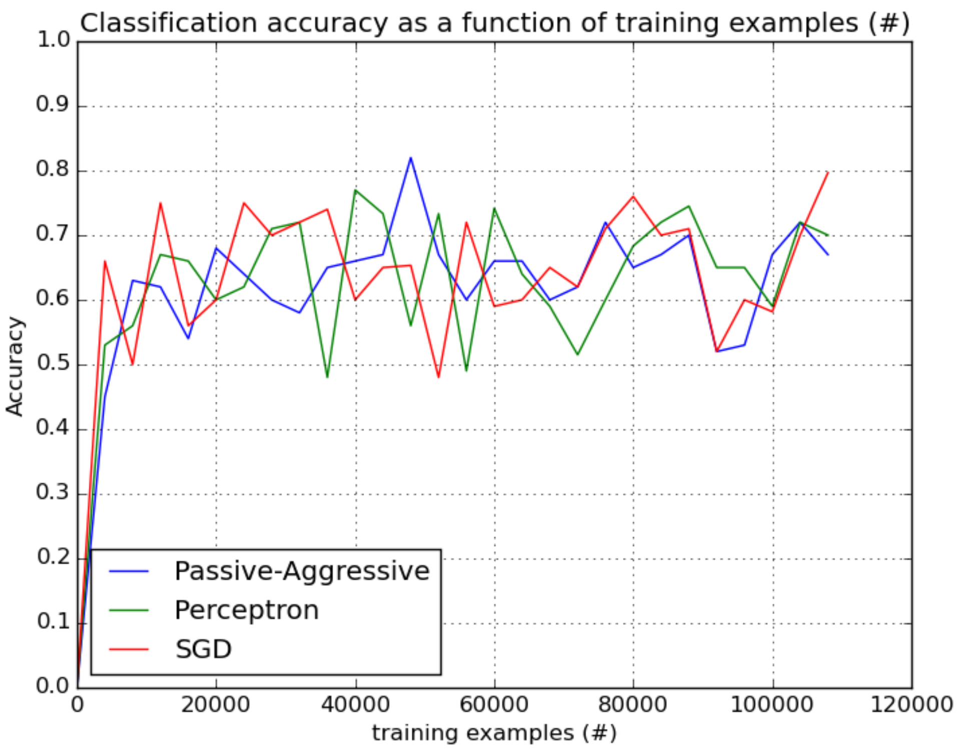


Figure 5.4: Results for online (top) and offline (bottom) learning for Stack Overflow; mini-batches of 5,000 samples each.



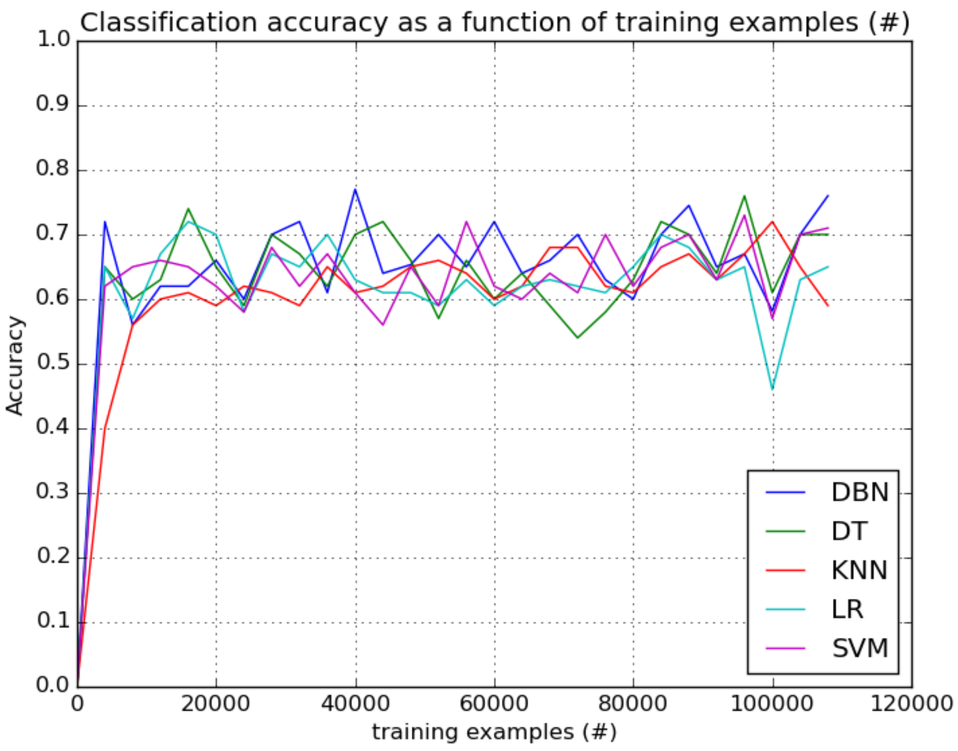
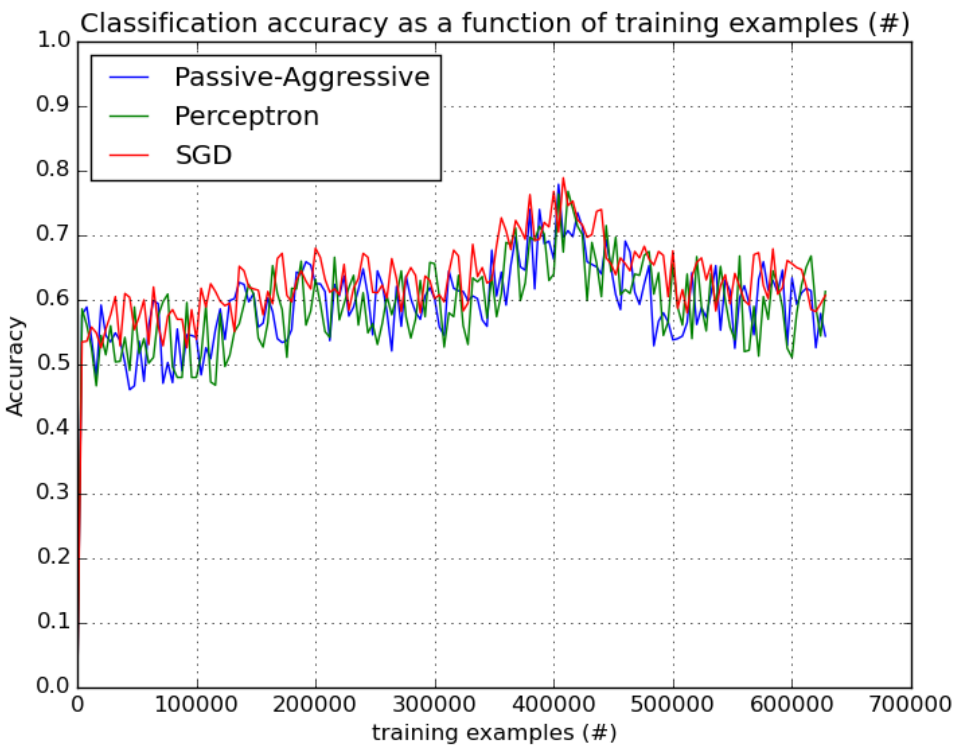


Figure 5.5: Results for online (top) and offline (bottom) learning for Stack Maths; mini-batches of 5,000 samples each.



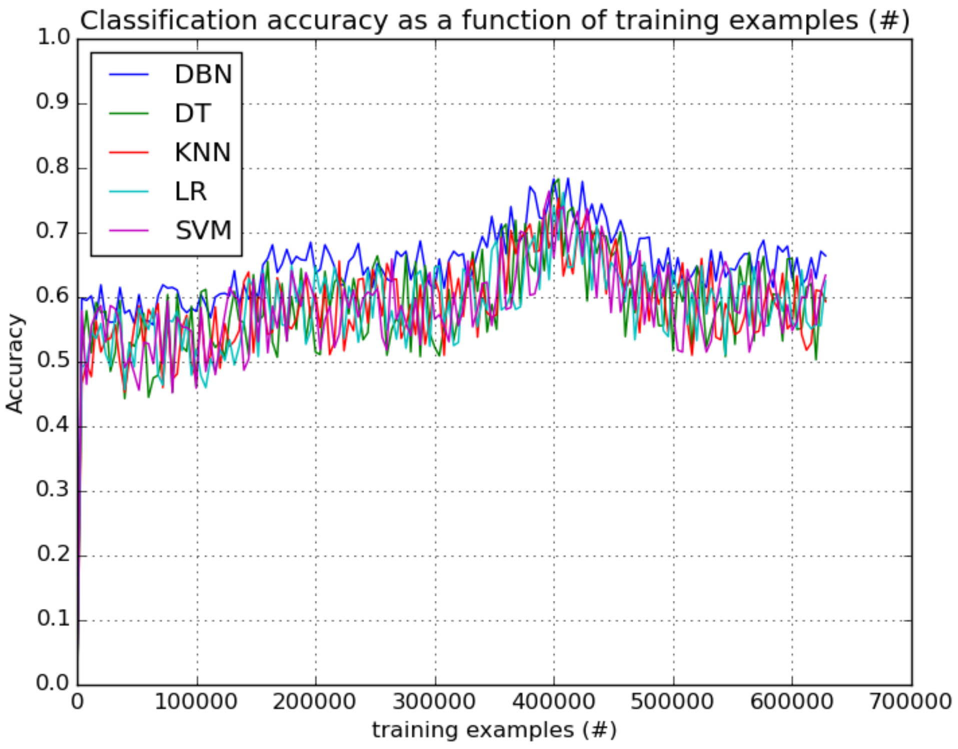


Figure 5.6: Results for online (top) and offline (bottom) learning for Twitter; mini-batches of 5,000 samples each.

accuracy of the prediction. LR showed relatively high accuracy, just 2-7% less than DBN considering the fact that the training and testing times for LR were the smallest. Considering this fact that DBN requires significantly longer time for training the model compared to LR, in some scenarios, when performance is of main concern, it might be reasonable to use LR for predicting user behaviour on the Web.

The number of hidden layers was varied from one to three. As a result, no consistency were found in influencing the number of hidden layers by the accuracy of the prediction. However, time performance of the DBN-based algorithm deteriorated with the increase of the number of hidden layers proportionally to the number of layers.

The results for different batch sizes for training are shown in Figure 5.7. The experiments were conducted for batch sizes of 500, 1000, 2000, 5000, 10000, 20000, and 50000 instances. Smaller batches led to lower accuracy compared to batches of sizes more than 2000-5000-10000 instances. As a result, batches with 5000 instances seemed to be large enough for training models with relatively high accuracy.

To summarise, the results showed that the DBN-based algorithm slightly outperformed LR, DT, SVM, and k-NN as well as online training algorithms using the same set-up such as the same set of features across three different datasets. Nevertheless, training and testing time of offline algorithms is significantly larger than training and testing time of offline algorithms which should be addressed in scenarios where time performance of a prediction is one of the main concerns. However, DBNs can be easily parallelised across multiple GPUs and CPUs which can significantly improve the time performance of a DBN-based algorithm.

5.8 Comparison to Related Studies

Below is the comparison of the results for the performance of offline and online algorithms to related studies in this field.

Offline Algorithms Complexity and Accuracy In [Page et al., 2014] the researchers compared k-NN, SVM, LR, and DBN for detecting seizure in patients using high-resolution, multi-channel EEG data. The authors explored the classification accuracy, computational complexity and memory requirements of these approaches. The researchers reported that a DBN approach showed the highest accuracy which corresponds to the results in this thesis. However, the computational complexity of their DBN as well as the time taken to train their model was significantly lower compared to the results for other approaches. As was expected, k-NN performed

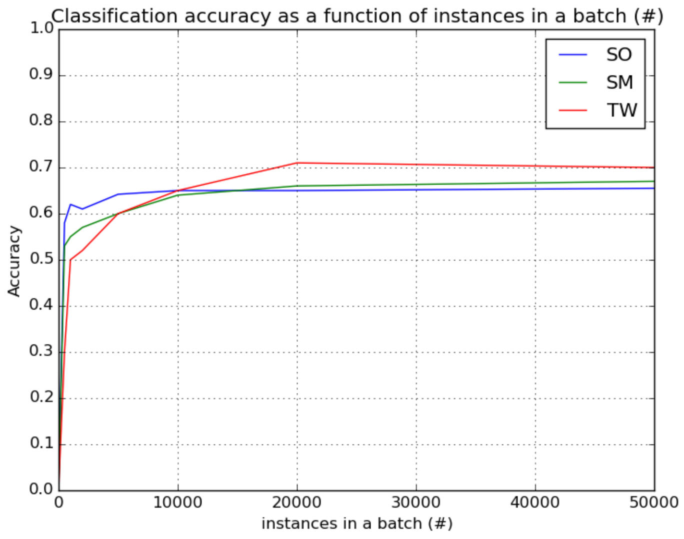


Figure 5.7: Results for varying the training batch sizes for DBN algorithm. The results are shown for the three datasets, SO - Stack Overflow, SM - Stack Mathematics, and TW - Twitter dataset.

the worst since k-NN requires one to store all the data. However, in the paper of [Page et al., 2014] k-NN required over $1,000x$ more computations compared to LR rather than $100x$ more computations in this thesis. The authors reported that SVM did the second best requiring roughly $500x$ more memory and almost equal amount of computation compared to LR which is a very surprising result since SVM is a computationally expensive algorithm, in this thesis it showed roughly 5600 times worse performance compared to LR. The results for DBN required $30x$ more computations compared to LR. In this thesis, DBN took $88x$ more computations. In this thesis, the reported results are for a DBN with two hidden layers, same as in the paper [Page et al., 2014]. The reason of choosing two layers in this thesis as well as in the paper [Page et al., 2014] was the fact that more layers did not provide any statistically significant improvements in the accuracy but increased the number of computations. Even more, in their paper training a DBN with only two layers did not show any advantage of DBN over other approaches. In this thesis, the trained DBN showed slightly better accuracy across three different datasets.

The authors in [Page et al., 2014] reported that LR performed very well both computationally and in terms of accuracy. One can argue that it was due to the nature of the problem which was easily solvable by a linear classifier. However, the boost in accuracy can be achieved by using more complicated approaches, for example DBN. Nevertheless, applying such models requires more computational and memory resources which was evaluated empirically in the experiments of this Chapter.

Online Algorithms Complexity In [Wilson and Martinez, 2003], online and offline training modes were compared across 26 applications. The training sets were from 42 to $12,000$ with an average of 1329 and median of 232. As a result, online training outperformed batch training by being more than 20 times faster and for training with more than 1000 instances, online training was 70 times faster. It was shown that the larger is a training set the slower batch training becomes with respect to online training.

In this thesis, the tested datasets were significantly larger with more than $2,000,000$ instances. The results showed that online learning can be $20 - 640$ times faster depending on the training algorithm and the size of the training dataset. Also, the experiments in this thesis were conducted for Perceptron, SGD, PA, and five batch algorithms namely DBN, k-NN, SVM, DT, LR. On the contrary, the experiments in [Wilson and Martinez, 2003] were done only for a multilayer Perceptron with gradient descent training. Generally speaking, the results in this thesis correspond to the results in [Wilson and Martinez, 2003] that online training can reach relatively same level of accuracy in less time.

Another paper presented results for online, mini-batch and full batch training of DBN for predicting user behaviour [Choi et al., 2013]. The results showed that DBN in online learning reached 72.67 ± 0.21 mini-batch learning 72.89 ± 0.32 and full-batch learning 72.86 ± 0.28. Thus, almost no difference in the accuracy. However, the authors did not report the training and testing times. Nevertheless, the reported equivalence of accuracy results for both online and batch learning corresponds to the results in this thesis.

The paper [Oza and Russell, 2001] compared the performance of online and batch versions of the popular bagging and boosting algorithms. It was demonstrated that both learning modes performed comparably in terms of classification accuracy. However, the running times for online learners were significantly faster. For example, for the two largest tested datasets of approximately $200,000$ and $400,000$ instances with 39 and 54 features accordingly the time was 20 minutes versus 7.1 hours for the first dataset and 4.3 hours versus 18.8 hours for the second dataset. The results in [Oza and Russell, 2001] correspond to the results in this thesis.

In [Read et al., 2012] *instance-incremental* or online and *batch-incremental* or batch approaches for classification were compared. As a result of empirical evaluation of datasets with up to 1 million training instances, it was concluded that *instance-incremental* methods perform similarly to *batch-learning* implementations of these methods but require less resources. The resources were defined as RAM-hour. Thus, the demonstrated results in [Read et al., 2012] correspond to the results in this thesis.

Deep Learning for Predicting User Behaviour Deep Learning showed promising results in several fields, especially image processing and natural language processing [Ciresan et al., 2012]. However, not much work has been performed on using deep learning for predicting user behaviour on the Web. For example, in [Choi et al., 2013] the authors used DBN for predicting when a sensor will be turned on in a smart home environment. The algorithm was tested on one dataset, MIT home dataset, where all inputs are binary values. Even though in this thesis a similar DBN architecture is used, the input data is considered being real-valued data rather than binary data. Even more, not only the accuracy performance of DBN approach was evaluated but time performance of the model was evaluated as well. Finally, in [Choi et al., 2013] authors used their own metric, named a rising edge accuracy (REA), for evaluating their results. On contrary, in this thesis a standard accuracy measure was used. Thus, the proposed DBN architecture represents a more general approach for training a model for predicting user behaviour on the Web. Also, the proposed DL approach in this Chapter demonstrated better performance compared to state-of-the-art ML algorithms across three datasets from Stack Exchange and Twitter similar to the DL approach in [Choi et al., 2013] where it outperformed a state-of-the-art ML algorithm, namely SVM.

5.9 Conclusions

Providing a fast but accurate prediction of user behaviour is challenging. In this Chapter the challenge is addressed by an attempt to find an efficient ML set-up for fast but accurate prediction. For this purpose, first, performance of online algorithms was compared to performance of offline algorithms and, second, performance of a DL algorithm was compared to state-of-the-art ML algorithms.

First, a method for comparing online and offline ML set-ups was proposed and then three online and five offline ML algorithms were used for a comparison across three different datasets. Second, a DL algorithm based on a DBN with a feature extraction approach was designed and then evaluated in a comparison to four other state-of-the-art ML algorithms.

The conducted empirical study showed that even though a DL algorithm, DBN-based algorithm with a feature extraction approach, demonstrated slightly better results in terms of accuracy, the time to train the model was several magnitudes higher compared to the simplest online learning algorithms. Also, the proposed approach for comparison of online and offline learning modes allowed to detect the bottle neck in the efficiency of the prediction in terms of the accuracy and time.

The main goal of this Chapter was to answer two research questions, whether online or offline learning perform better and whether a DL algorithm can outperform other state-of-the-art algorithms. As a result, a guideline for choosing a ML set-up is formulated. First, to achieve a fast prediction of user behaviour on the Web one should use online algorithms, and to achieve an accurate but much slower predictions, one should use a DL-based approach with a feature selection in a mini-batch setting. Also, the experiments demonstrated that a DL algorithm with a feature selection approach can be used in a natural way to predict user behaviour on the Web. Moreover, using a DL approach allows one to achieve higher prediction accuracy. However, further experiments have to be conducted in order to determine the most efficient way to train a DL network.

5.10 Summary

In this Chapter, the problem of choosing a ML set-up to predict user behaviour on the Web efficiently is explored. This includes comparing and then choosing an online or offline learning mode to train a model. First, a novel method to compare online and offline learning algorithms in terms of accuracy and time performance was proposed. Second, the performance of online training algorithms was compared to offline state-of-the-art algorithms in predicting user behaviour on the Web across three different datasets using the proposed comparison method. Finally, the performance of a DL approach, a DBN-based algorithm with a feature selection approach, was compared to four state-of-the-art ML algorithms where the DBN-based algorithm slightly outperformed the state-of-the-art algorithms in terms of accuracy but showed relatively low time performance.

In the next Chapter, the problem of visual exploration of human interaction is explored as well as an automatic approach for choosing visualisations for exploration is presented and discussed.

Chapter 6

Visual Data Exploration

Visualising user behaviour on the Web can facilitate understanding of interaction and relationship development between these users. Indeed, in data mining terminology such visualisation can be considered as Exploratory Data Analysis (EDA) of user behaviour. As one of results, visualising user behaviour can help to make a better informed decision by identifying and choosing features to extract and then to build a predictive model. However, due to a large number of currently available visualisations, choosing appropriate visualisations to explore user behaviour is a challenging task. The problem addressed in this Chapter is how to automatically choose visualisations in order to help people to analyse, explore and understand user behaviour on the Web.

First, a brief introduction of visualisations for the exploration of user behaviour is introduced in Section 6.1. A set of exploration tasks is summarised in Section 6.2. The proposed approach to automatically choose appropriate visualisations of user behaviour is discussed in Section 6.3. An experiment of using the proposed approach on data from *IBM Many Eyes* is presented in Section 6.4. The results of the experiment are discussed in Section 6.5. The results are compared to related studies in Section 6.6. Finally, Section 6.7 and Section 6.8 provide a conclusion and a summary of the Chapter.

6.1 Introduction

Visualisation of user behaviour is a popular topic amongst researchers. As a result, many visualisation approaches and software tools have been proposed and developed so far. The area of visualisation of human communication and social networks formed by people has been researched thoroughly. A very brief list of such visualisations with their evaluations can be found in [Donath et al., 2010; Frau et al., 2005; Lee et al., 2013; Viégas and Smith, 2004; Farrugia et al., 2011; Sathiyanarayanan and Burlutskiy, 2015a,b].

Development of the best, or more specifically the most appropriate for any given goal, single visualisation of human communication seems to be an impossible task. However, several visualisations proved to be preferable over other possible visualisations for specific tasks and communication types. For example, the visualisation of e-mail communication has been researched for several years [Jovicic, 2000; Ahn et al., 2014; Angus et al., 2012]. Visualising social networks and users' interaction networks as a dynamically changing graph is a common ground for researchers in this field [Hadlak et al., 2013]. An attempt to visualise conversations to uncover their social and temporal patterns was conducted in [Smith and Fiore, 2001]. In their work, a few visualisations, namely a *graph*, a *tree map*-like visualisation, and a *bar chart* were shown to be useful in presenting properties of newsgroups. In [Venolia and Neustaedter, 2003] the authors proposed a fusion of two visualisations for showing relationships among the sent and received messages of a conversation. Developing a fusion of visualisations consisting of treemaps, Euler diagrams, and graphs is an active area of research as well [Sathiyanarayanan and Burlutskiy, 2015a,b].

A graph model can be a powerful model to describe user behaviour on the Web. Ideally, visualising a graph model $G = (U, E)$ of user behaviour will provide a better understanding of the dynamics of human communication. As a result, formulating an appropriate prediction task will be easier for the user. Also, identifying features of user behaviour for extraction will be easier and eventually the user will use these features for building a predictive model.

The visualisation of user's data must support interaction with the data behind his/her visualisation together with flexibility to change the visualisation for the user's needs. However, a single visualisation does not suit all user's needs which is one of the main motivations to develop means to recommend visualisations to a user automatically. This motivates one to address the following research question in this Chapter:

1. How can one automatically choose appropriate visualisations to explore user behaviour on the Web?

To answer this question, an approach using Case-Based Reasoning (CBR) is chosen. The proposed approach enables one to automatically choose a few visualisations based on previous visualisations for similar datasets and then to recommend these visualisations to a user. The cases are defined as experiences of previously chosen visualisations of user behaviour on the Web. A similarity measure to find the most

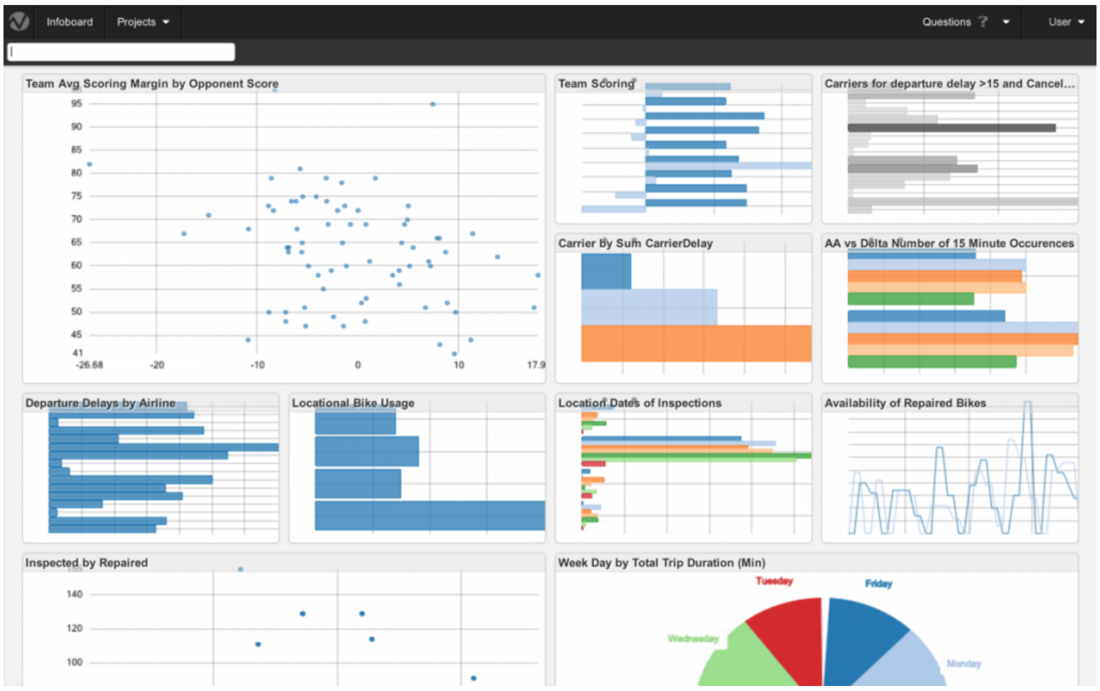

Infoboard
Projects
Questions
User
Team Avg Scoring Margin by Opponent Score
Team Scoring
Carriers for departure delay >15 and Cancel...
Carrier By Sum CarrierDelay
AA vs Delta Number of 15 Minute Occurences
Departure Delays by Airline
Locational Bike Usage
Location Dates of Inspections
Availability of Repaired Bikes
Inspected by Repaired
Week Day by Total Trip Duration (Min)
Tuesday
Friday
Wednesday
Monday

Figure 6.1: Different visualisations of user behaviour[1].

appropriate visualisations for a target dataset is constructed and applied. In this work, the web-based visualisation platform *IBM Many Eyes* is chosen to collect data on previously chosen visualisations of user behaviour on the Web. As a result of applying the proposed approach, a user is recommended a few appropriate visualisations and then he/she can decide whether to use the recommended visualisations or whether to use another visualisation for his/her model. In both scenarios, the system can utilise that information to learn from the user choices of visualisations in order to improve the recommendation mechanism. At the end of this Chapter, the results of visualisation for a sample email conversation using the proposed CBR approach are demonstrated and evaluated.

6.2 Exploration Tasks

First, the most common problems people encounter when exploring user behaviour on the Web are analysed. For this purpose, the literature on visualisation of user behaviour was studied and then *IBM Many Eyes* platform was used for understanding what kind of problems people are trying to solve by visualising user behaviour on the Web [Donath et al., 2010; Frau et al., 2005; Lee et al., 2013; Viégas and Smith,

[1]https://www.logianalytics.com/tips-tricks/how-to-choose-the-right-visualization-for-user's-data/

2004; Farrugia et al., 2011; Sathiyanarayanan and Burlutskiy, 2015a,b]. The list of these problems is below:

- *Finding a sequence of messages:* each message is considered to be an event of an atomic communication between people. To understand in what sequence messages have been sent or received, a multi-modal visualisation can enhance user's experience in exploration of his communication [Venolia and Neustaedter, 2003].
- *Discovering patterns in communication:* the user can be directed toward an answer to his/her questions regarding the patterns in which he/she has communicated with people in a particular conversation by providing him/her with visualisations where cyclic or sequential patterns of communication are represented explicitly.
- *Aggregate the information on a conversation:* if user's conversation become very long then he/she might want to see only an aggregated view of his/her conversation.
- *Understanding the granularity of time:* user's conversations exist at different time granularity. For example, the intensity of user's day-to-day conversations or user's month-to-month messages can be a feature of interest. By looking at different granularity, one can have an insight into an aggregated picture of user communication rather than being lost in user's daily messages.
- *Supporting temporal questions:* the problem of finding a particular message or a conversation of interest is common when exploring user's communication. The visualisation can allow one to answer temporal queries on user's communication easily.

The goal of visualisations of interaction between people is to get a deeper insight of the nature and properties of communication. A well-chosen visualisation of interaction can help people to find answers for their questions.

6.3 Proposed Approach for Choosing Visualisations

In this Chapter, user behaviour on the Web is considered as interaction and relationship development between people on the Web. Human interaction and relationship development is modelled as a graph $G = (U, E)$ where nodes U are associated with users and edges E are associated with interaction and relationship between the users. The graph $G(U, E)$ can be visualised by using various visualisation types V_i, for example, by a network diagram or a bar chart. Then the problem to be solved is formulated as following:

Problem Statement: *Find appropriate visualisation types V_i for the model $G_i = (U_i, E_i)$ using knowledge of chosen visualisation types V for previously visualised models of human communications $G(U, E)$.*

The choice of appropriate visualisation types V_i is achieved by applying CBR approach [Aamodt and Plaza, 1994; Petridis et al., 2014] in the context of user interaction and relationship development modelled as a graph $G(U, E)$. The proposed method allows one to choose a few of the most appropriate visualisation types V_i for a model $G = (U, E)$. The choice is dictated by previous visualisations of similar graph-based models. The classical CBR approach for problem solving has four steps [Aamodt and Plaza, 1994]:

1. Retrieving one of previously experienced cases;
2. Reusing the retrieved case;
3. Revising the chosen solution;
4. Retaining the new experience for further usage.

Adapting the classical CBR approach for choosing visualisation types for a model of user's communication requires one to define what a case is for this particular problem domain.

6.3.1 Visualisations as Cases

A case c_i is a pair consisting of a model $G_i(U, E)$ and a visualisation type V of the model $G_i(U, E)$:

$$c_i = (G_i(U, E), V) \qquad (6.1)$$

Each model $G_i(U, E)$ of the collected cases was visualised in the past using a visualisation type V. The goal is to choose k the most appropriate visualisation types $\{V_{input}\}$ for a new input model G_{input}. For this purpose, the input model G_{input} is compared to all visualised in the past data models $G(U, E) = \{G_i(U, E)\}$ using a similarity measure D (see Subsection 6.3.2). As a result of the comparison, the closest model $G_{closest}$ is chosen from $G(U, E)$ for the input model G_{input} in accordance to the similarity measure D. Then k visualisation types $\{V_{closest}\}$ chosen in the past for visualising the closest model $G_{closest}$ are chosen for visualising the input model G_{input}.

It is important to mention that the choice of visualisation types V for a particular model $G(U, E)$ depends on the user's task. In this work, it is assumed that the user explores the model and the task can be one of the following: finding a sequence of messages, discovering patterns in communication, aggregating the information on a conversation, understanding the granularity of time, and performing time filtering queries (see Section 6.2). For each of these tasks there is a set of appropriate visualisation types $\{V_i\}$ which depends on the properties of the model $G(U, E)$ such as the properties of the nodes U and the properties of the edges E by which the nodes U are connected.

6.3.2 Similarity Measure

In order to decide which visualisation types are the most appropriate to use for visualising user's communication in the form of a model $G_{input}(U, E)$, k-Nearest Neighbours (k-NN) algorithm is executed. The idea of this algorithm is in choosing k neighbouring models $G_n = \{G_{n1}, ..., G_{nk}\}$ for the input model G_{input} as the most similar to the input model. Each neighbouring model has a visualisation type associated to it.

For choosing a neighbouring model G_{ni} of the input model G_{input}, a distance metrics d_j between the input model G_{input} and a historically visualised model $G_j(U_j, E_j)$ is introduced:

$$d_j = difference(G_{input}, G_j) \tag{6.2}$$

Where $difference(x, y)$ is a function which calculates a difference, or a distance d_j, between x and y as a real number. Choosing this function for a particular pair of datasets depends on the available data in the models $G(U, E)$, computational complexity of calculating the distance d, the accuracy this metrics leads to. In this Chapter, Euclidean metrics is used due to its simplicity and the data available in the model $G(U, E)$, and the model is considered to be a point in Euclidean space. Thus, Euclidean distance between two models can be defined as the distance between two points. For example, for two dimensional space the distance between two points (x_1, y_1) and (x_2, y_2) is $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

6.3.3 Recommending Visualisations

After the user is given the k most similar visualisation types $\{V\}$ for his/her input model $G(U, E)$, the user can choose either none of them (the recommendation was bad) or one or even all k recommended visualisation types. This choice will form a new case c_j and will be saved in a knowledge database.

6.4 Experiment Outline

In this experiment, the proposed approach is implemented and then it is tested on a real dataset crawled from *IBM Many Eyes* platform. Finally the automatic choice of visualisation types is evaluated by participants.

6.4.1 IBM Many Eyes as a Source of Visualisations

IBM Many Eyes is a visualisation platform where a user can upload his/her dataset in a format of comma-separated values or text and then he/she can choose a visualisation type with an ability to customise the visualisation. The platform provides twenty one different visualisations including the most common visualisations such as a bar chart, bubble chart, tree map, graph, scatter plot, etc. A user can choose a visualisation type for his/her dataset and then he/she can manipulate his/her visualisation. For instance, the user can flip axes, zoom in or zoom out, and interact with the visualisation.

According to *IBM Many Eyes* recommendation, the procedure of using the service is as follows:

- Upload user's dataset. The dataset can be a spreadsheet or a text file.
- Select a visualisation for user's dataset or use a visualisation recommended by *Many Eyes*.
- Share user's visualisation over the Web. A visualisation can be embedded in a blog or be shared on *Facebook* and *Twitter* easily.

Since its start in 2007 *Many Eyes* has attracted thousands of people to upload and visualise their datasets.

6.4.2 Crawling Data

The data for the experiment was collected by writing and then executing a web crawler for downloading ∼0.5 million datasets from the website of *IBM Many Eyes*. The number of calls was limited to 100,000 calls per day and the crawling process took almost five days. Then the extracted data was assessed for quality and validity. Some datasets and visualisations could not be downloaded but the total number of those datasets was ∼2%.

6.4.3 Extracting Cases

On the 20th of October, 2014 *IBM Many Eyes* had 490,982 datasets and only 186,314 visualisations shared by 96,710 users. Thus, less than 38% of all the uploaded datasets were visualised. In this work, only the visualisations of human communication were of interest. Therefore, the datasets were filtered using keywords *'conversation'*, *'email'*, *'chat'*, *'twitter'*, and *'facebook'*. In total, there were 5,819 datasets and only 2,322 visualisations for this set of search words.

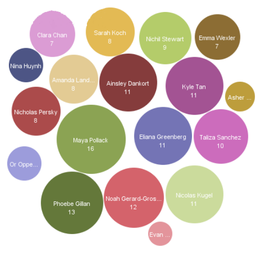
After extracting the 2,322 visualised datasets of human communication, all of them were analysed to understand what kind of information can be extracted to fit the graph model $G = (U, E)$. For the first iteration, it was decided to simplify the graph model to a set of two features $(|U|, |M|)$ where $|U|$ is the number of users and $|M|$ is the number of messages in their communication. The reason for such simplification was the fact that this information was presented in many datasets and could be extracted.

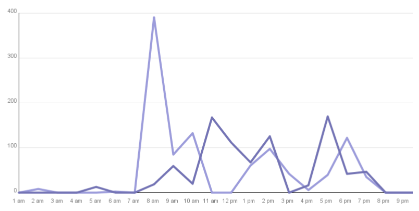
Then, the extracted 2,322 datasets with the visualisations were filtered again. At this step, the datasets without information on the number of users $|U|$ and the number of messages $|M|$ were removed. This step allowed to map the filtered datasets D_V to the proposed model $(|U|, |M|)$ of human communication: $D_V \to (|U|, |M|)$. The number of users $|U|$ and the number of messages $|M|$ in a case c_i were defined as follows:

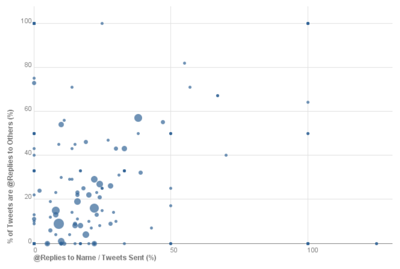
- *Number of users:* $|U|$ the number of people involved in a conversation which is the number of nodes $|U|$ of the graph $|G|$.
- *Number of messages:* $|M|$ the number of messages sent and received by the users U. For example, if a user $u_1 \in U$ sent a message $m_1 \in M$ to another user $u_2 \in U$ and received a reply $m_2 \in M$, the number of messages is two, $|M|$=2.

Unfortunately, it seemed very hard to automatise the process of matching the automatically extracted datasets D_V to the graph model G_i since the uploaded datasets D_V were very diverse and in many cases they were obfuscated for more anonymity. Also, some datasets and their visualisations seemed to be blindly thrown into the tool or had not much information about the visualisations and datasets. However, 40 cases $c = ((|U|, |M|), V)$ were successfully extracted from the datasets D_V for this study (Table 6.1).

Some examples of such extracted models and their visualisations are shown in Figure 6.2. In Figure 6.2a there is a *Bubble Chart* visualisation of a conversation between 18 users, who are represented as circles. The size of the circles is proportional to the number of messages sent by each actor. Figure 6.2b represents a *Line Graph* where the number of email messages M received by one actor U over a fixed time is shown. In Figure 6.2c the *Scatter Plot* shows users U as circles with their


(a) Bubble chart.


(b) Line graph.


(c) Scatter plot.

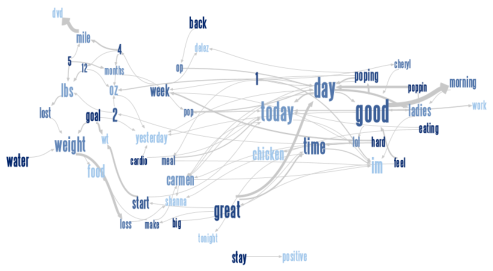

(d) Graph-like visualisation.

Figure 6.2: Examples of visualisations of conversations in Many Eyes.

sizes proportional to the number of messages M sent to each of them. Figure 6.2d depicts a *Word Cloud* with the most common words in communication between two people over time.

All the 40 cases extracted from *Many Eyes* models of human communication were visualised by eight different types of visualisations:

- *Bar Chart*,
- *Matrix Chart*,
- *Line Graph*,
- *Bubble Chart*,
- *Treemap*,
- *Word Cloud*,
- *Block Histogram*, and
- *Network Diagram*.

Thus, the visualisation types had five cases each, total forty visualisations. The performance of the proposed CBR approach for choosing a visualisation for an input model G_{input} was tested using the extracted 40 models and their visualisations as cases.

Table 6.1: Conversation datasets and their visualisations.

Keywords	Datasets	Visualisations	Cases
Conversation	459	71	11
Email	385	112	2
Chat	650	112	10
Twitter	1,060	707	15
Facebook	3,265	1,320	2
Total	5,819	2,322	40

6.4.4 Case-Based Reasoning Step

An algorithm based on Case-Based Reasoning (CBR) with k nearest neighbours (k-NN) was used in order to predict visualisation types a user would choose for a new uploaded model of human communication. First, there is a need to choose the parameter k of the k-NN classifier. The choice of k is non-parametric and a general rule of thumb for choosing the value of $k < \sqrt{N}$ where N is the number of instances in the training dataset [Hassanat et al., 2014]. On the other hand, small values of k lead to high sensitivity of a k-NN classifier to noise [Hassanat et al., 2014]. Thus, a rule of thumb is used for choosing k such that $k = \sqrt{N}/2$, where $N = 40$ since there are 40 cases. Substituting $N = 40$ in this equation gives $k = \sqrt{40}/2 \approx 6.32/2 \approx 3$. As a result, three ($k = 3$) neighbouring models $G_n = \{G_{n1}, G_{n2}, G_{n3}\}$ were chosen and their types of visualisation $V_n = \{V_{n1}, V_{n2}, V_{n3}\}$ were used for visualising a user's model G_{input}.

For simplicity, each model G was associated to a point with coordinates $\{|U|, |E|\}$ in Euclidean space. Thus, the distance metric d_j between two models G_{input} and G_j is the Euclidean distance between the two points x and x_j associated to the models G_{input} and G_j correspondingly:

$$d_j = \sqrt{\omega(x - x_j)^2} \tag{6.3}$$

Where d_j is an Euclidean distance between the two points, the point representing an input dataset $x = (|U_{input}|, |E_{input}|)^T$ and the point representing a historical dataset $x_j = (|U_j|, |E_j|)^T$, $j \leq 40$ where 40 corresponds to the number of cases, or the number of historical datasets with corresponding visualisations. The points x and x_j have a number of users $|U|$ and events $|E|$ as their coordinates. The vector $\omega = (\omega_1, \omega_2)$ represents the importance of the features, namely the number of users $|U|$ and the number of events $|E|$, for comparing two models. The weights in the vector ω are represented as numerical values between 0 and 1. In this work,

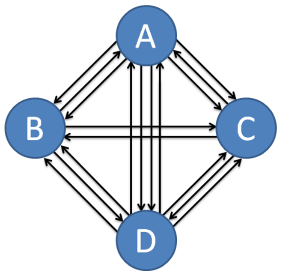


Figure 6.3: A multigraph representing the email conversation between four users.

the weights with equal stress on importance for both the number of users $|U|$ and messages $|E|$ were assigned. Thus $\omega_1 = 0.5$ and $\omega_2 = 0.5$. The result of the k-NN algorithm is a choice of the closest visualisation types in the described metrics for an input model based on previous visualisations. Since the k-NN algorithm was applied, the user was given the k closest visualisation types for his model of human communication $G(U, E)$.

6.4.5 Evaluation

The results of the prediction were evaluated in a pilot empirical study by questioning people on visualisation types for a newly uploaded model of human communication they would prefer and why. Then the answers were compared with the results of the automatic choice of the visualisation types by the CBR approach.

An Example for Visualisation: An Email Conversation

In this section, the work of the proposed CBR approach for a sample e-mail conversation is shown. The conversation had four users $U = \{A, B, C, D\}$ and it had eighteen messages sent between the users (Figure 6.3). The order of this directed multi-graph of the conversation $|U|$ is 4 and the size $|M|$ equals to 18. Thus, the dataset corresponds to the following point: $(|U|, |M|) = (4, 18)$.

First, the distance between the sample dataset and each of the 40 case datasets was calculated, it resulted in 40 values. Since k=3 for the k-NN algorithm, the three nearest neighbours for this dataset were chosen by selecting the three smallest numbers from these 40 values calculated in accordance with Formulae 6.3.

$$d_1 = \sqrt{0.5(4-1)^2 + 0.5(18-18)^2} = 2.1$$
$$d_2 = \sqrt{0.5(4-1)^2 + 0.5(18-11)^2} = 5.4$$
$$d_3 = \sqrt{0.5(4-1)^2 + 0.5(18-10)^2} = 6.0$$

Thus, the distances $\{d_1, d_2, d_3\}$ for the three nearest neighbours were $\{2.1, 5.4, 6.0\}$. The visualisation types for these datasets chosen in the past were *Line Graph*, *Bubble Chart*, and *Matrix Chart* accordingly. Ideally, each of the three neighbouring datasets would have had the same visualisation type. However, it seems that for the datasets similar to the sample dataset users decided that these three visualisation types are the most appropriate. In order to understand why the proposed CBR approach gave these visualisations and how appropriate these visualisations are for the corresponding datasets, an evaluation was designed and conducted.

Human Experiments

The goal of the designed pilot evaluation was:

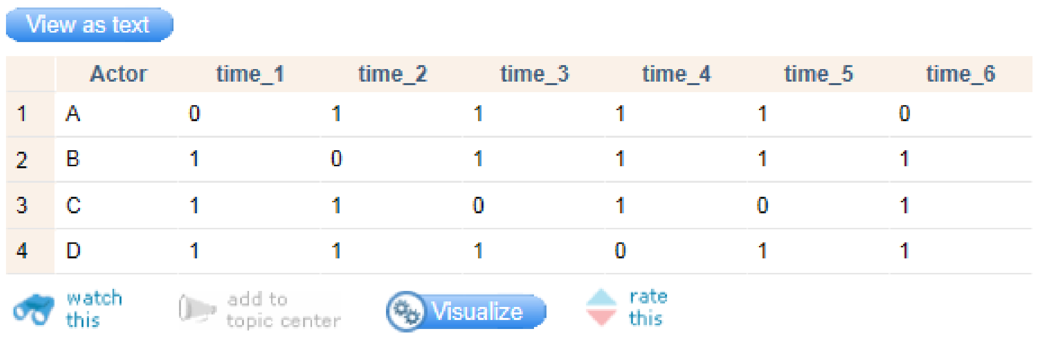
- Evaluate the choice of conversation visualisations from eight different visualisations, and
- Compare the manual selection of a visualisation by participants to the automatic choice made by the proposed CBR algorithm.

As the main part of the evaluation, four participants were asked to answer seven questions for each of the eight alternative visualisations of the sample email conversation described above (Figure 6.3).

At the beginning of an interview, each participant was introduced to the eight basic different types of visualisations used in *Many Eyes*: *Bar Chart*, *Matrix Chart*, *Line Graph*, *Bubble Chart*, *Tree Map*, *Word Cloud*, *Block Histogram*, and *Network Diagram*. Then the participant received an explanation for the meaning of graphical elements in each of the visualisations. A simple example of a visualisation for a conversation between two people was given in order to confirm that the participant understood the layout of all eight visualisation types.

Finally, the eight visualisations designed using *IBM Many Eyes* for the sample email conversation described above (Figure 6.3) were given to the participant (Figure 6.5) and the following seven questions were asked:

- How many people participated in the conversation?
- How many messages were sent?
- Who initiated the conversation?
- Who sent the second message?
- Who sent the third message?
- How many time steps there were in the conversation?
- How many messages did the person A sent to the person B?


View as text

	Actor	time_1	time_2	time_3	time_4	time_5	time_6
1	A	0	1	1	1	1	0
2	B	1	0	1	1	1	1
3	C	1	1	0	1	0	1
4	D	1	1	1	0	1	1

Figure 6.4: The dataset uploaded to *Many Eyes* as a spreadsheet.

Table 6.2: The mean value and the standard deviation for each visualisation.

Question / Visualisation Type*	1	2	3	4	5	6	7	8
1. The Mean Value of the questions score	3.3	4.6	2.8	4.0	4.1	4.0	4.5	4.7
2. The SD** of the questions score	1.7	0.5	1.1	1.1	0.9	1.0	0.8	0.3

*Visualisation Type: 1) bar chart, 2) matrix chart, 3) line graph, 4) bubble chart, 5) treemap, 6) word cloud, 7) block histogram, and 8) network diagram.
**SD - standard deviation.

Each question was ranked by the participant on a scale from *zero* to *five*. The higher the value, the easier the visualisation for answering questions over the visualised conversation. In other words, 0 meant that the visualisation was very confusing for answering the question and 5 meant that it was easy to answer the question using the particular visualisation. The results of the average scores for the answered questions are in Figure 6.6. The mean value with standard deviation of the scores for each of the visualisations are in Table 6.2. At the end of the interviews, the participants were requested to give their feedback on what they thought about the visualisations.

The evaluation showed that the *Network Diagram*, *Matrix Chart* and *Block Histogram* outperformed other visualisations with the average scores of 4.7, 4.6, 4.5 and standard deviations of 0.3, 0.5, 0.8. The relatively low values of the standard deviations meant the answers of the participants were quite consistent. The worst visualisations with regard to the participants' answers were *Line Chart* and *Bar Chart* with 2.8, 3.3 for the average scores and 1.1, 1.7 for the standard deviations.

In the next Section, the results of the evaluation are discussed and then the results are compared to the choice of the CBR approach.

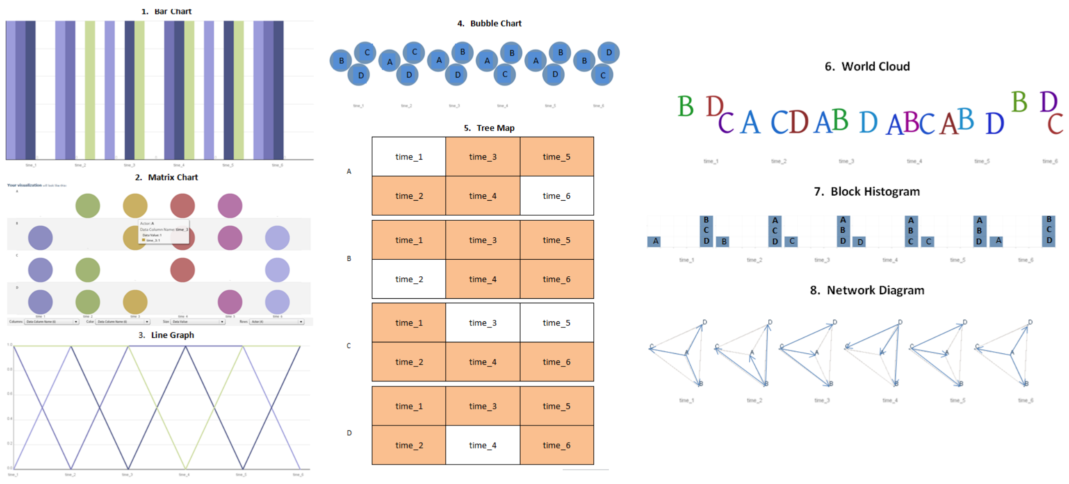


Figure 6.5: The eight visualisations of the email conversation for the evaluation: 1) bar chart, 2) matrix chart, 3) line graph, 4) bubble chart, 5) treemap, 6) word cloud, 7) block histogram, and 8) network diagram.

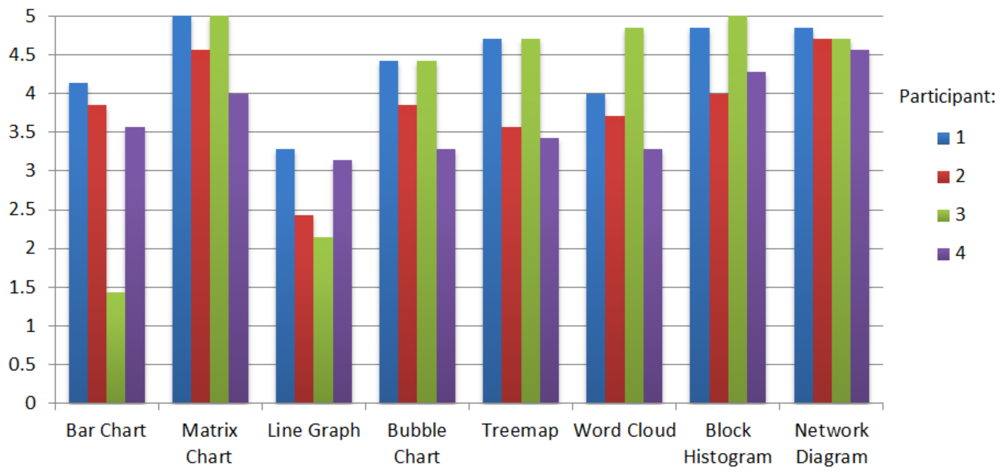


Figure 6.6: The results of the evaluation for each of the four participants by visualisation.

6.5 Results and Discussions

The results of the automatic choice of the CBR approach and the human evaluation showed a discrepancy, which can be explained by the design of the CBR approach and the conducted evaluation. The three nearest visualisations for the sample email conversation were *Bubble Chart*, *Line Graph* and *Matrix Chart*. In these cases, the users visualised a) the number of his Facebook and email messages received by days, b) the number of emails received over the time, and c) the number of Facebook messages received during the day at different times. On the contrary, the evaluation showed that the *Line Graph* visualisation of the sample email conversation was not suitable for answering the questions over the conversation. Thus, the questions the user asked over his visualisation are very important and must be fixed or known in advance. In this Chapter, the asked questions were fixed to five (Section 6.2).

Also, the application of the proposed CBR for choosing a visualisation type of a conversation showed a necessity in further elaboration on the following aspects: 1) refinement of the case model for visualisations of conversations, 2) what attributes other than number of users and messages in a conversation are important for a case, 3) how many cases are required for statistically significant results of the CBR, and 4) how to evaluate the results of the CBR for visualisation of user's conversation.

IBM Many Eyes lacks flexibility in terms of visualisation since the visualisation types and formats for uploading data are limited. From another side this simplicity makes *IBM Many Eyes* easily accessible by a wide audience without any previous experience in visualisation. This fact motivates one to look into the visualisations offered by *IBM Many Eyes* for visualising conversations and developing a recommendation system for advising visualisations of users' data and conversations in particular.

6.6 Comparison to Related Studies

Several authors tried to classify various visualisation techniques and provide users with guidelines on which visualisation to choose for which data and which problem. For example, in [Keim, 2002] the author classified visualisation techniques into five groups, namely (1) standard 2D and 3D graphics, (3) geometric techniques, (2) icon-based techniques, (4) dense pixel techniques, and (5) hierarchical techniques. The author provides with a guideline for a visualisation expert based on the introduced classification. On contrary, the approach proposed in this Chapter allows one to recommend a few visualisation types for a data automatically. Thus, a user does not need to be an expert in data visualisation and exploration since the user is given a choice of visualisation types selected by an algorithm.

Development of visualisation recommender systems is an area of active research. For example, in [Vartak et al., 2015], a vision on automatic identification and visualisation recommendation for a particular analytical task is discussed. This work provides with measures to use for recommending visualisations. These include data characteristics, intended task, domain knowledge, visual ease of understanding, and user preference. In this Chapter, data characteristics of the previously visualised datasets were chosen to be the main parameters for choosing visualisations for recommendation. Even more, a proof-of-concept experiment on real dataset from *IBM Many Eyes* along with evaluation of recommended visualisations by participants was conducted in this Chapter.

Another work on developing a visualisation recommender system was proposed in [Kaur et al., 2015]. The authors introduced a semi-automatic visualisation recommendation based on captured domain knowledge. In this Chapter, visualisations are recommended to a user which are based on historically visualised datasets extracted automatically from *IBM Many Eyes*, a visualisation as a service platform.

6.7 Conclusions

Visualising user behaviour to facilitate the process of understanding patterns of human communication and interaction is challenging due to the diversity of visualisation types and settings. Even though modelling user behaviour as a graph and then visualising this graph in a form of network diagram is a common practice, several alternative visualisations exist and are useful for understanding user behaviour. In order to choose a visualisation, previous experience of visualisations is helpful to build an approach for choosing visualisations automatically. Such an approach based on CBR was proposed and explored in this Chapter [Burlutskiy et al., 2014b]. The conducted empirical study demonstrated usefulness of the proposed approach in assisting people to choose or recommend visualisations for their datasets and models. Finally, the results of this Chapter showed evidence that CBR is capable of using this previous experience to improve the appropriateness of recommended visualisations.

6.8 Summary

In this Chapter, how to use historical datasets of human communication and their visualisations for automatically choosing and recommending appropriate visualisations for a new input dataset of human communication was shown. The proposed approach was tested on data crawled from *Many Eyes*, a large visualisation platform built by *IBM*. Also, the automatic results of visualisation recommendation were evaluated by human participants.

In the next Chapter, conclusions on predicting user behaviour on the Web are presented and discussed.

Chapter 7

Conclusions

7.1 Introduction

This Chapter concludes the thesis with a summary of the conducted work, revisits research questions and presents the main findings and contributions of the work accomplished. Finally, a few directions for further potential research work are described. The Chapter is organised as follows: in Section 7.2 an overall summary of the thesis is introduced; then the main findings and contributions are discussed in Section 7.3; finally, a list of ideas for future research work is suggested in Section 7.4.

7.2 Summary of Thesis

The thesis investigated the problem of predicting user behaviour on the Web. This involves modelling, visually exploring, feature extracting, and predicting user behaviour on the Web using ML techniques. The developed process of predicting user behaviour in this thesis is presented in Figure 7.1.

First, related works and the background of the problem were introduced in Chapter 2. This included an overview of predicting user behaviour on the Web with a set of popular prediction tasks. Then models of user behaviour were reviewed and visual data exploration of user behaviour was discussed. Finally, feature extraction techniques and the state-of-the-art ML methods for predicting user behaviour on the Web were presented.

Second, a model of user behaviour, more specifically a model of human interaction and relationship development on the Web, was proposed in Chapter 3 (see Figure 7.1 module Data). The core of this model is a novel Time-Varying Attributed Graph (TVAG) proposed in this thesis. The model consists of two TVAGs, an *interaction graph* and a *relationship graph.* The proposed model allows one to capture both structural and temporal properties of user behaviour on the Web as well as facilitates visual exploration of user behaviour on the Web. Even more, the model facilitates feature extraction from user behaviour on the Web. Then these features are used for building predictive models. In order to demonstrate how one can use the proposed TVAG-based model, three different platforms were modelled using the proposed model. First, the largest Q&A platform, *Stack Exchange*, second, the largest micro-blogging platform, *Twitter*, and finally the largest social network, *Facebook* were modelled using the proposed TVAG-based model.

Third, the proposed TVAG-based model was used in Chapter 4 for extracting features influencing user behaviour on the Web (see Figure 7.1 modules Feature Extraction, Feature Space Representation, Feature Selection). First, these features were classified *semantically* into user, message, relationship, structural, and temporal groups. Then all these features were classified in accordance to their *computational complexity* into raw, easy, and hard features. Finally it was shown how these groups of features influence the accuracy of prediction. As a result, a guideline for extracting features for predicting user behaviour on the Web was proposed. The proposed guideline allows one to extract features efficiently in terms of both accuracy and time performance. The experiments were conducted for three different datasets, two from *Stack Exchange* and one from *Twitter*.

Fourth, the efficiency of various ML set-ups for predicting user behaviour on the Web were explored in Chapter 5 (see Figure 7.1 module ML set-up). First, an offline (batch) learning mode was compared to an online learning mode. In the absence of guidelines to compare these two different learning modes, a method for comparison of online and offline (batch) learning modes was proposed and evaluated. The comparison was performed for three large datasets, two from *Stack Exchange* and one from *Twitter*.

Fifth, a DL algorithm, an algorithm based on a DBN with a feature selection approach, for predicting user behaviour was designed (see Figure 7.1 module ML set-up). Then the performance of this algorithm was compared to four state-of-the-art ML algorithms across three different datasets, two from *Stack Exchange* and one from *Twitter*.

Finally, the problem of visual exploration of user behaviour, more precisely the problem of choosing visualisations for exploration of user behaviour was researched, however, this is an ongoing and additional work on top of the main work in the thesis. As a result, an approach for an automatic choice of visualisations was proposed and evaluated in Chapter 6 (see Figure 7.1 modules Data Analysis/Exploration). The approach is based on Case-Based Reasoning (CBR). A dataset from IBM Many Eyes platform was extracted to demonstrate the proposed approach. The results of the automatic choice of visualisations were evaluated by a few participants in an empirical study.

7.3 Main Findings and Contributions

This section revisits the research questions introduced in Chapter 1 (Section 1.3) and demonstrates what are the main findings of this thesis resulted from answering the research questions. The first addressed research question was the following:

1. How can one model human interaction and relationship development in order to capture, explore, and facilitate understanding of user behaviour on the Web?

A *model* of user communication and relationship development was developed in Chapter 3. The main reason to propose such a model was the fact that graphs are a powerful tool for the representation and exploration of social media networks. However, the importance of temporal properties of such graphs is often underestimated. First, a Time-Varying Attributed Graph (TVAG) was introduced and then a TVAG-based model for modelling interactions M and development of relationships R between people U was proposed. The time-varying attributes of TVAG allow one to capture the temporal properties and structural characteristics of human interaction and relationship development. The proposed model consists of two graphs, namely an *interaction* graph $G_{int} = (U, M)$ capturing user communication and a *relationship* graph $G_r = (U, R)$ capturing relationship development.

A variety of social platforms and online communities can be easily transformed into the proposed model which was demonstrated for three of the most popular social platforms such as *Twitter*, *Facebook*, and *Stack Exchange* platforms (see Section 3.4). This transformation does not lose any data but allows one to capture and explore temporal user behaviour on the Web. On the other hand, it was noticed that the three platforms of interest ignore the temporal nature of a substantial number of attributes. Nevertheless, regardless of data models and data structures used by a social platform, the proposed TVAG-based model is useful to capture user behaviour on the Web. The main contribution of the proposed TVAG model is the time varying component of features of the graph nodes and edges.

The graph nature of the model facilitates feature extraction as a part of ML

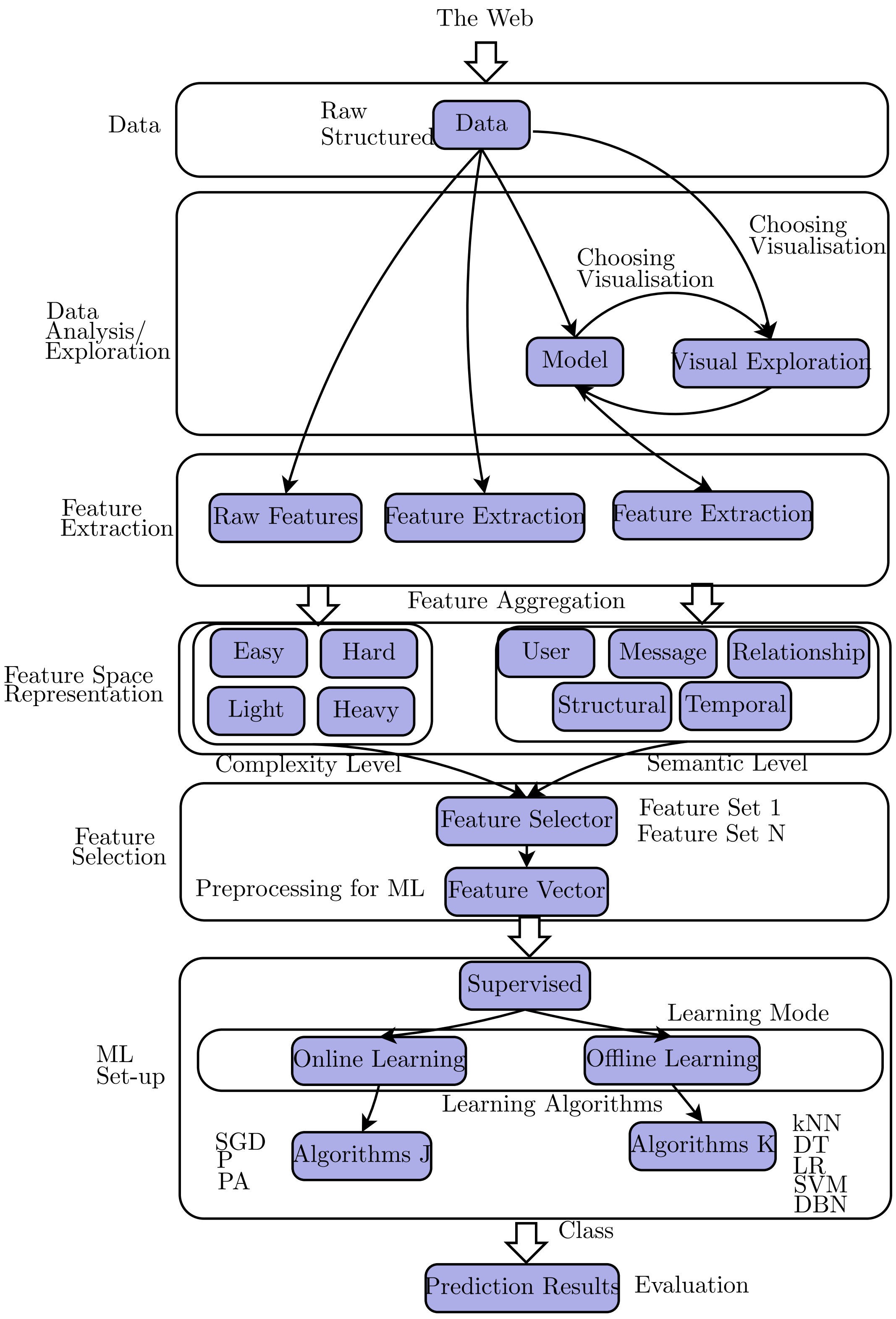


Figure 7.1: The process of predicting user behaviour on the Web.

process. In ML, a feature extraction step is an important step which often determines the success of the whole ML process. It was shown in the literature that structural features of human communication are of high impact on many prediction tasks [Cai and Chakravarthy, 2013; Hu et al., 2013; Teinemaa et al., 2015]. However, in real world applications the number of nodes and edges in the graphs G_r and G_{int} may reach millions and it is usually very computationally expensive to calculate the aforementioned features for the whole graph G defined on time interval $\mathbb{T}$. Nevertheless, explicit temporal properties of both nodes and edges allow one to find a smaller subgraph G_T defined on a time interval $T \in \mathbb{T}$ and calculate the features for this subgraph rather than for the whole graph G. Thus, temporal features of the graph allow one to calculate and then use the structural features even where the graph G is large.

Data reduction approaches are important for dealing with large datasets for training a classifier. Such approaches involve feature selection techniques, sampling, dimension reduction algorithms. The second research question was a question on how one can use the proposed TVAG-based model of user behaviour for extracting features efficiently:

2. How can one construct a set of features for an *accurate* but *fast* prediction of user behaviour on the Web?

As a result of answering this research question, a guideline for an efficient feature extraction from the proposed model of user behaviour was introduced in Chapter 4. This approach allows one to extract *semantic* groups of features associated with users, their relationships, messages, temporal, and structural features. Also, it was shown how to calculate computational complexity of these features for grouping them into *computational complexity* groups, *easy* and *hard* feature groups. The proposed approach for feature extraction showed that using structural and temporal features is advantageous for the accuracy of a prediction but extracting such features can significantly increase the time of a feature extraction step. As a result, a guideline for selecting and constructing an efficient feature set based on the proposed classification was proposed. The proposed guideline of feature engineering involves the following steps (see Subsection 4.6.4 for more details):

1. Divide all *raw* features into *light* and *heavy* features;
2. Reduce the dimension for *heavy* raw features, for example, by creating a *'bag of words'* and then possible dimension reduction using Principal Component Analysis (PCA);

3. Calculate *easy* features (calculations are performed in constant time, computational complexity is $O(1)$);
4. Calculate *hard* features (computational complexity is harder than $O(1)$).

A predictive model can be trained on raw, easy, raw and easy, and then all features including the hard features. Also, a semantic grouping inside each of computational complexity group can be used. As a result, a set of features leading to the highest accuracy but with a reasonable time to extract and calculate these features can be derived.

The third research question was addressing how do temporal features affect prediction performance:

3. How important are temporal features for predicting user behaviour on the Web in terms of accuracy and time efficiency?

The proposed approach for feature extraction showed that using temporal features is advantageous for the accuracy of a prediction but extracting such features can increase the time of a feature extraction step. This was shown, first, by predicting users' response time for the largest Q&A platform, *Stack Exchange*, using the proposed framework, where *temporal* features played an important role to achieve high accuracy. Second, predicting tweets that will be retweeted showed that *structural* and *temporal* features are important. However, it was shown that extracting temporal and structural features is a computationally expensive task.

Choosing a ML set-up for predicting user behaviour on the Web involves making several decisions. First, one has to decide which learning mode to use, online or offline learning. Thus, the fourth research question was:

4. How do *online* and *offline* algorithms compare in terms of their time complexity and accuracy performance in predicting user behaviour?

A procedure for comparison of *online* and *offline* learning modes was proposed in Chapter 5. As a part of this procedure, Algorithm 2 is applied to online learners and then the same algorithm is repeated for offline learners. (see **Algorithm** below). The outputs of running this Algorithm, the accuracy A_{av}, training time T_{tr}, and testing time T_{tt}, allow one to compare the performance of online and offline learning modes. It was shown in an experiment (see Section 5.6) across three different datasets that *online* learning mode can be much faster than *offline (batch)* learning mode without much loss in the accuracy of prediction.

Understanding whether a DL approach can be more advantageous compared to the state-of-the-art algorithms was the fifth question:

Algorithm Calculating the accuracy of learners

```
1: Initialise training and testing times: $T_{tr} = 0, T_{tt} = 0$
2: Sort the data $D$ chronologically from the oldest to the latest;
3: Divide the data $D$ into $m$ equal batches $D_i$
4: for i=1,2,..,m do
5:     Divide each batch $D_i$ into a training set $D_{itr}$ and test set $D_{itt}$;
6:     Train a learner $l_i$ on $D_{itr}$;
7:     Record the time taken for the training $T_{itr}$;
8:     Update the training time $T_{tr} = T_{tr} + T_{itr}$;
9:     Test $l_i$ on $D_{itr}$;
10:    Record the time taken for the testing $T_{itt}$;
11:    Update the testing time $T_{tt} = T_{tt} + T_{itt}$;
12:    Calculate the average accuracy $A_i$ over $D_{itr}$;
13: Calculate the average accuracy $A_{av}$ over all $A_i$
```

5. How does the performance of state-of-the-art ML algorithms compare? Is deep learning advantageous compared to other algorithms?

A DL approach for predicting user behaviour on the Web was introduced in Chapter 5. Then it was shown that this approach can be superior to other ML methods in terms of accuracy but slow in training and predicting (see Section 5.6). Also, it was shown that LR is fast and allows one to achieve slightly inferior accuracy in much shorter time compared to the other state-of-the art algorithms including a DL approach.

Finally, the last addressed question was the following:

5. How can one automatically choose appropriate visualisations to explore user behaviour on the Web?

A method based on CBR for choosing visualisations of user behaviour was proposed in Chapter 6. The feasibility of the method was demonstrated for a visualisation-as-a-service platform, IBM Many Eyes. The results of the proposed method were evaluated by participants. As a result, an application of the proposed method showed that visualisations can be chosen and then recommended to a user automatically. However, answering the last research question is an ongoing and additional work to the main work conducted in this thesis.

Returning to the main research question addressed in this thesis:

"How can one model, explore, and predict user behaviour on the Web efficiently using data mining techniques?"

The process of answering this research question in this thesis has resulted the following methodology. To sum up, it is advantageous to follow a ML process tailored specifically to predict user behaviour on the Web (see Figure 7.1). This process includes modelling user behaviour in a form of two Time-Varying Attributed Graphs (TVAGs), an interaction graph and a relationship graph. This graph-based model enables one to choose automatically and then use visualisations to investigate user behaviour on the Web in a straightforward way as well as extracting features associated with such user behaviour. Then these features must be classified semantically and in accordance to the computational effort of their extraction. This classification allows one to find a set of features which leads to an efficient prediction where there is a trade-off between the accuracy of a prediction and the time performance of the prediction. Finally, it was shown that a DL algorithm, a DBN-based algorithm with a feature selection approach, is advantageous for predicting user behaviour on the Web. However, building a model for large datasets can take substantial time. In instances where time performance is crucial, online learning can be advantageous as it can be several multitudes faster without much loss in accuracy of prediction.

7.4 Future Directions

The work presented in this thesis demonstrated a methodology for predicting user behaviour on the Web (see Figure 7.1). The work showed how one can achieve an efficient prediction for three popular platforms, including Twitter, Facebook, and the largest Q&A Stack Exchange website, Stack Overflow. Despite the successful results achieved, several directions for further research exists. A list of possible future works is as follows:

- *Modelling.* The proposed TVAG-based model can be explored further formally. For example, the expressive power and efficiency of algorithms on the proposed model can be studied and validated. Also, one can compare the proposed model with other state-of-the-art models, for example, dynamic models.
- *Visualisation.* In future work, it is necessary to test the proposed CBR approach for other datasets in order to understand how to tune the approach for more accurate choices of visualisations. For this purpose, clarification of what a case of a visualisation is needed, temporal and context attributes of human interaction must be introduced. Only a small set of visualisations crawled from *Many Eyes* was analysed, however, by extending the search beyond the used keywords in this thesis, one can extract more data for their experiments.

Also, other sources for data, for example, *Google Fusion Tables* can be used [Google, 2014]. The accessibility of the CBR choices of visualisations to an audience with different levels of expertise in using visualisations for analysis. A formal evaluation of the visualisations using "Physics of Notation" guideline [Moody, 2009] and its operationalisation methods based on this guideline [Strrle and Fish, 2013] has been planned.

- *Feature Extraction.* The proposed classification of features can be explored further. For example, the computational complexity of features can be considered at more detailed level rather than considering only two groups of features, hard and easy features. Also, analysis of the features and their complexity should be done for more datasets from different domains.

- *Online and Offline (Batch) learning.* A DL approach showed higher accuracy but required significant time to train and test the algorithm. In future work, it is important to parallelise such algorithms and utilise multiple GPUs as well as CPUs for faster computations.

- *Deep Learning.* A DL approach, namely a DBN-based approach, showed higher accuracy but significantly worse time performance for training and testing the algorithm compared to four state-of-the-art algorithms such as LR, DT, *k*-NN, and SVM. Nevertheless, DBN is a highly parallelisable algorithm both at data and model levels, for example, across multiple GPUs as well as CPUs for faster computations. In future work, it is important to explore the performance of a parallelised version of this algorithm. Also, applying other DL algorithms using, for example, Recurrent Neural Networks (RNN), there is a possibility to improve the performance of prediction further.

- *Parallelisation.* Parallelising ML algorithms is one of the solutions to improve performance of prediction. For example, splitting data and computational tasks over multiple computers, CPUs, GPUs, and threads can significantly improve the performance of algorithms. Parallelising ML algorithms is an area of active research [Stahl and Bramer, 2013].

Appendix A

Datasets

This Appendix provides with the information on three datasets used in this thesis, namely *Stack Exchange*, *Twitter*, and *Facebook*. Firstly, the attributes of *Stack Exchange* websites data are introduced in Section A.1. Secondly, attributes of a *Twitter* dataset used in this thesis are presented in Section A.2. Finally, Section A.3 provides with attributes of *Facebook* data.

A.1 Stack Exchange

Attributes of *Stack Exchange* data are listed below:
Badges:

- UserId, e.g.: "420"
- Name, e.g.: "Teacher"
- Date, e.g.: "2008-09-15T08:55:03.923"

Comments:

- Id
- PostId
- Score
- Text, e.g.: "@Stu Thompson: Seems possible to me - why not try it?"
- CreationDate, e.g.:"2008-09-06T08:07:10.730"
- UserId

Posts:

- Id
- PostTypeId
  - 1: Question
  - 2: Answer
- ParentID (only present if PostTypeId is 2)
- AcceptedAnswerId (only present if PostTypeId is 1)
- CreationDate
- Score
- ViewCount
- Body
- OwnerUserId
- LastEditorUserId
- LastEditorDisplayName="Jeff Atwood"
- LastEditDate="2009-03-05T22:28:34.823"
- LastActivityDate="2009-03-11T12:51:01.480"
- CommunityOwnedDate="2009-03-11T12:51:01.480"
- ClosedDate="2009-03-11T12:51:01.480"
- Title=
- Tags=
- AnswerCount
- CommentCount
- FavoriteCount

Post History:

- Id

- PostHistoryTypeId
  - 1: Initial Title - The first title a question is asked with.
  - 2: Initial Body - The first raw body text a post is submitted with.
  - 3: Initial Tags - The first tags a question is asked with.
  - 4: Edit Title - A question's title has been changed.
  - 5: Edit Body - A post's body has been changed, the raw text is stored here as markdown.
  - 6: Edit Tags - A question's tags have been changed.
  - 7: Rollback Title - A question's title has reverted to a previous version.
  - 8: Rollback Body - A post's body has reverted to a previous version - the raw text is stored here.
  - 9: Rollback Tags - A question's tags have reverted to a previous version.
  - 10: Post Closed - A post was voted to be closed.
  - 11: Post Reopened - A post was voted to be reopened.
  - 12: Post Deleted - A post was voted to be removed.
  - 13: Post Undeleted - A post was voted to be restored.
  - 14: Post Locked - A post was locked by a moderator.
  - 15: Post Unlocked - A post was unlocked by a moderator.
  - 16: Community Owned - A post has become community owned.
  - 17: Post Migrated - A post was migrated.
  - 18: Question Merged - A question has had another, deleted question merged into itself.
  - 19: Question Protected - A question was protected by a moderator
  - 20: Question Unprotected - A question was unprotected by a moderator
  - 21: Post Disassociated - An admin removes the OwnerUserId from a post.
  - 22: Question Unmerged - A previously merged question has had its answers and votes restored.
- PostId
- RevisionGUID: At times more than one type of history record can be recorded by a single action. All of these will be grouped using the same RevisionGUID
- CreationDate: "2009-03-05T22:28:34.823"

- UserId
- UserDisplayName: populated if a user has been removed and no longer referenced by user Id
- Comment: This field will contain the comment made by the user who edited a post
- Text: A raw version of the new value for a given revision - If PostHistoryTypeId = 10, 11, 12, 13, 14, or 15 this column will contain a JSON encoded string with all users who have voted for the PostHistoryTypeId - If PostHistoryTypeId = 17 this column will contain migration details of either “from ¡url¿” or “to ¡url¿”
- CloseReasonId
  - 1: Exact Duplicate - This question covers exactly the same ground as earlier questions on this topic; its answers may be merged with another identical question.
  - 2: off-topic
  - 3: subjective
  - 4: not a real question
  - 7: too localized

Post Links:

- Id
- CreationDate
- PostId
- RelatedPostId
- PostLinkTypeId
  - 1: Linked
  - 3: Duplicate

Users:

- Id

- Reputation
- CreationDate
- DisplayName
- EmailHash
- LastAccessDate
- WebsiteUrl
- Location
- Age
- AboutMe
- Views
- UpVotes
- DownVotes

Votes:

- Id
- PostId
- VoteTypeId
  - ‘ 1‘: AcceptedByOriginator
  - ‘ 2‘: UpMod
  - ‘ 3‘: DownMod
  - ‘ 4‘: Offensive
  - ‘ 5‘: Favourite - if VoteTypeId = 5 UserId will be populated
  - ‘ 6‘: Close
  - ‘ 7‘: Reopen
  - ‘ 8‘: BountyStart
  - ‘ 9‘: BountyClose
  - ‘10‘: Deletion

  - ‘11‘: Undeletion
  - ‘12‘: Spam
  - ‘13‘: InformModerator

- CreationDate
- UserId (only for VoteTypeId 5)
- BountyAmount (only for VoteTypeId 9)

Quality of the Data The first step was to analyse the data and check it for quality. For this purpose, several queries were designed and executed to verify if there are missing values, errors, and outliers.

- *Missing values*: some questions and answers missed users' information so it was impossible to identify who posted some questions and answers. It might be related with the fact that some users deleted their profiles or administrators of the *Stack Exchange* decided to delete some users' profiles. The entries with missing values were excluded from the analysis. However, there were only a few missing entries compared to the volume of the data (less than 0.01% of missing entries).

- *Errors*: there were a few entries with $t(m_{ai}) < t(m_{qi})$ which is physically impossible since a question cannot be answered before it was asked. These erroneous entries were excluded from the analysis.

- *Outliers*: the data, where possible, was examined in order to find values which are out of range. For example, the initial analysis of the question response time showed that there were quite a lot (1%-3% of total number of questions) of questions answered in the extremely short time of a few seconds. Moreover, the answers were accepted as meaningful and answering the questions. It was noted that most of these questions were posted and answered by the same author. Further search on the Web showed that some users used questions with answers from other forums for posting at the *Stack Exchange* website. Possible motivation for these users could be to gain reputation in the *Stack Exchange* community. Also, another possible motivation was to contribute to *Stack Exchange* knowledge database. However, this user behaviour can strongly decrease a predicted response time if included in modelling temporal user behaviour. As a result, if a question was answered by the same author

Table A.1: Overall statistics for two *Stack Exchange* websites, Stack Overflow and Stack Maths, used in the experiments.

Forums	**Users**	**Questions**	**Answers**	**Comments**	**Questions with accepted Answers**
Stackoverflow	3,472,204	7,990,488	13,683,746	32,506,203	4,596,829
Math	138,856	323,334	479,708	1,506,639	178,765

in a very short time, the questions were excluded from the dataset for further analysis.

After an assessment of the quality of the data, the main descriptive statistics of the data were constructed and analysed. These statistics are presented below.

Data Statistics The statistics for the two datasets used in this thesis, Stack Overflow and Stack Maths, are presented in Table A.1. *Stack Overflow* was introduced in 2008 and is the oldest and the largest website at *Stack Exchange*. However, the websites introduced in 2010 and 2011 have attracted large number of users and questions as well, for example Stack Maths website (see Table A.1).

Overall statistics for question response time are shown in Table A.2. The mean answering time for the eleven largest Q&A forums varies from less than 3 days to almost 15 days. The median answering time varies from 20 minutes to 133 minutes. Statistics on the ratio of questions answered faster than in an hour, a day, and a month varies significantly among different forums as well. However, more than 90% of all questions are answered within 1 month. Nevertheless, even 10% of unanswered within a month questions sums up to thousands of questions.

A.2 Twitter

A Twitter dataset from [De Domenico et al., 2013] where the authors extracted tweets by hashtags *lhc*, *cern*, *boson*, *higgs* was used in this thesis. The tweets were filtered by date so only the tweets from $00:00$ AM 1^{st} July 2012 to $11:59$ PM 7^{th} July 2012 were considered. As a result $985,692$ tweets were extracted. Each tweet has a timestamp, unique id of a user who posted it, information on the tweet such as whether the tweet is a reply (RE), retweet (RT), or a mention(MT). These tweets form an interaction graph where each node corresponds to a user and directed edges

Table A.2: Statistics on question response time for Stack Overflow (SO) and Maths.

Forums	**Questions**		**Question Response Time**						
	A*	**SA****	**%, time statistics**				**% answered within**		
			mean, d	**med, m**	**min, m**	**max, days**	**1 hour**	**1 day**	**1 month**
SO	4,133,896	.11	5.7	20	0.20	2,087	67	90	97
Math	174,685	.26	2.8	27	0.37	1,314	66	92	98

* Answered Questions; the difference in the number of answered questions in this table and Table A.1 is due to the fact that the questions answered by the same author were excluded.
** Self Answered Questions: a ratio of questions answered by the person who submitted the question.

represent the flow of tweets from that user. Each edge has two attributes, namely the time a tweet was posted and the type of a tweet - RE, RT, or MT.

Quality of the Data Since the dataset was used by [De Domenico et al., 2013], it was already cleaned and no outliers, missing or corrupted entries were identified.

Data Statistics The *Twitter* statistics is represented below. First, this dataset has $985,692$ tweets and $245,234$ users who produced these tweets. Second, the these tweets cover the tweets from $00:00$ AM 1^{st} July 2012 to $11:59$ PM 7^{th} July 2012.

A.3 Facebook

The attributes of Facebook data are listed below:
User attributes A_u:

- Registration Date: The date the user joined Facebook.
- Account Status History: The dates when a user's account was reactivated, deactivated, disabled or deleted. This information include date, time, device, IP address, machine cookie and browser information.
- About Me: Information the user added to the About section of user's Timeline like relationships, work, education, where the user lives. It includes any updates or changes the user made in the past and what is currently in the About section.
- Favourite Quotes: Information the user added to the Favourite Quotes section of the About section of user's Timeline.

- Hometown: The place the user added to hometown in the About section of their Timeline.
- Current City: The city the user added to the About section of their Timeline.
- Address: User's current address or any past addresses the user had on their account.
- Name: The name used in the user's Facebook account.
- Alternate Name: Any alternate names the user has on their account (ex: a maiden name or a nickname).
- Screen Names: The screen names the user added to their account, and the service they are associated with.
- Name Changes: Any changes the user made to the original name the user used when he/she signed up for Facebook.
- Date of Birth: The date the user added to Birthday in the About section.
- Birthday Visibility: How user's birthday appears on user's Timeline.
- Spoken Languages: The languages the user added to Spoken Languages in the About section.
- Work: Any current information the user added to Work in the About section.
- Vanity URL: User's Facebook URL (ex: username or vanity for user's account).
- Education: Any information the user added to Education field in the About section.
- Emails: Email addresses added to user's account.
- Linked Accounts: A list of the accounts the user linked to user's Facebook account.
- Facial Recognition Data: A unique number based on a comparison of the photos the user was tagged in.
- Gender: The gender that the user added to the About section.
- Phone Numbers: Mobile phone numbers the user added to his/her account, including verified mobile numbers the user added for security purposes.
- Political Views: Any information the user added to Political Views in the About section.
- Religious Views: The current information the user added to Religious Views in the About section.
- Status Updates: Any status updates the user posted.
- Privacy Settings: Your privacy settings.
- Notification Settings: A list of all user's notification preferences and whether he/her has email and text enabled or disabled for each.
- Locale: The language the user selected to use Facebook in.
- Pages You Admin: A list of pages the user admins.

- Physical Tokens: Badges the user added to his/her account.
- Networks: Networks (affiliations with schools or workplaces) that the user belongs to on Facebook.
- Currency: User's preferred currency on Facebook.
- Credit Cards: If user makes purchases on Facebook (ex: in apps) and has given Facebook his/her credit card number.
- IP Addresses: A list of IP addresses where the user logged into user's Facebook account.
- Logins: IP address, date and time associated with logins to Facebook account of the user.
- Logouts: IP address, date and time associated with logouts from Facebook account of the user.
- Last Location: The last location associated with an update.
- Check-ins: The places the user checked into.
- Recent Activities: Actions the user has taken and interactions the user recently has had.
- Searches: Searches the user made on Facebook.
- Apps: All of the apps the user has added.
- Ads Clicked: Dates, times and titles of ads clicked by the user.
- Ad Topics: A list of topics that the user may be targeted against based on his/her stated likes, interests and other data the user put in his/her Timeline.
- Likes on Other Sites: Likes the user made on sites.

Relationship attributes A_r:

- Connections: The people who have liked a Page or Place of the user, RSVPed to his/her event, installed his/her app or checked in to his/her advertised place within 24 hours of viewing or clicking on an ad or Sponsored Story.
- Family Friends: the people whom the user has indicated as his/her family members.
- Followers: A list of people who follow the user.
- Following: A list of people the user follows.
- Friends: A list of friends.
- Friend Requests: Pending sent and received friend requests.
- Removed Friends: People the user removed as friends.
- Pending Friend Requests: Pending sent and received friend requests.
- Hidden from News Feed: Any friends, apps or pages the user hidden from News Feed.

Message attributes A_m:

- Chat: A history of the conversations the user had on Facebook Chat.
- Messages: Messages the user sent and received on Facebook.
- Posts by You: Anything you posted to user's own Timeline, like photos, videos and status updates.
- Posts by Others: Anything posted to the Timeline by someone else, like wall posts or links shared on user's Timeline by friends.
- Posts to Others: Anything the user posted to someone elses Timeline, like photos, videos and status updates.
- Likes on Others' Posts: Posts, photos or other content the user liked.
- Likes on Your Posts from others: Likes on own posts, photos or other content.
- Pokes: A list of whos poked you and who the user poked.
- Shares: Content (ex: a news article) the user shared with others on Facebook using the Share button or link.
- Notes: Any notes the user has written and published to user's account.
- Photos: Photos the user uploaded to user's account.
- Photos Metadata: Any metadata that is transmitted with uploaded photos.
- Videos: Videos the user posted to the Timeline.
- Events: Events the user has joined or been invited to.
- Groups: A list of groups the user belong to on Facebook.

Quality of the Data The data has missing values which were eliminated (less than 5%).

Data Statistics The *Facebook* statistics is represented below. Firstly, the data represents only one user registered in June 2009 and who had regularly used his Facebook account. As a result, there are $1,212$ conversations. In total, the user sent $19,549$ messages and received $24,352$ messages. The user had 440 friends, all the friendships are timestamped.

Appendix B

Feature Complexity for Datasets

This Appendix provides information on features and their complexity across three different datasets used in this thesis, namely Stack Exchange, Twitter, and Facebook. These features are presented below in Sections B.1, B.2, and B.3 respectively.

B.1 Stack Exchange

This subsection introduces semantic groups of features and their calculated complexities for Stack Exchange datasets. In this thesis, Stack Overflow and Stack Maths datasets were used. User features are presented in Table B.1, message features are in Table B.2, temporal features are in Table B.3, and structural features are in Table B.4.

Table B.1: User Features.

F_n	**Description**	**Comments**	**Complexity**
F_1	User Id	A unique number associated with every user	Raw
F_2	User Info	The location of a user	Raw
F_3	User Info	User's latitude	O(1)
F_4	User Info	User's longitude	O(1)
F_5	User Info	User's time zone	O(1)
F_6	User Info	The reputation of the user	Raw
F_7	User Info	The total number of user's profile views by other users	Raw
F_8	User Info	The number of times a user was up-voted by other users	Raw
F_9	User Info	The number of times a user was down-voted by other users	Raw

Table B.2: Message Features.

F_n	**Description**	**Comments**	**Complexity**
F_{10}	Source User	The ID of a user who asked the question	Raw
F_{11}	Destination User	The ID of a user who answered the questions	Raw
F_{12}	Type	The type of the message (question)	Raw
F_{13}	Text length	The length of the question title in chars	O(1)
F_{14}	Text length	The length of the question body	O(1)
F_{15}	Occurrence of a word	The number of url links in the body of a question	O(1)
F_{16}	Occurrence of a word	The number of images in the body of a question	O(1)
F_{17}	Occurrence of a set of words	The number of 'wh' words in a question title	O(1)
F_{18}	Occurrence of a set of words	The number of 'wh' words in a question body	O(1)
F_{19}	Occurrence of a set of words	The number of active verbs in a question title	O(1)
F_{20}	Occurrence of a set of words	The number of active verbs in a question body	O(1)
F_{21}	Occurrence of a set of words	The number of times a user mentioned himself or themselves, for example, 'we', 'I', 'me'	O(1)
F_{22}	Occurrence of a set of words	The number of tags a question is tagged with	O(1)
F_{23}	The most frequent set of words	The popularity of the tags a question is tagged with	$kO(n\log(n))$ where n is the total number of questions and k is the number of tags k
F_{24}	The most frequent set of words	The number of popular tags a question is tagged with	The same as for F_{23}
F_{25}	# of message views	The total number of times a question was viewed	Raw
F_{26}	# of messages sent by a user	The total number of question asked by a user	O(1)
F_{27}	# of messages received	The total number of answers given by a user	O(1)
F_{28}	# of messages answered	The total number of answers which were accepted by other users	O(1)

Table B.3: Temporal Features.

F_n	**Description**	**Comments**	**Complexity**
F_{29}	Message timestamp	The day when a question was asked	O(1)
F_{30}	Message timestamp	The hour when a question was asked	O(1)
F_{31}	Message timestamp	The minute when a question was asked	O(1)
F_{32}	Message timestamp	The second when a question was asked	O(1)
F_{33}	Time passed since a user u registered	The number of days a user has been registered	O(1)
F_{34}	# of messages by a user during the period T	The number of questions asked by a user during the last week	$O(1)$ for chronologically ordered questions
F_{35}	# of messages by a user during the period T	The number of answers given by a user in the last week	$O(1)$ for chronologically ordered answers
F_{36}	# of messages by a user during the period T	The number of comments and replies for a question during last month	$O(k)$ where k is the number of comments and replies during the last day
F_{37}	The average messaging activity of a user	The average answering time for user's questions: $\frac{1}{n}\sum_{j=1}^{n} t(a_j) - t(q_j)$, where $t(a_j)$ the time when an answer a_j was given, $t(q_j)$ is the time when the question q_j was asked, n is the total number of questions	$O(n)$ where n is the number of all answered questions by a user
F_{38}	The frequency of temporal activity by a time period T	A vector with twelve entries where each entry represents the number of posts for each month in last year: $(p_1(u_i), ..., p_{12}(u_i))$ where p_i is the number of posts by a user u_i in a particular month j	$O(k)$ where k is the total number of posts
F_{39}	The frequency of temporal activity by a time period T	A vector with twenty four entries where each entry represents the average number of posts for each hour: $(p_1(u_i), ..., p_{24}(u_i))$, where p_j is the average number of posts posted by a user u_i at a particular hour j	$O(k)$ where k is the total number of posts

Table B.4: Structural Features.

F_n	**Description**	**Comments**	**Complexity**
F_{40}	Inbetweeness centrality $C_D(v_i)$	The number of users who follow the user u	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{41}	Pagerank	Pagerank of a node associated with a user u	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.

B.2 Twitter

This subsection introduces semantic groups of features and their calculated complexities for a Twitter dataset. User features are presented in Table B.5, message features are in Table B.6, relationship features are in Table B.7, temporal features are in Table B.8, and structural features are in Table B.9.

Table B.5: User Features.

F_n	**Description**	**Comments**	**References**
F_1	User Id	A unique number associated with a user	Raw

Table B.6: Message Features.

F_n	**Description**	**Comments**	**Complexity**
F_2	Source User	The ID of the source user	Raw
F_3	Destination User	The ID of the destination user	Raw
F_4	Message Type	Tweet, retweet, reply, or mention	Raw
F_5	# of messages sent	The number of tweets sent	O(1)
F_6	# of messages sent	The number of tweets retweeted by a user	O(1)
F_7	# of messages sent	The number of tweets mentioned by a user	O(1)
F_8	# of messages received	The number of tweets received by a user	O(1)
F_9	# of messages answered	The number of tweets retweeted by others	O(1)
F_{10}	# of messages answered	The number of tweets mentioned by others	O(1)

Table B.7: Relationship Features.

F_n	**Description**	**Comments**	**References**
F_{11}	Source User	The ID of the source user	Raw
F_{12}	Destination User	The ID of the destination user	Raw
F_{13}	Type	The type of the relationship: following	Raw
F_{14}	# of incoming relationships	The number of followers for a user u	$O(\|E\|)$ where $\|E\|$ is the number of incoming edges
F_{15}	# of outcoming relationships	The number of following users for a user u	$O(\|E\|)$ where $\|E\|$ is the number of outcoming edges

Table B.8: Temporal Features.

F_n	**Description**	**Comments**	**References**
F_{16}	Message timestamp	The day a tweet was posted	O(1)
F_{17}	Message timestamp	The hour a tweet was posted	O(1)
F_{18}	Message timestamp	The minutes a tweet was posted	O(1)
F_{19}	Message timestamp	The seconds a tweet was posted	O(1)
F_{20}	# of messages by a user during the period T	The number of tweets retweeted the last day	O(1)
F_{21}	# of messages by a user during the period T	The number of tweets replied last day	O(1)
F_{22}	# of messages by a user during the period T	The number of tweets mentioned last day	O(1)
F_{23}	The average user's messaging activity	The average retweeting time for a tweet	$O(n)$ where n is the number of all answered questions
F_{24}	The frequency of temporal activity by a time period T	A vector with 24 entries where each entry represents the number of tweets for each hour in last three days: $(p_1(u_i), ..., p_{24}(u_i))$ where p_i is # of posts by a user u_i in a j^{th} month	$O(k)$ where k is the total number of posts

Table B.9: Structural Features.

F_n	**Description**	**Comments**	**References**
F_{25}	Inbetweeness centrality	Inbetweeness centrality for the relationship (social) network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{26}	Inbetweeness centrality	Inbetweeness centrality for the interaction (tweets) network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{27}	Pagerank	Pagerank for the relationship (social) network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{28}	Pagerank	Pagerank for the interaction (tweets) network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.

B.3 Facebook

In this Subsection, semantic groups of features with their computational complexities are presented. User features are presented in Table B.10, message features are in Table B.11, relationship features are in Table B.12, temporal features are in Table B.13, and structural features are in Table B.14.

Table B.10: User Features.

F_n	**Description**	**Comments**	**References**
F_1	User Id	A unique number for a user	O(1)
F_2	User Info	User name	O(1)
F_3	User Info	User gender	O(1)
F_4	User Info	The location of a user	O(1)
F_5	User Info	User's latitude	O(1)
F_6	User Info	User's longitude	O(1)
F_7	User Info	User's time zone	O(1)

Table B.11: Message Features.

F_n	**Description**	**Comments**	**References**
F_8	Source User	The ID of the source user	Raw
F_9	Destination User	The ID of the destination user	Raw
F_{10}	Message Type	The type of a message: chat	Raw
F_{11}	Text length	The length of a message in chars	O(1)
F_{12}	Occurrence of a word	The number of url links in the body of a question	O(1)
F_{13}	Occurrence of a word	The number of images in the body of a question	O(1)
F_{14}	Occurrence of a set of words	The number of punctuation marks in a text	O(1)
F_{15}	The most frequent set of words	The number of popular words in a message	$kO(n\log(n))$ where n is the total number of questions and k is the number of tags k
F_{16}	The number of messages sent by a user	The total number of messages sent by a user	O(1)
F_{17}	The number of messages received by a user	The total number of messages received by a user	O(1)

Table B.12: Relationship Features.

F_n	**Description**	**Comments**	**References**
F_{18}	Source User	The ID of the source user	Raw
F_{19}	Destination User	The ID of the destination user	Raw
F_{20}	Relationship Type	The type of a relationship: friendship	Raw
F_{21}	The number of friends	degree of a relationship graph nodes	O(1)

Table B.13: Temporal Features.

F_n	**Description**	**Comments**	**References**
F_{22}	Message timestamp	The day of a message	O(1)
F_{23}	Message timestamp	The hour of a message	O(1)
F_{24}	Message timestamp	The minute of the message	O(1)
F_{25}	Message timestamp	The second of the message	O(1)
F_{26}	The number of messages by a user during the period T	The number of messages sent the last day	O(1)
F_{27}	The number of messages by a user during the period T	The number of messages received the last day	O(1)
F_{28}	The number of messages by a user during the period T	The number of messages received the last month	O(1)

Table B.14: Structural Features.

F_n	**Description**	**Comments**	**References**
F_{29}	Inbetweeness centrality	Inbetweeness centrality for the relationship network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{30}	Pagerank	Pagerank for the relationship network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{31}	Inbetweeness centrality	Inbetweeness centrality for the interaction network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.
F_{32}	Pagerank	Pagerank for the interaction network	$\Theta(V^2)$ for a graph represented as a dense *adjacency matrix* or $\Theta(E)$ in a sparse matrix representation.

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI COMMUNICATIONS*, 7(1):39–59.

Adedoyin-Olowe, M., Gaber, M. M., and Stahl, F. T. (2013). A survey of data mining techniques for social media analysis. *Computing Research Repository (CoRR)*, abs/1312.4617:1–25.

Ahn, J.-W., Plaisant, C., and Shneiderman, B. (2014). A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):365–376.

Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2012). Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA. ACM.

Aneetha, A. S., Indhu, T. S., and Bose, S. (2012). Hybrid network intrusion detection system using expert rule based approach. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, pages 47–51, New York, NY, USA. ACM.

Angus, D., Watson, B., Smith, A., Gallois, C., and Wiles, J. (2012). Visualising conversation structure across time: Insights into effective doctor-patient consultations. *PLoS ONE*, 7(6):1–13.

Artzi, Y., Pantel, P., and Gamon, M. (2012). Predicting responses to microblog posts. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 602–606, Stroudsburg, PA, USA. Association for Computational Linguistics.

Asaduzzaman, M., Mashiyat, A. S., Roy, C. K., and Schneider, K. A. (2013). Answering questions about unanswered questions of Stack Overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 97–100, Piscataway, NJ, USA. IEEE Press.

Badea, L. M. (2014). Predicting consumer behavior with artificial neural networks. *Procedia Economics and Finance*, 15:238 – 246. Emerging Markets Queries in Finance and Business (EMQ 2013).

Bhat, V., Gokhale, A., Jadhav, R., Pudipeddi, J., and Akoglu, L. (2014). Min(e)d your tags: Analysis of question response time in Stack Overflow. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 328–335, Beijing, China.

Bianchini, M. and Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(8):1553–1565.

Bifet, A., de Francisci Morales, G., Read, J., Holmes, G., and Pfahringer, B. (2015). Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15*, pages 59–68, New York, NY, USA. ACM.

Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2012). A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3):483–519.

Bottou, L. (2010). *Proceedings of 19th International Conference on Computational Statistics (COMPSTAT'2010), Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, chapter Large-Scale Machine Learning with Stochastic Gradient Descent, pages 177–186. Physica-Verlag HD, Heidelberg.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177.

Bronson, N., Amsden, Z., Cabrera, G., Chakka, P., Dimov, P., Ding, H., Ferris, J., Giardullo, A., Kulkarni, S., Li, H., Marchukov, M., Petrov, D., Puzar, L., Song, Y. J., and Venkataramani, V. (2013). Tao: Facebook's distributed data store for the social graph. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference*, USENIX ATC'13, pages 49–60, Berkeley, CA, USA. USENIX Association.

Burlutskiy, N., Petridis, M., Fish, A., and Ali, N. (2014a). Enabling the visualization for reasoning about temporal data. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 179–180, Australia, Melbourne. IEEE.

Burlutskiy, N., Petridis, M., Fish, A., and Ali, N. (2014b). How to visualise a conversation: case-based reasoning approach. *19th UK Workshop on Case-Based Reasoning (UKCBR 2014)*, pages 1–12.

Burlutskiy, N., Petridis, M., Fish, A., and Ali, N. (2015). Prediction of users' response time in q&a communities. In *ICMLA'15, International Conference on Machine Learning and Applications*, pages 618–623, Miami, FL, USA.

Burlutskiy, N., Petridis, M., Fish, A., Ali, N., and Chernov, A. (2016). An investigation on online versus batch learning in predicting user behaviour. In *Thirty-sixth SGAI International Conference on Artificial Intelligence (AI-2016)*, pages 10–25, England, Cambridge. Springer.

Cai, Y. and Chakravarthy, S. (2013). Answer quality prediction in Q&A social networks by leveraging temporal features. *International Journal of Next-Generation Computing (IJNGC)*, 4(1):1–27.

Campbell, W. M., Dagli, C. K., and Weinstein, C. J. (2013). Social Network Analysis with Content and Graphs. *LINCOLN LABORATORY JOURNAL*, 20(1):1–20.

Casteigts, A., Flocchini, P., Quattrociocchi, W., and Santoro, N. (2010). Time-varying graphs and dynamic networks. *Computing Research Repository (CoRR)*, abs/1012.0009:1–20.

Cattuto, C., Quaggiotto, M., Panisson, A., and Averbuch, A. (2013). Time-varying social networks in a graph database: A neo4j use case. In *First International Workshop on Graph Data Management Experiences and Systems*, GRADES '13, pages 1–6, New York, NY, USA. ACM.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Computing Research Repository (CoRR)*, abs/1603.02754:1–13.

Cheng, J., Romero, D. M., Meeder, B., and Kleinberg, J. M. (2011). Predicting reciprocity in social networks. In *IEEE Third International Conference on Social Computing (SocialCom)*, pages 49–56, Boston, MA, USA.

Choi, S., Kim, E., and Oh, S. (2013). Human behavior prediction for smart homes using deep learning. In *2013 IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 173–179, Gyeongju, South Korea.

Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA. IEEE Computer Society.

Comarela, G., Crovella, M., Almeida, V., and Benevenuto, F. (2012). Understanding factors that affect response rates in Twitter. In *Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, HT '12, pages 123–132, New York, NY, USA. ACM.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585.

De Domenico, M., Lima, A., Mougel, P., and Musolesi, M. (2013). The anatomy of a scientific rumor. *Scientific Reports*, 3(02980):1–11.

De la Rosa, J. L., Mollet, R., Montaner, M., Ruiz, D., and Muñoz, V. (2007). Kalman filters to generate customer behavior alarms. In *Artificial Intelligence Research and Development, Proceedings of the 10th International Conference of the ACIA, CCIA 2007, October 25-26, 2007, Sant Julià de Lòria, Andorra*, pages 416–425.

Dekel, O. (2009). From online to batch learning with cutoff-averaging. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 377–384. Curran Associates, Inc.

Ding, Y., Yan, S., Zhang, Y., Dai, W., and Dong, L. (2016). Predicting the attributes of social network users using a graph-based machine learning method. *Computer Communications*, 73, Part A:3 – 11.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

Donath, J., Dragulescu, A., Zinman, A., Viégas, F., and Xiong, R. (2010). Data portraits. In *ACM Special Interest Group on Computer GRAPHics and Interactive Techniques 2010 Art Gallery*, SIGGRAPH '10, pages 375–383, New York, NY, USA. ACM.

Dror, G., Maarek, Y., and Szpektor, I. (2013). Will my question be answered? predicting "question answerability" in community question-answering sites. In Blockeel, H., Kersting, K., Nijssen, S., and Zelezny, F., editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8190 of *Lecture Notes in Computer Science*, pages 499–514. Springer Berlin Heidelberg.

Facebook (2016). Facebook data model. Retrieved on 2016-05-18 from https://www.facebook.com/help/405183566203254.

Farrugia, M., Hurley, N., and Quigley, A. (2011). Exploring temporal ego networks using small multiples and tree-ring layouts. In *4th International Conference on Advances in Human Computer Interfaces (ACHI)*, pages 79–88, Guadeloupe, France.

Frau, S., Roberts, J. C., and Boukhelifa, N. (2005). Dynamic coordinated email visualization. In Skala, V., editor, *WSCG05 - 13th International Conference on Computer Graphics, Visualization and Computer Vision'2005*, pages 182–196, Plzen, Czech Republic. (Jan 31 - Feb 4).

Freyne, J. and Smyth, B. (2010). Creating visualizations: A case-based reasoning perspective. In Coyle, L. and Freyne, J., editors, *Artificial Intelligence and Cognitive Science*, volume 6206 of *Lecture Notes in Computer Science*, pages 82–91. Springer Berlin Heidelberg.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Garriss, S., Kaminsky, M., Freedman, M. J., Karp, B., Mazières, D., and Yu, H. (2006). Re: Reliable email. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 22–22, Berkeley, CA, USA. USENIX Association.

Gilchrist, W. (1984). *Statistical Modelling.* John Wiley and Sons, Chichester, UK. pages 34–36.

Glady, N., Baesens, B., and Croux, C. (2009). Modeling churn using customer lifetime value. *European Journal of Operational Research*, 197(1):402–411.

Goel, D. and Batra, D. (2009). Predicting user preference for movies using netflix database. *Department of Electrical and Computer Engineering, Carniege Mellon University*, pages 1–7.

Google (2014). Fusion tables. Retrieved on 2014-03-25 from http://tables.googlelabs.com/.

Gorodetsky, V., Samoylov, V., Liu, H., Motoda, H., Setiono, R., and Zhao, Z. (2010). Feature Extraction for Machine Learning: Logic Probabilistic Approach. In *Proceedings of the 4th Workshop on Feature Selection in Data Mining*, pages 55–65.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., and Wang, G. (2015). Recent advances in convolutional neural networks. *Computing Research Repository (CoRR)*, abs/1512.07108:1–37.

Guille, A. and Hacid, H. (2012). A predictive model for the temporal dynamics of information diffusion in online social networks. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 1145–1152, New York, NY, USA. ACM.

Gummadi, K. P., Mislove, A., and Druschel, P. (2006). Exploiting social networks for internet search. In *Proceedings of the 5th Workshop on Hot Topics in Networks*, pages 79–84, Irvine, CA.

Guyon, I. and Elisseeff, A. (2006). *Feature Extraction: Foundations and Applications*, chapter An Introduction to Feature Extraction, pages 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hadlak, S., Schumann, H., Cap, C. H., and Wollenberg, T. (2013). Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2267–2276.

Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. (2014). Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. *CoRR*, abs/1409.0919.

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., and Candela, J. Q. n. (2014). Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, pages 1–9, New York, NY, USA. ACM.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hira, Z. M. and Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in Bioinformatics*, 2015:1–13.

Hoi, S. C., Wang, J., and Zhao, P. (2014). Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, 15:495–499.

Horowitz, D. and Kamvar, S. D. (2010). The anatomy of a large-scale social search engine. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 431–440.

Hu, H., Liu, B., Wang, B., Liu, M., and Wang, X. (2013). Exploring social features for answer quality prediction in CQA portals. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on*, volume 04, pages 1904–1909, Tianjin, China.

IBM (2014). Many eyes. Retrieved on 2014-03-25 from http://www.manyeyes.com.

Jernite, Y., Halpern, Y., Horng, S., and Sontag, D. (2013). Predicting chief complaints at triage time in the emergency department. *NIPS Workshop on Machine Learning for Clinical Data Analysis and Healthcare*, pages 1–5.

John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the 11th International Conference*, pages 121–129. Morgan Kaufmann.

Jovicic, S. (2000). Role of memory in email management. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 151–152, New York, NY, USA. ACM.

Karnstedt, M., Hennessy, T., Chan, J., and Hayes, C. (2010). Churn in social networks: A discussion boards case study. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 233–240.

Kaur, P., Owonibi, M., and Knig-Ries, B. (2015). Towards visualization recommendation - a semi- automated domain-specific learning approach. In *Proceedings of the 27th GI-Workshop Grundlagen von Datenbanken*, pages 30–35.

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8.

Kim, M. and Leskovec, J. (2011). Modeling social networks with node attributes using the multiplicative attribute graph model. *Computing Research Repository (CoRR)*, abs/1106.5053:1–15.

Knuth, D. E. (1976). Big omicron and big omega and big theta. *Special Interest Group on Algorithms and Computation Theory (SIGACT) News*, 8(2):18–24.

Kojadinovic, I. and Wottka, T. (2000). Comparison between a filter and a wrapper approach to variable subset selection in regression problems. In *Proceedings of the European Symposium on Intelligent Techniques*, pages 311–321.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands. IOS Press.

Larivière, B. and Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert System Applications*, 29(2):472–484.

Lattanzi, S. (2010). *Algorithms and models for social networks.* PhD thesis, Universita di Roma.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *Computing Research Repository (CoRR)*, abs/1405.4053:1–9.

Lee, J., Lin, S., and Karahalios, K. (2013). Visualizing patterns of social and communicative behavior in children using plexlines. In *Visual Analytics in Healthcare (VAHC) 2013*, pages 1–10, Washington, DC, USA.

Lee, K., Mahmud, J., Chen, J., Zhou, M. X., and Nichols, J. (2014). Who will retweet this? automatically identifying and engaging strangers on Twitter to spread information. *Computing Research Repository (CoRR)*, abs/1405.3750:1–10.

Lezina, C. G. E. and Kuznetsov, A. M. (2012). Predict closed questions on Stack Overflow. Report on Kaggle Competition, pages 1-5.

Li, B., Lyu, M., and King, I. (2012). Communities of Yahoo! Answers and Baidu Zhidao: Complementing or competing? In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Brisbane, Australia.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *Neural Networks, IEEE Transactions on*, 17(6):1411–1423.

Lim, T.-S., Loh, W.-Y., and Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228.

Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *Computing Research Repository (CoRR)*, abs/1506.00019:1–38.

Loumiotis, I., Adamopoulou, E., Demestichas, K., and Theologou, M. (2014). On trade-off between computational efficiency and prediction accuracy in bandwidth traffic estimation. *Electronics Letters*, 50(10):754–756.

Ma, H., Zhou, T. C., Lyu, M. R., and King, I. (2011). Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems*, 29(2):1–23.

Mahmud, J., Chen, J., and Nichols, J. (2013). When will you answer this? estimating response time in Twitter. In *International AAAI Conference on Weblogs and Social Media*, pages 697–700.

Markov, A., Last, M., and Kandel, A. (2008). The hybrid representation model for web document classification. *International Journal on Intelligent Systems*, 23:654–679.

McDuff, D., Kaliouby, R. E., Cohn, J. F., and Picard, R. W. (2015). Predicting ad liking and purchase intent: Large-scale analysis of facial responses to ads. *IEEE Transactions on Affective Computing*, 6(3):223–235.

Minka, T. P. (2003). A comparison of numerical optimizers for logistic regression. Technical report. pages 1–18.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

Moody, D. L. (2009). The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779.

Morris, M. R., Teevan, J., and Panovich, K. (2010). What do people ask their social networks, and why? A survey study of status message Q&A behavior. In

Proceedings of Conference on Human Factors in Computer Systems (CHI) 2010, pages 1–10. ACM.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective.* Cambridge, MA.

Nazerfard, E. and Cook, D. (2013). Using bayesian networks for daily activity prediction. In *AAAI Plan, Activity, and Intent Recognition Workshop*, pages 32–38, Washington, DC, USA.

Nohuddin, P. N. E., Coenen, F., Christley, R., and Sunayama, W. (2015). Visualisation of trend pattern migrations in social networks. In *Advances in Visual Informatics - 4th International Visual Informatics Conference, IVIC 2015, Bangi, Malaysia, November 17-19, 2015, Proceedings*, pages 77–88.

Oza, N. C. and Russell, S. (2001). Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 359–364, New York, NY, USA. ACM.

Page, A., Turner, J., Mohsenin, T., and Oates, T. (2014). Comparing raw data and feature extraction for seizure detection with deep learning methods. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 284–287, Pensacola Beach, FL, USA.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report. Stanford InfoLab, pages 1-17.

Park, C. and Seo, J. (2013). Consideration of purchase dependence in inventory management. *Computers & Industrial Engineering*, 66(2):274 – 285.

Pentland, A. and Liu, A. (1999). Modeling and prediction of human behavior. *Neural Computation*, 11(1):229–242.

Pentreath, N. (2015). *Machine Learning with Spark.* Packt Publishing Ltd, UK, London.

Petridis, M., Kapetanakis, S., Ma, J., and Burlutskiy, N. (2014). Temporal knowledge representation for case-based reasoning based on a formal theory of time. In *Workshop on Reasoning about Time in CBR, RATIC 2014*, pages 1–10, Cork, Ireland.

Pfaltz, J. L. (2013). A mathematical model of dynamic social networks. *Social Network Analysis and Mining*, 3(4):863–872.

Pfeiffer, III, J. J., Moreno, S., La Fond, T., Neville, J., and Gallagher, B. (2014). Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 831–842, New York, NY, USA. ACM.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Radinsky, K., Svore, K., Dumais, S., Teevan, J., Bocharov, A., and Horvitz, E. (2012). Modeling and predicting behavioral dynamics on the web. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 599–608, New York, NY, USA. ACM.

Raikwal, J. S., Singhai, R., and Saxena, K. (2013). Article: Integrating markov model with knn classification for web page prediction. *International Journal of Computer Applications*, 61(22):11–15.

Read, J., Bifet, A., Pfahringer, B., and Holmes, G. (2012). *Advances in Intelligent Data Analysis XI: 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings*, chapter Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data, pages 313–323. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ruangkanokmas, P., Achalakul, T., and Akkarajitsakul, K. (2016). Deep belief networks with feature selection for sentiment classification. In *7th International Conference on Intelligent Systems, Modelling and Simulation*, pages 9–14, Bangkok, Thailand.

Sadilek, A. and Krumm, J. (2012). Far out: Predicting long-term human mobility. In *2012 AAAI Conference on Artificial Intelligence*, pages 1–7, Toronto, Ontario, Canada.

Samiei, M., Dill, J., and Kirkpatrick, A. (2004). EzMail: using information visualization techniques to help manage email. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*, pages 477–482, London, England.

Santoro, N., Quattrociocchi, W., Flocchini, P., Casteigts, A., and Amblard, F. (2011). Time-varying graphs and social network analysis: Temporal indicators and metrics. *Computing Research Repository (CoRR)*, abs/1102.0629:1–6.

Sathiyanarayanan, M. and Burlutskiy, N. (2015a). Design and evaluation of euler diagram and treemap for social network visualisation. In *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–6, Bangalore, India. IEEE.

Sathiyanarayanan, M. and Burlutskiy, N. (2015b). Visualizing social networks using a treemap overlaid with a graph. *Procedia Computer Science*, 58:113–120.

Scellato, S., Noulas, A., and Mascolo, C. (2011). Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1046–1054, New York, NY, USA. ACM.

Schenker, A., Bunke, H., Last, M., and Kandel, A. (2005). *Graph-Theoretic Techniques for Web Content Mining*. World Scientific.

Schramm, W. and Roberts, D. F. (1971). *The Process and effects of mass communication / edited by Wilbur Schramm and Donald F. Roberts*. University of Illinois Press Urbana, rev. ed. edition.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2503–2511. Curran Associates, Inc.

Semertzidis, K. and Pitoura, E. (2016). Time traveling in graphs using a graph database. In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference*, pages 1–6, Bordeaux, France.

Shannon, C. E. (1948). A mathematical theory of communication. 27(4):623–656.

Sharma, A. and Panigrahi, P. K. (2013). A neural network based approach for predicting customer churn in cellular network services. *Computing Research Repository (CoRR)*, abs/1309.3945:26–31.

SkyTree (2014). Managing online fraud with skytree advanced analytics. Retrieved on 2015-01-12 from http://skytree.net/wp-content/uploads/2014/10/SkytreeFinSvcSolnPaper.pdf.

Smith, M. A. and Fiore, A. T. (2001). Visualization components for persistent conversations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 136–143, New York, NY, USA. ACM.

Sontag, E. D. (1998). *Mathematical Control Theory: Deterministic Finite Dimensional Systems (2Nd Ed.).* Springer-Verlag New York, Inc., New York, NY, USA.

Spiro, E., Irvine, C., DuBois, C., and Butts, C. (2012). Waiting for a retweet: modeling waiting times in information propagation. In *2012 Neural Information Processing Systems (NIPS) workshop of social networks and social media conference*, volume 12, pages 1–8, Montreal, Canada.

StackExchange (2014). Stack Exchange dump data, September 2014. Accessed on 2015-03-16 from https://archive.org/details/stackexchange.

Stahl, F. and Bramer, M. (2013). Scaling up classification rule induction through parallel processing. *The Knowledge Engineering Review*, 28:451–478.

Steurer, M. and Trattner, C. (2013). Predicting interactions in online social networks: An experiment in Second Life. In *Proceedings of the 4th International Workshop on Modeling Social Media*, MSM '13, pages 1–8, New York, NY, USA. ACM.

Strrle, H. and Fish, A. (2013). Towards an operationalization of the physics of notations for the analysis of visual languages. In *Model-Driven Engineering Languages and Systems*, volume 8107 of *Lecture Notes in Computer Science*, pages 104–120. Springer.

Su, J. and Zhang, H. (2006). A fast decision tree learning algorithm. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 500–505, Boston, Massachusetts. AAAI Press.

Tang, J., Musolesi, M., Mascolo, C., and Latora, V. (2009). Temporal distance metrics for social network analysis. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 31–36, New York, NY, USA. ACM.

Teevan, J., Morris, M. R., and Panovich, K. (2011). Factors affecting response quantity, quality, and speed for questions asked via social network status messages. In Adamic, L. A., Baeza-Yates, R. A., and Counts, S., editors, *ICWSM*. The AAAI Press.

Teinemaa, I., Leontjeva, A., Dumas, M., and Kikas, R. (2015). Community-based prediction of activity change in skype. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 73–80, Beijing, China.

TIBCO (2014). Spotfire. Retrieved on 2014-03-25 from http://spotfire.tibco.com.

Toivonen, R. (2009). Social networks: modeling structure and dynamics. Technical report.

Vartak, M., Huang, S., Siddiqui, T., Madden, S., and Parameswaran, A. (2015). Towards visualization recommendation systems. In *Workshop on Data Systems for Interactive Analytics (DSIA)*, pages 1–6, Chicago, USA.

Venolia, G. D. and Neustaedter, C. (2003). Understanding sequence and reply relationships within email conversations: A mixed-model visualization. Technical Report MSR-TR-2002-102, Microsoft Research.

Viégas, F. B. and Smith, M. (2004). Newsgroup crowds and authorlines: Visualizing the activity of individuals in conversational cyberspaces. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4*, HICSS '04, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Wang, G., Gill, K., Mohanlal, M., Zheng, H., and Zhao, B. Y. (2013). Wisdom in the social crowd: An analysis of Quora. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1341–1352, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Weerkamp, W. and De Rijke, M. (2012). Activity Prediction: A Twitter-based Exploration. In *2012 Proceedings of SIGIR Workshop on Time-aware Information Access*, pages 1–5, Portland, USA.

Wei, W., Li, J., Cao, L., Ou, Y., and Chen, J. (2012). Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475.

Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451.

Xiong, R. and Donath, J. (1999). PeopleGarden: Creating data portraits for users. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, UIST '99, pages 37–44, New York, NY, USA. ACM.

Yang, L., Bao, S., Lin, Q., Wu, X., Han, D., Su, Z., and Yu, Y. (2011). Analyzing and predicting not-answered questions in community-based question answering services. In *25th AAAI Conference on Artificial Intelligence*, pages 1273–1278, Austin, Texas, USA.

Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. (2006). Sybilguard: Defending against sybil attacks via social networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 267–278, New York, NY, USA. ACM.

Yu, S. and Kak, S. (2012). A survey of prediction using social media. *Computing Research Repository (CoRR)*, abs/1203.1647:1–20.

Zhang, C. and Zhang, X. (2013). Uncertain attribute graph model of web social network and its application. In *Computer Science Education (ICCSE), 2013 8th International Conference on*, pages 111–116, Colombo, Sri Lanka.

Zhang, P., Wang, X., and Li, B. (2014). *Online Social Media Analysis and Visualization*, chapter Evaluating Important Factors and Effective Models for Twitter Trend Prediction, pages 81–98. Springer International Publishing, Cham.

Zheng, B., Thompson, K., Lam, S. S., Yoon, S. W., and Gnanasambandam, N. (2013). Customers behavior prediction using artificial neural network. In *Industrial and Systems Engineering Research Conference (ISERC)*, pages 700–709, San Juan, Puerto Rico. Institute of Industrial Engineerings.

Zhu, Y., Zhong, E., Pan, S. J., Wang, X., Zhou, M., and Yang, Q. (2013). Predicting user activity level in social networks. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 159–168, New York, NY, USA. ACM.