IDENTIFYING USER INTERESTS IN AN ONLINE DISCUSSION FORUM WITH
DEEP LEARNING

by

Nicholas Buhagiar

BSc (Honours) Physics, University of Ontario Institute of Technology, 2016

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2018

© Nicholas Buhagiar, 2018

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Identifying User Interests in an Online Discussion Forum with Deep Learning

Master of Science, 2018

Nicholas Buhagiar

Computer Science

Ryerson University

Abstract

The probabilistic topic model Latent Dirichlet Allocation (LDA) was deployed to model the themes of discourse in discussion threads on the social media aggregation website Reddit. Abstracting discussion threads as vectors of topic weights, these vectors were fed into several neural network architectures, each with a different number of hidden layers, to train machine learning models that could identify which discussion would be of interest for a given user to contribute. Using accuracy as the evaluation metric to determine which model framework achieved the best performance on a given user's validation set, these selected models achieved an average accuracy of 66.1% on the test data for a sample set of 30 users. Using the predicted probabilities of interest made by these neural networks, recommender systems were further built and analyzed for each user.

Acknowledgements

The deepest thanks must be extended to my MSc thesis supervisors, Dr. Bahram Zahir and Dr. Abdolreza Abhari. Their guidance and encouragement throughout this whole research process has not only helped this author develop the proper mindset for conducting research, but has also been deeply educational on applying many of the facets of data science and machine learning to complex problems. This work here could not have been accomplished without their direction, and their help throughout this whole process is greatly appreciated.

Gratitude must also be extended to the developers of Reddit [1], who not only created an excellent platform for people to communicate, but also made the data collected on their website publicly available via their API [2] for researchers to study. Thanks must also be given to Jason Baumgartner of pushshift.io [3] for collecting the Reddit comment data used in this study, and making it freely and easily available for others to access and use. His hard work and generosity have been of huge service to this work, and is greatly appreciated.

Contents

Abstract	iii
List of Tables	vii
List of Figures	viii
List of Abbreviations	xii
List of Appendices	xiii
1 Introduction	1
1.1 Thesis Statements	2
1.2 Research Contributions	2
1.3 Thesis Outline	2
2 Background Information and Related Work	4
2.1 What is Reddit?	4
2.2 Machine Learning	5
2.3 Recommender Systems	6
2.4 Latent Dirichlet Allocation	8
2.5 Natural Language Processing	11
2.6 Deep Learning	12
2.7 Supplementary Machine Learning Models	16
2.7.1 Naive Bayes Classifier	16
2.7.2 Support Vector Machine	17
3 Methodology	20
3.1 Dataset	20
3.2 Evaluation Metrics	26

3.2.1	F2-Score	26
3.2.2	Accuracy	29
3.3	Neural Network Architectures	29
3.4	Recommender System Framework	31
3.5	Initial Experiment: Using the F2-Score to Evaluate Performance	32
3.6	Final Experiment: Using Accuracy to Evaluate Performance	37
3.7	Computer Configurations	39
4	Experimental Results and Analysis	41
4.1	Initial Experiment Results and Analysis	41
4.2	Final Experiment Results and Analysis	48
5	Conclusion and Future Work	62
Appendix A LDA Model Topics: 100-Topic Model		65
Appendix B Neural Network Architectures		73
Appendix C LDA Model Topics: 50-Topic Model		79
Appendix D LDA Model Topics: 25-Topic Model		83
Appendix E User Recommendations Topic Comparison		85
Bibliography		115

List of Tables

1	Dataset used for this research, illustrating the number of comments, the total number of discussion threads they collectively appeared in, and the total number of users who collectively made them, in each of the train, validation, and test sets.	21
2	Technical specifications of computers used in this study.	40
3	Mean Precision, F2-Score, and Recall values predicted by the sample users' neural networks on their corresponding test sets.	45
4	Normalized confusion matrices for the test set of each user in the sample set. The values in a given user's confusion matrix were normalized by dividing all 4 values by their sum. The mean confusion matrix was calculated as simply the average values of these normalized confusion matrices.	46
5	Normalized confusion matrices for the test set of each user in the sample set. The values in a given user's confusion matrix were normalized by dividing all 4 values by their sum. The mean confusion matrix was calculated as simply the average values of these normalized confusion matrices.	51

List of Figures

1	Simple neural network architecture.	13
2	Plot of the ReLU function $f(z) = \max(0, z)$ [4].	14
3	Plot of the sigmoid/logistic function $f(z) = 1/(1 + \exp(-z))$ [4].	14
4	A visualization of a maximal margin classifier dividing a feature space into two distinct regions for each class of observations, created by following the methodology for plotting support vector machines in Python demonstrated in VanderPlas [5]. The bold line dividing the two regions represents the hyperplane, while the dashed lines are the margin surrounding it [5]. Any observations lying on this margin are the support vectors [6].	18
5	A visualization of a support vector machine with an RBF kernel dividing a feature space into two distinct regions for each class of observations, created by following the methodology for plotting support vector machines in Python demonstrated in VanderPlas [5]. The bold line dividing the two regions represents the hyperplane, while the dashed lines are the margin surrounding it [5]. Any observations lying on this margin are the support vectors [6].	19
6	Bar chart of the number of discussion threads in the training set exhibiting a given number of topics, each with a probability greater than 0.01.	23
7	Bar chart of the number of discussion threads in the training set exhibiting each of the LDA model's topics with a probability greater than 0.01.	23
8	Word cloud visualization of the ten most probable words comprising the most popular topic found by the LDA model in the training set. The greater the size of the word in the visualization, the greater the probability it has of occurring in that topic.	24

9	Framework of the most complex neural network architecture used in this work. Out of six neural network frameworks, the first contained no hidden layers, and each subsequent framework contained 1 additional hidden layer with half the number of nodes found in the previous layer floored until the above framework was reached.	31
10	F2-Scores obtained for each of the 30 sample users' test sets in relation to their corresponding selected neural network architectures, represented by the number of hidden layers they contained.	41
11	Bar chart of selected neural network architectures for the sample set of users, with the neural network architecture being represented by the number of hidden layers it contained. Due to no users having a selected model with 0 or 1 hidden layers, 0 and 1 are not present on the x-axis.	42
12	Validation F2-Score for each sample user's selected model at each epoch of training.	43
13	Kernel Density Estimation (KDE) plots of F2-Score test results obtained for the sample set of users using several different methods of classification. A KDE plot estimates the density of given data based on a set function or kernel, which was Gaussian for this visualization. <i>ANN</i> is an abbreviation for artificial neural network, <i>NB</i> is an abbreviation for the naive Bayes classifier, and <i>SVM</i> is an abbreviation for support vector machine.	44
14	Accuracy scores obtained for each of the 30 sample users' test sets in relation to their corresponding selected neural network architectures, represented by the number of hidden layers they contained.	49
15	Bar chart of selected neural network architectures for the sample set of users, with the neural network architecture being represented by the number of hidden layers it contained. Due to no users having a selected model with zero hidden layers, zero is not present on the x-axis.	50

16	Validation accuracy score for each user’s selected model at each epoch of training.	52
17	Kernel Density Estimation (KDE) plots of test accuracy results obtained for the sample set of users using several different methods of classification. A KDE plot estimates the density of given data based on a set function or kernel, which was Gaussian for this visualization. <i>ANN</i> is an abbreviation for artificial neural network, <i>NB</i> is an abbreviation for the naive Bayes classifier, and <i>SVM</i> is an abbreviation for support vector machine.	53
18	Line graph of test accuracy results obtained for each individual user in the sample set using several different methods of classification, sorted by test accuracy score for the 100-Topic deep learning approach in decreasing order. <i>ANN</i> is an abbreviation for artificial neural network, <i>NB</i> is an abbreviation for the naive Bayes classifier, and <i>SVM</i> is an abbreviation for support vector machine.	54
19	Bar graph showing the number of users who had a given number of recommendations that matched discussion threads they had actually been seen to contribute to, out of 20 recommendations made using each user’s given test set. The 20 recommendations made to each user in this approach corresponded to the 20 discussion threads in their corresponding test set that had the highest probability of being of interest to them as deemed by their selected interest predictor.	56

20	Bar graph showing the number of users who had a given number of recommendations that matched discussion threads they had actually been seen to contribute to, out of 20 recommendations made using each user's given test set. The 20 recommendations made to each user in this approach were discussion threads randomly selected from their corresponding test set, using their user interest probabilities as deemed by their selected interest predictor as selection weights.	57
21	Bar graph comparing the mean topic distribution of all the discussions User C contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User C in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	58
22	Bar graph comparing the mean topic distribution of all the discussions User Ω contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Ω in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	59
23	Bar graph comparing the mean topic distribution of all the discussions User H contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User H in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	60

List of Abbreviations

ANN - Artificial Neural Network

API - Application Programming Interface

NB - Naive Bayes

FN - False Negatives

FP - False Positives

hLDA - Hierarchical Latent Dirichlet Allocation

KDE - Kernel Density Estimation

LDA - Latent Dirichlet Allocation

MOOC - Massive Open Online Course

NLP - Natural Language Processing

NLTK - Natural Language Toolkit

RBF - Radial Basis Function

ReLU - Rectified Linear Unit

RMSE - Root-Mean-Square Error

SVM - Support Vector Machine

TN - True Negatives

TP - True Positives

List of Appendices

1	Simplest neural network architecture used in this work, with 0 hidden layers.	73
2	Neural network architecture used in this work, with 1 hidden layer.	74
3	Neural network architecture used in this work, with 2 hidden layers.	75
4	Neural network architecture used in this work, with 3 hidden layers.	76
5	Neural network architecture used in this work, with 4 hidden layers.	77
6	Most complex neural network architecture used in this work, with 5 hidden layers.	78
7	Bar graph comparing the mean topic distribution of all the discussions User A contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User A in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	85
8	Bar graph comparing the mean topic distribution of all the discussions User B contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User B in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	86
9	Bar graph comparing the mean topic distribution of all the discussions User C contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User C in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	87

10	Bar graph comparing the mean topic distribution of all the discussions User D contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User D in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	88
11	Bar graph comparing the mean topic distribution of all the discussions User E contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User E in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	89
12	Bar graph comparing the mean topic distribution of all the discussions User F contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User F in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	90
13	Bar graph comparing the mean topic distribution of all the discussions User G contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User G in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	91
14	Bar graph comparing the mean topic distribution of all the discussions User H contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User H in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	92

15	Bar graph comparing the mean topic distribution of all the discussions User I contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User I in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	93
16	Bar graph comparing the mean topic distribution of all the discussions User J contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User J in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	94
17	Bar graph comparing the mean topic distribution of all the discussions User K contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User K in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	95
18	Bar graph comparing the mean topic distribution of all the discussions User L contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User L in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	96
19	Bar graph comparing the mean topic distribution of all the discussions User M contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User M in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	97

20	Bar graph comparing the mean topic distribution of all the discussions User N contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User N in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	98
21	Bar graph comparing the mean topic distribution of all the discussions User O contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User O in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	99
22	Bar graph comparing the mean topic distribution of all the discussions User P contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User P in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	100
23	Bar graph comparing the mean topic distribution of all the discussions User Q contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Q in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	101
24	Bar graph comparing the mean topic distribution of all the discussions User R contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User R in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	102

25	Bar graph comparing the mean topic distribution of all the discussions User S contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User S in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	103
26	Bar graph comparing the mean topic distribution of all the discussions User T contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User T in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	104
27	Bar graph comparing the mean topic distribution of all the discussions User U contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User U in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	105
28	Bar graph comparing the mean topic distribution of all the discussions User V contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User V in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	106
29	Bar graph comparing the mean topic distribution of all the discussions User W contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User W in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	107

30	Bar graph comparing the mean topic distribution of all the discussions User X contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User X in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	108
31	Bar graph comparing the mean topic distribution of all the discussions User Y contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Y in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	109
32	Bar graph comparing the mean topic distribution of all the discussions User Z contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Z in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	110
33	Bar graph comparing the mean topic distribution of all the discussions User Θ contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Θ in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	111
34	Bar graph comparing the mean topic distribution of all the discussions User Ψ contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Ψ in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	112

35	Bar graph comparing the mean topic distribution of all the discussions User Ω contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Ω in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	113
36	Bar graph comparing the mean topic distribution of all the discussions User Φ contributed to in the training set (labelled as <i>Mean User Interests</i>) with the mean topic distribution of all the discussions recommended to User Φ in their given test set (labelled as <i>Mean Recommendations</i>) sorted by <i>Mean User Interests</i> topic weight in decreasing order.	114

1 Introduction

The modelling of the interactions between entities in a system has always been of interest to the scientific community. With the inception of the World Wide Web, humans are increasingly interacting with content made available online. The growing amount of this content available on the web has led web application users to be increasingly exposed to an overwhelming amount of articles, videos, pictures, etc. This overwhelming amount of content has led to the phenomenon of online users being totally unaware of a great deal of available material that he/she would be interested in. Recommender systems have been built as a means in which to bypass this problem, presenting content to a user which has been predicted to be of interest to him/her in the hope that he/she will interact with it [7]. The stunning growth of the World Wide Web has also enabled online users to interact with one another to discuss such web content. One immensely popular online platform that facilitates these discussions is the social media aggregation website Reddit [1].

Reddit is an online venue that enables users of its platform to post links to articles, photos, videos, etc. and discuss its content with other users in the form of a discussion thread associated with that content [8]. Within these threads, users can both comment on the web content as well as reply to the comments made by other users [8]. Reddit also allows users to upvote/downvote submitted web content as well as the comments made by others [8]. With the constant influx of new content being posted on Reddit, and the ceaseless production of comments made by its users, it is near impossible for a user of Reddit to be aware and contribute to all the content and associated discussions that he/she would be interested in contributing. This issue is not limited to Reddit, but is a problem that can be applied to users who contribute to any online discussion forums with a large number of users. Previous studies have been conducted in developing recommender systems for users in these online forums. However, no previous research seems to have been done for an online discussion forum with a high user base like Reddit [8].

The application of deep learning to this problem also does not seem to have had much

study. Deep learning comprises a branch of algorithms that has led to huge advances in fields such as speech recognition and computer vision. To the best of this author’s knowledge, the study of the use of deep learning as a means in which to classify user interests in an online forum seems to be lacking in the scientific literature.

1.1 Thesis Statements

This work seeks to demonstrate that the discussions in an online forum that a user is interested in can be identified using deep learning, and that these deep learning models can be used as the basis for a recommender system to recommend other discussions for a user to contribute to. This work also seeks to confirm that this model, identifying what discussions a user would be interested in, can be developed for a user of an online forum consisting of hundreds of thousands of discussion threads and users.

1.2 Research Contributions

The contributions made by this work to the field of data science are as follows:

- The first implementation in the literature of using deep learning to identify what discussions in an online forum a given user would be interested in to the best of this author’s knowledge
- The development of these deep learning models for users in an online discussion forum of a scale not previously seen in the literature to the best of the author’s knowledge
- An outline for extending these deep learning models to serve as a recommender system to users

1.3 Thesis Outline

The remainder of this work is divided into four main chapters. Chapter 2 discusses both some of the theory behind the methodology and analysis conducted in this work, as well

as the work of others that related to and influenced this study. Chapter 3 outlines the experiments made to confirm this work's thesis statements. Chapter 4 presents the results obtained from these experiments, as well as analyses of these results and their justification of the thesis statements. Chapter 5 outlines both concluding remarks on this research, as well as plans for future work.

2 Background Information and Related Work

2.1 What is Reddit?

Reddit is a web content aggregation website, primarily used as a forum for the promotion and discussion of what’s currently trending in the world [9]. Accessing the home or front page of the self-proclaimed “front page of the internet” [1], one can find a list of trending posted web content. Selecting one of the given links (usually linking to a news article, photo, or photo), one can view said content or also view and comment on its corresponding discussion thread [9]. Posted submissions do not always have to be linked to web content, but can also be discussion prompts or questions to be discussed and answered by the community. Reddit users can upvote or downvote content based on whether or not they thought the given content had a positive or negative value respectively to the Reddit community [9]. Comments in a discussion thread can also be upvoted or downvoted by users [9]. Web content is given a submission score indicating the difference between the number of upvotes and downvotes the web content received [9]. On a given user’s page, one can see every comment and submission made by the user (with their associated score) as well as their overall post and comment ‘karma’ which are analogous to their overall score for submissions and comments respectively [10]. ‘Karma’ is a rough indication of how many upvotes subtracted by downvotes one has received for their posts and comments [10], and are essentially just brownie points indicating the impact a user has had on the community [9]. Reddit is also divided into numerous subcommunities called subreddits [9]. Subreddits are created by Reddit users and can be based on anything. There is a ‘relationship advice’ subreddit, a ‘science’ subreddit, a ‘babyelephantgifs’ subreddit, and many more [11]. Users can subscribe to a subreddit if they enjoy the content posted within it [9]. This will lead to content from that subreddit appearing on the user’s front page or main feed to allow easy viewing and access to the subreddit’s content [9]. Users are also able to subscribe to as many subreddits as they would like, and are actually immediately subscribed to some of the most popular subreddits

when they first create their account [9].

Reddit is a hugely popular website, with a great deal of traffic. In 2015 alone, Reddit had over 82.54 billion pageviews, over 73.15 million submissions, 725.85 million comments made by 8.7 million different authors containing 19.36 billion words, with 6.89 billion upvotes occurring, and 88,700 active subreddits [12]. As of November 2017, Reddit was ranked as the 4th most visited website in the United States of America, with over 330 million monthly active users, over 138 thousand active subreddits, and 14 billion average screen views per month [8].

2.2 Machine Learning

The methods that were used extensively in this work to model data and gather insight and make predictions from it were machine learning algorithms [13]. Machine learning refers to the collection of techniques used to computationally model data [13]. This process of modelling the data is referred to as training the model [13]. When developing a machine learning algorithm, usually a dataset is partitioned into three separate sets: a training set, a validation set, and a test set [13]. The training set is used to offer examples of data instances for the machine learning model to learn from [13]. The validation set is used during training to see how well the model performs on data it has not previously seen, as well as to perhaps offer insight on any changes that need to be made to the model to make it more robust [13]. The test set is used to serve as a final evaluation for the performance of the machine learning model on new data [13].

Machine learning can be divided into two main subcategories, supervised and unsupervised learning [13]. Supervised learning algorithms try and model the relationship between some observable quantities and a related outcome of these quantities [13]. These observable quantities will be referred to interchangeably as inputs, features, or predictors and the related outcomes will be referred to interchangeably as outputs, labels, responses, or classes in this work. If these outputs are continuous values, than the supervised learning model

can be referred to as a regression function, while if the outputs are discrete, then the model can be referred to as a classifier [13]. Unsupervised learning models refer to models that only require features of the data and try to discover hidden structure within it [13]. Both supervised and unsupervised machine learning methods were used in this research.

2.3 Recommender Systems

Recommender systems are a class of models, primarily web applications, that focus on predicting how a user will respond to a collection of options [7]. Recommender systems are used for a wide range of content, including products on online retail stores, movies on streaming media services such as Netflix, and online media such as news articles, online videos, blogs, etc. [7]. There are two primary categories of recommender systems: content-based systems and collaborative-filtering systems [7]. Content-based systems recommend content to a user that shares features or properties similar to those of the content the user has shown an interest in [7]. Conversely, collaborative-filtering systems recommend content to a user that other, as identified by the recommender system, like-minded users have shown an interest in [7]. For example, if a user bought a book on an online shopping service, a content-based recommender system would most likely recommend other books to that user, while a collaborative-filtering system would recommend other items bought by users who also bought the same book.

Recommender systems are used and still being developed and refined in a variety of online platforms, including discussion forums. A collaborative-filtering recommender system was developed by Castro-Herrera et al. [14] for instance for the customer management system SugarCRM. Analyzing user feature requests distributed across almost 300 discussion threads, Castro-Herrera et al. [14] used a clustering approach to connect users of similar interests with one another. Developing a recommender system for an online forum associated with a university course, Abel et al. [15] discovered that basing recommendations on the discussions a user participated in yielded superior results to using the discussion thread voting system as

a basis for recommendations. In their study, Abel et al. [15] also found that not much data was necessary to build a precise recommender system. A recommender system for a forum associated with a massive open online course (MOOC) was similarly built by Yang et al. [16], using several different approaches including their proposed adaptive matrix factorization method to achieve this. A recommender system was also developed by Li et al. [17] for online news articles, using the content of the article as well as the comments made in the discussion thread associated with the article to recommend articles of similar subject matter to a user. A study was further conducted by Li et al. [17] on how the subject matter of articles related to one another, examining whether or not another article was simply a duplicate of the original article, or detailed a more general or more specific aspect of the original article's content. In a separate experiment, the comments made in the discussion threads related to news articles was further analyzed by Messenger and Whittle [18]. Extracting ten phrases from a given article's discussion thread, such as the most mentioned terms or most mentioned people, Messenger and Whittle [18] entered each of these phrases into a Google News search engine, recommending to a user the top article returned by the search engine for each phrase. Though these studies revealed interesting information about applying a recommender system to a discussion forum, none of the above studies seemed to analyze datasets with more than a few thousand discussion threads. Since it is unclear how successful these approaches would be in a much larger discussion forum with hundreds of thousands of users and threads, this work seeks to elucidate this matter.

Recommender systems received a huge boost in research during the Netflix Prize challenge from 2006-2009 [19]. The Netflix Prize was a competition in which challengers would receive a \$1 million prize if they could yield recommendations better than Netflix's in-house recommender system by 10% on the root-mean-square error (RMSE) evaluation metric [19]. This was to be accomplished using a dataset released by Netflix consisting of over 100 million ratings made by almost 500,000 users on around 18,000 movies [19]. Numerous lessons were learned by competitors in the Netflix Prize challenge, such as the value of crowdsourcing

work and the value in ensemble methods in predictive modelling [19]. Since one of the goals of this work was to recommend discussion threads to users based on other threads they have participated in, the decision was made to develop a content-based recommender system. This would recommend threads to users with similar topics of discussion as other threads they had participated in. One of the biggest lessons learned from the previously mentioned Netflix Prize challenge that influenced this work was the value of latent factor models in predictive modelling, in which many single (i.e. not ensemble) models were constructed [19]. Latent factors/variables are unobservable attributes of a given object. Latent factor models therefore try to represent given observable variables using latent, unobservable variables, which are typically much less in number than their observable counterparts [20]. This serves as a means in which to reduce the number of features in a dataset [20]. During the Netflix Prize challenge, this was used to simplify how users and movies were represented, with movies being represented by latent factors such as presence of political humour, dialogue to action ratio, etc. and users being represented by their predilections for those latent factors (identified by the rating they gave for movies highly showcasing a given latent factor) [19]. Since the comments collected in discussion threads in the dataset consist of hidden, latent factors as well (e.g. sentiment, topic, style, etc.) it was decided to use a latent factor model to simplify the representation of the data.

2.4 Latent Dirichlet Allocation

Identifying the subject matter of discussion threads on a large-scale is of course difficult and time-consuming. Thankfully, researchers have developed several different topic modelling algorithms to complete this task computationally. One of these such algorithms, the probabilistic topic model Latent Dirichlet Allocation (LDA) [21] was implemented in this work to achieve this task. Developed by Blei et al. [21], LDA is an unsupervised machine learning algorithm that can be trained on a corpus of documents to represent any document as a mixture of the corpus topics with given probabilities. In this context, a topic is defined

as itself a distribution of all the words in the corpus vocabulary [22]. LDA makes three key assumptions about a corpus: the number of topics is fixed and known, the order of the words in each document is irrelevant, and the order of the documents themselves within the corpus is irrelevant [22]. These assumptions were deemed as acceptable for the work of this thesis since the number of topics can just be set to a particular value, and after analyzing the results, these number of topics could perhaps be changed to improve the predictions. Since the focus of this work was primarily on the overall themes of a given discussion thread, the order of words in a thread were deemed as unimportant as well, with each of these documents simply being treated as a “bag of words”, a term commonly used in the literature to refer to documents where word ordering is ignored [22]. The order of the discussion threads in the datasets were also deemed irrelevant since each thread was treated as a single distinct document that a given user may or may not be interested in participating.

One further assumption LDA makes about a corpus of documents is the process in which these documents were generated, outlined as follows. First, the corpus topics are generated as a distribution of words in the corpus vocabulary [22]. This word distribution for a given topic is a multinomial distribution [23], and can be understood as a probability distribution of each word in the corpus vocabulary being exhibited by that given topic [22]. A Dirichlet distribution is then used to choose the topic distribution for a given document [22]. The Dirichlet distribution can be described as a probability density function over distributions, where in LDA these distributions would be the topic multinomial distributions [23]. Each word in the document is then randomly generated from the document’s Dirichlet distribution of topics and the corresponding topics’ multinomial distributions of words [22].

Making the assumption that this was done for every document in the corpus, LDA seeks to backtrack and discover the overall topic structure that likely created this corpus [22]. The method in which LDA achieves this is based on its assumption that there is a joint probability distribution over the observed and hidden variables in the corpus, where the observed and hidden variables are the words and topic structure of the documents respectively [22]. This

joint probability distribution can be defined by Equation (1), where β_i is the i^{th} topic, θ_d is the topic proportions for document d , z_d is the topic assignments for document d , w_d is the words in document d , $z_{d,n}$ and $w_{d,n}$ are the topic assignment and word respectively for the n^{th} word in document d , N is the total number of words in a given document, K is the total number of topics, and D is the total number of documents in the corpus [22].

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^K p(\beta_i) \prod_{d=1}^D p(\theta_d) \left(\prod_{n=1}^N p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right) [22] \quad (1)$$

Using this joint probability distribution, the conditional/posterior distribution of the hidden variables given the observable variables can be inferred [22]. This posterior distribution can be summarised by Equation (2) [22].

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} [22] \quad (2)$$

The numerator in Equation (2) is just the joint probability distribution from Equation (1), while the denominator is the probability of having the given observable variables regardless of the overall topic structure [22]. The denominator of Equation (2) can be calculated as the sum of the joint probability distribution for every possible configuration of the overall topic structure for the corpus [22]. However, this calculation is infeasible due to the vast number of overall topic structure configurations possible for a given corpus, therefore, approximation methods are utilized to circumvent this issue [22]. The implementation of LDA in the Python topic modelling package Gensim that was used for this work used the methodology presented in Hoffman et al. [24] to compute this approximation [25].

The utilization of LDA to model discussions in an online forum has been studied by researchers in several interesting ways. Similar to this work, both Tran and Ostendorf [26] and Weninger et al. [27] trained an LDA model on Reddit comments for their respective research. LDA was used by Tran and Ostendorf [26] to model the topics in a collection

of Reddit comments, analyzing whether or not this information could be used to identify whether or not a given user or discussion belonged to a particular subcommunity. Whether or not there was any correlation between the subject matter of a user’s comments and the given user’s popularity within that subcommunity was further studied by Tran and Ostendorf [26]. This possible popularity correlation was studied for discussion threads as well [26]. Conversely, Weninger et al. [27] used an extension of LDA called Hierarchical LDA (hLDA) to collect comments into a topic hierarchy, with comments of similar subject matter being more closely related with one another. Since comments in a Reddit discussion thread can be seen as forming a hierarchical tree, with comments, replies to those comments, and replies to those replies, Weninger et al. [27] found that there was a correlation between the comment hierarchy and topic hierarchy, with comment subbranches mirroring sub topics. The comment and topic hierarchy of a discussion thread were disregarded in the work of this thesis, with the preference being to focus on the overall subject matter of a given thread.

Two extensions of LDA, LinkLDA and CommentLDA, were used by Yano et al. [28] to not only model the themes of discourse in discussion threads related to political articles, but to predict who would comment on a new post as well as what would be the content of that comment. Using several different variations in which to make these predictions, Yano et al. [28] achieved precision scores between 25%-37%. LDA was also used by Ezen-Can et al. [29] to identify the subject matter of clusters of comments made in the associated discussion forum of a MOOC.

2.5 Natural Language Processing

A natural language is a language used by humans to communicate with one another, like English or French, and is hard to explicitly frame due to its constant evolution [30]. Natural language processing (NLP) can be viewed as an umbrella term to encompass any kind of manipulation of natural language using computers [30]. NLP was extensively used as a means in which to preprocess all the comments in the discussion threads prior to training

an LDA model on the dataset in this work. NLP methods that were used included converting text to lowercase, the removal of stop words, the tokenization of words into unigrams, and the stemming of those unigrams.

Converting all text in a corpus to lowercase is done to ensure that capitalized words are not distinguished from their lowercase form. Stop words are words that appear with a high frequency in a corpus that offer little contextual information [30], and are therefore usually deemed irrelevant and removed. Common stop words are *a*, *and*, *it*, and *the* [31]. Tokenization is the methodology of segmenting a given text into *tokens* based on a particular criterion [32]. A simple tokenizer which tokenizes a string based on whitespace would tokenize:

$$\text{'How are you?'} \rightarrow [\text{'How'}, \text{'are'}, \text{'you?'}]$$

Stemming is the process of removing any affixes from a token so that one is left with only its root word [30]. If implemented correctly, this will result in the tokens *look*, *looks*, and *looking* all being simply converted to the token *look*. N-grams are sequences of tokens of a given quantity N [32]. Though using N-grams with N equal to 2 or 3 (referred to as bigrams and trigrams respectively in the literature [32]) could be enabled to identify collections of tokens instead of just single tokens (unigrams) [30] in a corpus (i.e. identify *New York City* as one term, instead of the three separate terms *new*, *york* and *city*), this requires additional computational power to carry out.

2.6 Deep Learning

With LDA serving as the method in which to extract features from the dataset of discussion threads, deep learning was the method used to predict the discussions a given user was interested in. The decision to use deep learning for this task was motivated by the improvements yielded by these algorithms in a variety of domains such as speech recognition and object detection over other more traditional machine learning approaches [4].

Deep learning refers to a collection of computational models that take inputted low

level data, and after passing it through several layers of non-linear modules, transforming it into a higher level representation [4]. These computational models are referred to as neural networks, and deep learning corresponds to the learning of a “deep” neural network consisting of many of these “hidden” layers of non-linear modules [33]. A simple “neural network” architecture can be seen in Fig. 1.

Figure 1: Simple neural network architecture.

At each node in a neural network, the sum of all weighted nodes connected to it from the previous layer with an additional bias term is calculated [4]. A non-linear activation function is then applied to this calculated sum and is used to calculate the value of nodes in the following layer [4]. Several different activation functions are in use today, including rectified linear units, better known as ReLUs ($f(z) = \max(0, z)$), plotted in Fig. 2) and the sigmoid/logistic function ($f(z) = 1/(1 + \exp(-z))$), plotted in Fig. 3) [4].

Figure 2: Plot of the ReLU function $f(z) = \max(0, z)$ [4].

Figure 3: Plot of the sigmoid/logistic function $f(z) = 1/(1 + \exp(-z))$ [4].

In the case of supervised deep learning, these predictions made by these neural networks are then compared with their actual values via some objective function that calculates the error between the two [4]. The deep learning model weights used to calculate these higher

level representations of the input data are then adjusted in order to minimize this error [4]. Using these modified parameters, the model can make new predictions for the input data and repeat this process [4]. Each of these training steps is referred to as an epoch, and this process continues until a particular stopping criterion is reached [4].

The objective function, also referred to as a cost function, loss function, or error function [34], that was used to train the neural networks in this work was binary cross-entropy loss. The formula for binary cross-entropy can be seen in Equation (3), where E_c is the cross-entropy, t represents the true response values, and p represents the predicted response values.

$$E_c(t, p) = -(t \cdot \log(p) + (1 - t) \cdot \log(1 - p)) [35] \quad (3)$$

Most deep learning algorithms involve the application of optimization to minimize this cost function at every epoch of training [34]. Though there are several optimization methods available, the decision was made to use the Adam Optimizer [36] due to the author's familiarity with it.

Deep learning has been used as the basis for a recommender system in several domains. Deep learning was used by Covington et al. [37] to recommend videos for users to watch on Youtube. A film recommender system was developed using neural networks by Christakou et al. [38]. Deep learning was used by both Van den Oord et al. [39] and Wang and Wang [40] to develop a system that recommended songs to users. Recommending content to users in several different online domains using deep learning was also developed by Elkahky et al. [41]. However, though Wong et al. [42] used neural networks to model what constitutes a useful comment made in an online domain and predict whether or not any new comments yield helpful information, as of yet no other work has employed deep learning to serve as a recommender system in a discussion forum to the best of the author's knowledge.

2.7 Supplementary Machine Learning Models

Though as previously stated, deep learning was the method that was used to identify which discussions a given user was interested in, these results were also validated using other, simpler machine learning methods. These additional machine learning methods that were used were Gaussian naive Bayes classifiers and support vector machines (SVMs) with radial basis function (RBF) kernels.

2.7.1 Naive Bayes Classifier

A naive Bayes classifier is a model that implements Bayes theorem with the naive assumption of feature independence for classification [31]. Bayes theorem states that the probability of a particular output occurring given its features is equal to the response's prior probability multiplied by the probability of each feature occurring given the corresponding response [31]. This is in turn divided by the probability of every feature occurring [31]. This can be summarized in Equation (4), where y represents a particular response, and x_i represents a given feature [31].

$$P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)} [31] \quad (4)$$

Since the thread topic weights are continuous, a Gaussian naive Bayes classifier, which assumes that the probability of a given feature x_i occurring given a response y is Gaussian, was used [31]. This is summarized in Equation (5) [31], where $\sigma_{y|x_i}$ and $\mu_{y|x_i}$ is the standard deviation and mean respectively of a response y given feature x_i [31].

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{y|x_i}^2}} \exp\left(-\frac{(x_i - \mu_{y|x_i})^2}{2\sigma_{y|x_i}^2}\right) [31] \quad (5)$$

The naive Bayes classifier determines the probability of each response occurring using Bayes theorem given in Equation (4), with both the prior probability of the response occurring, and the mean and standard deviation of a response for a given feature both being cal-

culated using the training data [30]. Since the denominator in Bayes theorem, $P(x_1, \dots, x_n)$, is constant no matter the response, this can be ignored when calculating the posterior probability of each response occurring for a given observation [31].

2.7.2 Support Vector Machine

A maximal margin classifier is a model that uses a hyperplane to separate observations and classify them within a given feature space [6]. Since there can be an infinite number of hyperplanes that can potentially do this, a maximal margin classifier chooses the one that has the highest margin or distance between the points from either category closest to the hyperplane [6]. These points that determine the position and orientation of the hyperplane are referred to as the hyperplane's support vectors [6]. A plot demonstrating the use of a maximal margin classifier to divide observations of two classes can be seen in Fig. 4, designed, as with all other plots in this work, using the Python plotting package Matplotlib [43].

Figure 4: A visualization of a maximal margin classifier dividing a feature space into two distinct regions for each class of observations, created by following the methodology for plotting support vector machines in Python demonstrated in VanderPlas [5]. The bold line dividing the two regions represents the hyperplane, while the dashed lines are the margin surrounding it [5]. Any observations lying on this margin are the support vectors [6].

A support vector, or soft margin, classifier is an extension of the maximal margin classifier that allows for observations to cross the hyperplane’s margin or even appear across the hyperplane in the other label’s feature space [6]. This allows for greater robustness in the model, with the severity of the violations allowed being tuned using the tuning parameter C [6]. In the Scikit-Learn implementation of the support vector classifier used in this work, the greater C is, the less tolerant the model is to observations violating the hyperplane’s margin [31].

A support vector machine is a special class of support vector classifiers that increase the feature space by allowing non-linear decision boundaries between observation class regions [6]. The specific way in which the feature space is enlarged is referred to as a kernel, of which there are numerous forms [6]. A popular kernel for support vector machines that will

be used in this work is the radial kernel [6], also known as the radial basis function (RBF) kernel [31]. A plot demonstrating the use of a support vector machine with an RBF kernel to divide observations of two classes can be viewed in Fig. 5,

Figure 5: A visualization of a support vector machine with an RBF kernel dividing a feature space into two distinct regions for each class of observations, created by following the methodology for plotting support vector machines in Python demonstrated in VanderPlas [5]. The bold line dividing the two regions represents the hyperplane, while the dashed lines are the margin surrounding it [5]. Any observations lying on this margin are the support vectors [6].

3 Methodology

3.1 Dataset

The data used in this study consisted of a collection of comments made on Reddit in June 2017, collected by Jason Baumgartner of pushshift.io [3] using Reddit’s API [2]. Each of these data samples consisted of plenty of information regarding the associated comment, including the comment’s author, the ID of the discussion thread that it was made in and the comment content itself. Having the associated discussion thread ID for each comment, these comments were aggregated into the discussions they appeared in. Knowing which user authored each comment, the discussion threads that a given user did or did not contribute to could also be easily identified. In order to build a model to identify which discussions a user would be interested in, the decision was made to treat this as a supervised machine learning task, where a model would be built for each user, treating the discussion topics as model predictors and whether or not the user contributed to the discussion as its label. In this work, discussions that a user did contribute to are referred to as positive instances, while ones in which he/she did not contribute to are referred to as negative instances.

As one would expect due to the vast amount of content on a popular platform such as Reddit, there was a huge class imbalance when training a machine learning model for a given user, with the number of discussions they had contributed to being vastly smaller in proportion to the discussions in which they had not. Considering these circumstances, large repositories of data were collected to ensure that there was enough positive instances for each user under study in the dataset to enable these machine learning models to have a decent understanding of a user’s varied interests. These large repositories of data were collected as chunks of comments, with these chunks being equally partitioned in terms of memory size into a training, validation, and test set. Due to this method of partitioning the dataset, this led to the possibility of different comments being made in the same discussion thread being partitioned into the train, validation, and test sets. These partitioned discussion threads

Table 1: Dataset used for this research, illustrating the number of comments, the total number of discussion threads they collectively appeared in, and the total number of users who collectively made them, in each of the train, validation, and test sets.

Dataset	# of Comments	# of Discussion Threads	# of Users
Train	3,691,514	416,300	869,680
Validation	3,685,952	411,788	865,161
Test	3,655,537	415,879	858,990

were treated as separate discussions consisting of their own unique comments. Since which discussions a user had contributed to would be different for each user, the train, validation, and test sets were made to be relatively the same size to help ensure that every user under study had discussions they had contributed to in each of their train, validation, and test sets. The dataset used in this study can be viewed in Table 1.

In order to represent these discussions in a manner suitable for a machine learning model to learn from, the probabilistic topic model LDA was trained on the comments in the training set, in order to identify the topics of discourse in the data, as well as to be further used to represent discussions as vectors of topic weights. In order to first preprocess the data prior to training the LDA model on it, the comments in the training set were vectorized into a matrix of token counts using the CountVectorizer object in the Python machine learning package Scikit-Learn [31]. This involved tokenizing the comments, as well as removing all English stop words from the comments, though some comments were seen to be written in a different language, and removing any tokens that appeared in more than 50% or less than 5 of the comments. Only the most frequently occurring 100,000 tokens were considered given these constraints. These tokens were further stemmed using the Snowball Stemmer [44] from the Python natural language processing package the Natural Language Toolkit (NLTK) [30]. The remaining default parameters of the Scikit-Learn CountVectorizer object were used, including only considering unigrams of 2 or more alphanumeric characters, treating punctuation as a token separator and ignoring it, and converting all tokens to lowercase [31]. An LDA model was then trained on this corpus of comments using the Python topic

modelling package Gensim [25]. All of the default parameters for the Gensim LDA object were kept as such, including the number of topics being set to 100 [25]. A full list of the LDA model’s topics with their ten most popular words can be found in Appendix A.

Using this LDA model, the themes of discourse for a given discussion thread could be identified, representing it as a vector of topic weights, where the first element in the vector corresponded to the weight in which the discussion exhibited the first LDA model topic, the second exhibiting the discussion’s weight in regard to the second topic, and so on. A discussion would therefore be represented in the following format:

$$\begin{aligned} Thread_{\gamma} = & [0.221 \cdot Topic_0 + 0.000 \cdot Topic_1 + ... \\ & + 0.106 \cdot Topic_{98} + 0.000 \cdot Topic_{99}] \end{aligned}$$

When using this LDA model to identify a discussion’s topic weights, if any weight was found to be less than 0.01, it would be automatically set to 0. This would initially cause the total topic weights for a given discussion to not sum to 1. To rectify this, all of the topic weights for a given discussion would be divided by their sum, to ensure that the topic weights for a given discussion would thereby sum to 1. Considering this, one can view a plot that showcases the number of threads in the training set exhibiting a given number of topics with nonzero weight in Fig. 6. Viewing Fig. 6, one can see that most discussions exhibited only a few topics, usually less than 10. Surprisingly, some discussions had all their topic weights set to zero, due to their exhibition of all topics having a very low weight.

Figure 6: Bar chart of the number of discussion threads in the training set exhibiting a given number of topics, each with a probability greater than 0.01.

One can also view the number of discussions exhibited by each topic in Fig. 7.

Figure 7: Bar chart of the number of discussion threads in the training set exhibiting each of the LDA model's topics with a probability greater than 0.01.

As one can see viewing Fig. 7, the most popular topic in the training set was Topic 15. A word cloud to showcase the most probable words comprising Topic 15 and constructed using the Wordcloud package in Python [45] can be viewed in Fig. 8. Interestingly, and perhaps unsurprisingly, the most probable words in the most popular topic are words that generally appear in Reddit’s lexicon, such as *post* for discussion post, *messag*, presumably *message* stemmed, for comments or private messages that can be exchanged between users, *subreddit* for the subcommunities within Reddit, *rule* and *question*, possibly referring to a rule or question made on Reddit, and *moder*, presumably *moderator* stemmed, possibly referring to moderators on Reddit.

Figure 8: Word cloud visualization of the ten most probable words comprising the most popular topic found by the LDA model in the training set. The greater the size of the word in the visualization, the greater the probability it has of occurring in that topic.

Another sample topic found using the LDA model is shown below:

$$\begin{aligned}
Topic_45 = & [0.097 \cdot trump + 0.042 \cdot op + 0.030 \cdot presid + \\
& 0.029 \cdot news + 0.024 \cdot elect + 0.017 \cdot russia + \\
& 0.016 \cdot fake + 0.015 \cdot republican + \\
& 0.015 \cdot campaign + 0.015 \cdot obama + \dots]
\end{aligned}$$

Though clearly this topic concerns American politics, the theme of some other topics are not so easy to decipher due to the ambiguity of the most popular tokens encompassing it, such as the following:

$$\begin{aligned}
Topic_57 = & [0.037 \cdot yea + 0.035 \cdot hang + 0.033 \cdot master + \\
& 0.033 \cdot pure + 0.032 \cdot cold + 0.031 \cdot rang + \\
& 0.030 \cdot suit + 0.024 \cdot educ + \\
& 0.024 \cdot buddi + 0.023 \cdot smile + \dots]
\end{aligned}$$

With the discussions in the train, validation, and test sets represented as topic weight vectors, a machine learning model could then be trained on this corpus for a given user. This would be done by having each user have their own distinct representation of the data set, with a 1 or a 0 appended to each discussion thread vector indicating whether or not the user had or had not contributed to the given discussion thread respectively. Due to the previously discussed class imbalance in the training set, an oversampling technique was implemented to increase the number of positive instances in a given user's training set. This not only gave a given user's model more positive instances to train from, but also eliminated the possibility of a model that predicted nothing but negative instances seeming to perform well, due to the vast number of negative instances in relation to positive instances in the data set. This

oversampling technique was implemented by adding copies of positive instances to a given user’s training set until the ratio of positive to negative instances was approximately 1:1. Furthermore, the weights of these copied topic vectors were multiplied by a randomly sampled number in a uniform distribution between 0.9 (inclusively) and 1.1 (exclusively), enabling these copied vectors to be different from their original counterparts, but not too different so as to exhibit a different topic weight distribution. This storage and manipulation of data was achieved using the Python scientific computing package Numpy [46] and the Python data analysis library Pandas [47]. Implementing this oversampling technique resulted in the number of data instances in a given user’s training set being effectively doubled to around 800,000 discussion threads. Though this oversampling process was similarly implemented for a given user’s validation and test sets, no noise manipulation of the copied vectors was done in the test set so as not to unintentionally misrepresent it.

A sample set of 30 users was used in this study, who each participated in 50 to 200 discussions inclusively in each of the training, validation, and test sets. The lower bound of user participation in 50 discussions was invoked to ensure that there was enough positive instances in each of the training, validation, and test sets to hopefully gather a varied view of that user’s interests. The upper bound of participating in 200 discussions was used in an attempt to minimize the number of bots in the sample set. To ensure user privacy, no real usernames or any other identifiable information will be used or revealed in this work.

3.2 Evaluation Metrics

When training these machine learning models, two primary metrics were taken into consideration to evaluate these models’ performance: the F2-Score and Accuracy.

3.2.1 F2-Score

When contemplating how to properly evaluate the performance of these machine learning models, it was quickly understood that due to the effective impossibility of building a model

that would be able to perfectly identify whether or not a user would be interested in a given discussion, some errors were to be expected. This however begged the question of whether or not there should be a preference for what errors will occur. A false positive, in this work’s context, is a classification made by a machine learning classifier of a discussion being of interest to a user when it actually wasn’t [48]. A false negative, on the other hand, is the incorrect classification of a thread not being of interest to a user when it actually was [48]. During the administration of this work’s first experiment, outlined in Chapter 3.5, it was decided that models making false positive errors were less of a concern then producing false negative errors. The rationale for this being that a false positive classification could be the machine learning model identifying a discussion that a user would be interested in, but did not contribute to due to possibly being unaware of it. Therefore, false negative errors were deemed much less excusable than false positives. Accordingly, a metric that evaluated the number of false negative errors made by a model seemed to be the appropriate evaluation metric to use. The performance metric that evaluates this in a classification problem is recall, which is defined in Equation (6).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} [48] \quad (6)$$

Analyzing Equation (6), one can gather that minimizing the number of false negatives made by a model achieves a maximum value of 1.0 or 100% recall, with the minimum value for recall being 0.0. Though it would therefore seem that recall would be a suitable performance metric for these machine learning models, one shortcoming of using recall as an evaluation metric is that it would give a perfect score to a model that identified all threads of being of interest to a user, since there would be no false negatives to speak of. This issue can be bypassed by using the F measure in place of recall as the performance metric. The F measure is the weighted harmonic mean between a model’s precision and recall scores [48]. Precision is a performance metric similar to recall that yields a perfect score for a model that outputs no false positives, as opposed to false negatives for recall [48]. Similar to recall,

the precision score can be in the range from 0.0 to 1.0 inclusively. The formula for precision is defined in Equation (7).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} [48] \quad (7)$$

The general formula for the harmonic mean can be seen in Equation (8), where n is the number of terms, x_i is one of those given terms, and H is the harmonic mean of those terms [49].

$$H \equiv \left(\frac{1}{n} \sum_{i=1}^n \frac{1}{x_i} \right)^{-1} [49] \quad (8)$$

The rationale for using the harmonic mean as opposed to the arithmetic mean ($\frac{1}{n} \sum_{i=1}^n x_i$) of precision and recall could perhaps be better understood via example. If one had 100 discussion threads where only 10 were of interest for a given user and a model predicted that the user would be interested in all of these 100 discussions, the arithmetic mean between precision and recall would be 0.55, and would always be at least 0.5 since recall would be 1.0 [48]. Comparing this to the harmonic mean for the same situation, which would yield a value of 0.18, one can see that this gives a better indication of how poor the model truly is [48]. The F measure can be summarized by Equation (9).

$$\text{F measure} = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} [48] \quad (9)$$

The F measure can have any value between 0.0 and 1.0 inclusively. Modifying the value of β in the F measure equation changes the weight in which precision and recall impact the F measure. A value of 1 yields equal weight between the two, while a value less than 1 lends more weight to precision and a value greater than 1 gives more weight to recall. Due to the previously mentioned preference for recall over precision, the value of β was set to 2, thereby lending more weight to recall while not totally disregarding a model's precision score. The terms F measure and F2-Score are used interchangeably in this work.

3.2.2 Accuracy

Due to several oversights in using the F2-Score as an evaluation metric for this work’s initial experiment, discussed in Chapter 4.1, a decision was made to conduct a second experiment to predict user interests, this time using model accuracy as the chosen evaluation metric. Accuracy can be thought of as simply the percentage of correct predictions made by a model [48]. The formula for accuracy can be seen in Equation (10), where TP stands for true positives, FP for false positives, FN for false negatives, and TN for true negatives.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} [48] \quad (10)$$

The accuracy score can have a value between 0.0 and 1.0 inclusively.

3.3 Neural Network Architectures

As previously mentioned, deep learning was the machine learning algorithm used to predict user interests in these experiments. These deep learning models were constructed and trained using the Python deep learning library Keras [50], using the Python numerical computation library Theano [35] as a backend to handle low-level operations. Each user’s neural network had 100 input nodes corresponding to the 100 topic weights of each discussion thread vector, as well as a sigmoid output unit to identify whether or not a given discussion thread was of interest to a user. The weights in these neural networks were initialized using the Xavier Uniform Initializer [51], with all bias terms initialized to 0. The Adam Optimizer [36], with all default parameters including a default learning rate of 0.001, was used when training these neural networks, as well as a binary cross-entropy loss function.

Training some preliminary neural networks for several users in the sample set, it quickly became evident that no one neural network architecture worked best for each user. This makes sense intuitively since users with more varied interests may need more flexible neural networks to accurately predict their interests. Several different neural network architectures

were therefore designed to be trained for each user in the sample set. Initially training a neural network consisting of no hidden layers on a user, five additional neural networks would be trained on him/her, adding a hidden layer to increase the model’s variability each time. When constructing these model frameworks, no concrete set of guidelines was found on how to determine the number of nodes to use in a given layer. Several rules of thumb were offered by Heaton [52], such as setting the number of hidden units to be between the number of input and output units, setting the number of hidden units to $2/3$ the number of input units plus the number of output units, and setting the number of hidden units to less than double the number of input units. For simplicity it was decided to set the number of nodes in a given hidden layer to be half the number of nodes in the previous layer floored. Therefore, the first added hidden layer would have 50 nodes, while the second would have 25, followed by 12, 6, and 3. Due to its prevalent use in the hidden nodes in plenty of deep learning architectures, all nodes in the hidden layers had a ReLU activation function [4]. Each of these frameworks can be viewed in Appendix B. The framework of the final and most complex deep learning architecture used in this study can be seen in Fig. 9.

Figure 9: Framework of the most complex neural network architecture used in this work. Out of six neural network frameworks, the first contained no hidden layers, and each subsequent framework contained 1 additional hidden layer with half the number of nodes found in the previous layer floored until the above framework was reached.

3.4 Recommender System Framework

With deep learning being used to identify which discussions a user would be interested in, an attempt was made to use these models as recommender systems to recommend a small subset of discussions identified as being of interest to a user, for that user. As previously mentioned, each of these neural networks contained a sigmoid output unit, which outputted a probability that a given discussion would be of interest to the corresponding user. To observe how these models would work as recommender systems, two different approaches were utilized to recommend discussions to users. For the first approach, 20 discussions in each user's test set that had the highest determined probability of being of interest to that given user were recommended to him/her. For the second approach, 20 discussions in each user's test set were randomly selected as recommendations for that user using their determined probabilities as selection weights. The results of the recommendations made by

these two approaches would then be analyzed.

3.5 Initial Experiment: Using the F2-Score to Evaluate Performance

The initial experiment conducted for this research focused on building a classifier to identify which discussions a given user would be interested in using the F2-Score as a performance metric. This approach was detailed and discussed in Buhagiar et al. [53].

Running some preliminary experiments training these models, it was soon found that the initial random weights of a given neural network had a sizeable influence on a model's performance in a short number of epochs. It was therefore decided that each user would not only have all six neural network architectures trained on him/her, but this would be done 10 times using different random weight initializations each time. Since this resulted in essentially training 60 different models for each user, each model was only trained for a maximum of 10 epochs to reduce computation time.

When training these neural networks, a decision needed to be made on how many training examples the optimization algorithm would take into account when optimizing the binary cross-entropy loss function at each iteration. Optimization algorithms that process all samples in the training set simultaneously are referred to as deterministic or batch methods, while those that process only one training set at a time are referred to as stochastic or online learning methods [34]. Minibatch optimization methods use a number in-between, and are used much more often [34]. Running several simple tests, it was found that training with a larger minibatch size led to the models achieving better results for the limited number of epochs allocated for training. Therefore, the decision was made to use batch learning when training these neural networks.

To further save computation time when training these models, the F2-Score achieved by a user's given model would be calculated after every epoch of training. The training procedure would only continue if the neural network had a validation F2-Score greater than

the score achieved in the previous epoch subtracted by 0.01. If this criterion was not met, training would cease. This would continue until the maximum of 10 epochs was reached. When conducting this process, the incorrect assumption was made that using only a small sample of a user’s validation set for validation would noticeably reduce computation time. This time reduction however turned out to be negligible, and this error was corrected in the following experiment detailed in Chapter 3.6. However, for this experiment, only 1% of a user’s validation set, randomly sampled, was used to validate their models. Out of all the neural network models trained for a given user, the one that scored the highest F2-Score on the user’s corresponding validation set would be the one selected for that user. To perhaps add further clarity to the approach conducted in this experiment, an outline of the training procedure conducted for a given user in the sample set can be seen in Algorithm 1.

Algorithm 1: Deep learning training procedure for a given user in the sample set for the initial experiment.

Result: User's deep learning model to predict his/her discussion interests

```

1  Compile the training data particularly catered for the user, with target values appended to each discussion vector and
   positive instances oversampled in the manner discussed in Chapter 3.1;
2  Similarly compile the validation data particularly catered for the user, randomly selecting only 1% of the oversampled
   validation set to be used for validation;
3  Prepare neural network architectures, as discussed in Chapter 3.3, to predict this user's interests using Keras [50], each
   having 100 input nodes, a sigmoid output unit, a binary cross-entropy loss function, and the Adam Optimizer [36]
   with its default parameters and set to Neural_Network_Architectures;
4  for  $i \leftarrow 0$  to 10 do
5      foreach  $Neural\_Network \in Neural\_Network\_Architectures$  do
6          Initialize bias terms to zero and randomly assign values to the weights using the Xavier Uniform Initializer
           for  $Neural\_Network$  [51];
7           $Baseline\_F2\_Score \leftarrow 0$ ;
8           $Validation\_F2\_Score \leftarrow 0$ ;
9           $Epoch \leftarrow 0$ ;
10         while  $Epoch < 10$  do
11             Train the  $Neural\_Network$  for 1 epoch using batch learning on the user's given training data;
12             Calculate the trained model's F2-Score on the user's corresponding validation set and set to
                 $Validation\_F2\_Score$ ;
13             if  $Validation\_F2\_Score < (Baseline\_F2\_Score - 0.01)$  then
14                 Break;
15             else
16                  $Baseline\_F2\_Score \leftarrow Validation\_F2\_Score$  ;
17             end
18              $Epoch \leftarrow Epoch + 1$ ;
19             Save  $Neural\_Network$  model;
20         end
21         Load saved  $Neural\_Network$  model;
22         Calculate final validation F2-Score;
23         Save  $Neural\_Network$  model, keeping note of it's final validation F2-Score;
24     end
25 end
26 Identify the user's interest predictor as being the model which scored the highest validation F2-Score;

```

Repeating the above training procedure for every user in the sample set, these selected models were then tested on their corresponding user's test set. After analyzing the results achieved by these user interest predictors, discussed in Chapter 4.1, the above methodology

was repeated using deep learning but with discussions represented as topic vectors using an LDA model trained on the training data using only 50 topics. A list of the ten most popular tokens incorporating each of these 50 topics can be viewed in Appendix C. This was done to see the influence the number of LDA model topics had on the results. When constructing the neural networks used for this approach with 50 topics, all neural networks still had 100 input nodes, but with half of them having input values set to zero.

Further validation of this approach was done by training a machine learning classifier for these users' interests using a Gaussian naive Bayes classifier and support vector machines instead. This was done in order to compare the results achieved by the deep learning approach with simpler machine learning methods. Both machine learning classifiers were implemented using the Python machine learning package Scikit-Learn [31]. Since naive Bayes classifiers cannot be tuned except with the addition of the prior class probabilities [31], the procedure for training a naive Bayes classifier for a given user was relatively straightforward, outlined in Algorithm 2.

Algorithm 2: Gaussian naive Bayes classifier training procedure for a given user in the sample set.

Result: User's Gaussian naive Bayes classifier to predict his/her discussion interests

- 1 Compile the training data particularly catered for the user, with target values appended to each discussion vector and positive instances oversampled in the manner discussed in Chapter 3.1;
- 2 Construct a Gaussian naive Bayes classifier for the user using Scikit-Learn [31];
- 3 Train the user's naive Bayes classifier on its training data;
- 4 Save the model;

In contrast to naive Bayes classifiers, SVMs could be fine-tuned similarly to neural networks to hopefully improve upon their performance. When training the SVMs for a given user, though all had an RBF kernel, multiple models were trained for said user using different values for the tuning parameter C , either 0.001, 0.01, 0.1, 1, 10, or 100. Each of these configurations were trained ten different times using different random number seeds for a user, for a maximum of 10 iterations (the number of steps in the solver of the Scikit-Learn

library’s implementation of the SVM to try to reach convergence [31]). This was done so as to be similar to how the deep learning models were trained for a maximum of 10 epochs. The SVM training procedure for a given user can be viewed in Algorithm 3.

Algorithm 3: SVM training procedure for a given user in the sample set for the initial experiment.

```

Result: User’s SVM to predict his/her discussion interests
1  Compile the training data particularly catered for the user, with target values appended to each discussion vector and
   positive instances oversampled in the manner discussed in Chapter 3.1;
2  Similarly compile the validation data particularly catered for the user, randomly selecting only 1% of the oversampled
   validation set to be used for validation;
3   $Tuning\_Parameters \leftarrow [0.001, 0.01, 0.1, 1, 10, 100]$ ;
4  for  $i \leftarrow 0$  to 10 do
5      foreach  $C \in Tuning\_Parameters$  do
6          Construct an SVM using Scikit-Learn [31] with the tuning parameter being set to  $C$ ;
7          Train the user’s SVM for 10 iterations;
8          Calculate final validation F2-Score;
9          Save the model, keeping note of it’s final validation F2-Score;
10     end
11 end
12 Identify the user’s interest predictor as being the model which scored the highest validation F2-Score;

```

After analyzing the results achieved by the deep learning approach used in this experiment, it was found that they were incompatible with the objectives this work seeks to achieve. Due to this revelation, the recommender system extension was not implemented for these deep learning models. This discovery and rumination is discussed in detail in Chapter 4.1. Accordingly, a second experiment was conducted to identify user interests, in the hopes of surpassing this shortcoming, using accuracy as a performance metric instead of the F2-Score. This was done in the hope of developing a model that could both accurately predict the interests of a given user, as well as serve as a recommender system for him/her.

3.6 Final Experiment: Using Accuracy to Evaluate Performance

Due to the unsuitability of the results found in the approach outlined in Chapter 3.5 for the intentions of this study, further discussed in Chapter 4.1, a second experiment was conducted to identify user interests. This experiment followed a similar methodology to that outlined in Chapter 3.5. In this experiment however, for each user only one random weight initialization was used for each model framework, and training lasted for a maximum of 30 epochs instead of 10. The entire validation set was also used for each user as opposed to 1% of it randomly sampled. The model selected for each user in this case was also deemed as the one which had the highest validation accuracy score, not F2-Score. An outline of this training procedure for a given user in the sample set can be viewed in Algorithm 4.

Algorithm 4: Deep learning training procedure for a given user in the sample set for the final experiment.

Result: User's deep learning model to predict his/her discussion interests

- 1 Compile the training data particularly catered for the user, with target values appended to each discussion vector and positive instances oversampled in the manner discussed in Chapter 3.1;
- 2 Similarly compile the validation data particularly catered for the user;
- 3 Prepare neural network architectures, as discussed in Chapter 3.3, to predict this user's interests using Keras [50], each having 100 input nodes, a sigmoid output unit, a binary cross-entropy loss function, and the Adam Optimizer [36] with its default parameters and set to *Neural_Network_Architectures*;
- 4 **foreach** *Neural_Network* $\in$ *Neural_Network_Architectures* **do**
 - 5 Initialize bias terms to zero and randomly assign values to the weights using the Xavier Uniform Initializer for *Neural_Network* [51];
 - 6 *Baseline_Accuracy* $\leftarrow$ 0;
 - 7 *Validation_Accuracy* $\leftarrow$ 0;
 - 8 *Epoch* $\leftarrow$ 0;
 - 9 **while** *Epoch* < 30 **do**
 - 10 Train the *Neural_Network* for 1 epoch using batch learning on the user's given training data;
 - 11 Calculate the trained model's accuracy on the user's corresponding validation set and set to *Validation_Accuracy*;
 - 12 **if** *Validation_Accuracy* < (*Baseline_Accuracy* - 0.01) **then**
 - 13 Break;
 - 14 **else**
 - 15 *Baseline_Accuracy* $\leftarrow$ *Validation_Accuracy* ;
 - 16 **end**
 - 17 *Epoch* $\leftarrow$ *Epoch* + 1;
 - 18 Save *Neural_Network* model;
 - 19 **end**
 - 20 Load saved *Neural_Network* model;
 - 21 Calculate final validation accuracy;
 - 22 Save *Neural_Network* model, keeping note of it's final validation accuracy;
- 23 **end**
- 24 Identify the user's interest predictor as being the model which scored the highest validation accuracy;

This approach was repeated using a 25-Topic and 50-Topic LDA model, to see the impact the number of topics had on the results. A list of the ten most popular tokens incorporating each of these 25 topics can be viewed in Appendix D. The neural networks constructed for both these implementations still had 100 input nodes, but with 75 and 50 of them having input values set to zero for the 25-Topic and 50-Topic implementation respectively. Naive

Bayes classifiers were again trained on the sample set of users to compare this deep learning approach to a simpler machine learning method, using the same procedure as outlined in Algorithm 2, but with accuracy of course being used instead of F2-Score as the performance metric. SVMs were also trained on the sample set of users. However, unlike in the previous experiment, multiple different instances of an SVM were not trained for each user, with the highest scoring model being selected. Instead, to reduce the amount of time spent tuning an approach secondary to the primary experiment, just the default SVM framework defined in Scikit-learn with a set random seed was used [31]. This was trained for 30 iterations (similar to how the deep learning models were trained for a maximum of 30 epochs). This training procedure for a given user is outlined in Algorithm 5.

Algorithm 5: SVM training procedure for a given user in the sample set for the final experiment.

Result: User’s SVM to predict his/her discussion interests

- 1 Compile the training data particularly catered for the user, with target values appended to each discussion vector and positive instances oversampled in the manner discussed in Chapter 3.1;
- 2 Construct an SVM using Scikit-Learn [31] with default parameters;
- 3 Train the user’s SVM for 30 iterations;
- 4 Save the model;

After validating and testing the deep learning models created in this experiment, recommender systems, as discussed in Chapter 3.4, were implemented as extensions to these models. This enabled these deep learning models to also yield recommendations for their given user.

3.7 Computer Configurations

Two machines were used in these experiments, referred to as *Computer 1* and *Computer 2*, with their technical specifications being shown in Table 2. *Computer 1* was used to train the LDA models and train the naive Bayes classifiers and SVMs in the initial experiment. *Computer 2* was used for all other experimentation. A 4 TB WD My Passport portable

storage drive was used to to store all data and models.

Table 2: Technical specifications of computers used in this study.

Technical Specification	Computer 1	Computer 2
Operating System	Ubuntu 16.04 LTS	Ubuntu 16.04 LTS
Memory	13.1 GiB	7.5 GiB
Processor	Intel®Core™i7-4790S CPU @ 3.20GHz x 4	Intel®Core™i5-4330M CPU @ 2.80GHz x 4
OS-Type	64-bit	64-bit
Disk	19.9 GB	212.9 GB

4 Experimental Results and Analysis

4.1 Initial Experiment Results and Analysis

Selecting the interest predictor models for each user based on their validation F2-Score as outlined in Chapter 3.5, these neural networks were subsequently tested on each corresponding user’s previously unseen test set discussion threads. A plot of the test F2-Scores for these models in relation to their neural network architecture, indicated by the number of hidden layers in their framework, can be viewed in Fig. 10. As one can see in Fig. 10, an average F2-Score of 82.5% and median of 83.2% was achieved by each user’s neural network on their corresponding test set.

Figure 10: F2-Scores obtained for each of the 30 sample users’ test sets in relation to their corresponding selected neural network architectures, represented by the number of hidden layers they contained.

The frequency of neural network architectures selected for each user can also be viewed in Fig. 11. As one can see in Fig 11, all sample users required a neural network consisting of at least two hidden layers to achieve their best results, with each deeper model architecture having more and more users.

Figure 11: Bar chart of selected neural network architectures for the sample set of users, with the neural network architecture being represented by the number of hidden layers it contained. Due to no users having a selected model with 0 or 1 hidden layers, 0 and 1 are not present on the x-axis.

The evolution of the validation F2-Score achieved by each user’s selected model during each epoch of its training procedure can be seen in Fig. 12. Interestingly, one can promptly see upon viewing Fig. 12 that most neural networks immediately achieved a high validation F2-Score. Models that didn’t immediately achieve a high F2-Score did so after only a few epochs. This was presumably due to the initial weights of these neural network models

happening to be desirable. After a rapid increase in the validation F2-Score, this value stayed effectively the same for the rest of the training procedure. This indicates that a maximum of 10 epochs may have been more than enough for these neural networks to adequately model user interests.

Figure 12: Validation F2-Score for each sample user’s selected model at each epoch of training.

As mentioned in Chapter 3.5, this experiment was repeated using deep learning models with a 50-Topic LDA model, as well as naive Bayes classifiers, and SVMs to model user interests. The Kernel Density Estimation (KDE) plots for the results obtained by each of these four approaches are shown in Fig. 13. As one can see viewing Fig. 13, the deep learning approaches performed much better than the simpler machine learning algorithms, though the impact that the number of topics in the LDA model had on these model results seems

to be inconclusive.

Figure 13: Kernel Density Estimation (KDE) plots of F2-Score test results obtained for the sample set of users using several different methods of classification. A KDE plot estimates the density of given data based on a set function or kernel, which was Gaussian for this visualization. *ANN* is an abbreviation for artificial neural network, *NB* is an abbreviation for the naive Bayes classifier, and *SVM* is an abbreviation for support vector machine.

One can view the comparison between the average test F2-Score with the average test precision and test recall scores for the sample set of users in Table 3. As one can see in Table 3, the relatively high average F2-Score achieved from this experiment was due to the neural networks having a mean of almost 96% recall and 54% precision on their corresponding test data.

One can gather further perspective on the type of classifications made by these models by viewing confusion matrices of their predictions. A confusion matrix is a table that shows how

Table 3: Mean Precision, F2-Score, and Recall values predicted by the sample users’ neural networks on their corresponding test sets.

	Precision	F2-Score	Recall
Average	0.542	0.825	0.956

many observations were predicted to be positive or negative in relation to what their actual labels were [6]. Viewing the confusion matrix for every user in the sample set in Table 4, it is quite clear that the high F2-Score and recall values for each user’s model was due to these models making predominantly positive predictions for all the test data instances. Upon further inspection of Table 4, one can see that several users even had nothing but positive predictions for their test set.

In Buhagiar et al. [53], the work of Yano et al. [28] was compared with these results in order to put these results into perspective. As mentioned in Chapter 2.4, Yano et al. [28] similarly produced an approach to predict whether or not a user would comment on a discussion post, achieving 37% precision on their test set. In Buhagiar et al. [53] it was concluded that though the precision score of 54% achieved by this experiment was not ideal, it did seem to be an improvement in that respect. However, on further analysis this conclusion may not be warranted. The precision score calculated by Yano et al. [28] was not on what discussions a given user would contribute to, but on what users would contribute to a given discussion. Therefore, the results found by Yano et al. [28] was achieved in a different context, and inappropriate for comparison with the results achieved in this work.

Regardless, the overwhelming number of positive predictions made by these deep learning models begs serious consideration. One of the primary objectives of this work was to predict user interests in an online discussion forum. Using this approach, it seems that this goal was not achieved, due to the large number of misclassification errors, false positives though they may be. Therefore, the obvious question raised from this conclusion then was where did the error in this approach stem from so it could be rectified. One possible oversight was in the

Table 4: Normalized confusion matrices for the test set of each user in the sample set. The values in a given user’s confusion matrix were normalized by dividing all 4 values by their sum. The mean confusion matrix was calculated as simply the average values of these normalized confusion matrices.

		<i>Actual Positive</i>	<i>Actual Negative</i>
User	<i>Predicted Positive</i>	0.500	0.500
A	<i>Predicted Negative</i>	0.000	0.000
User	<i>Predicted Positive</i>	0.479	0.470
B	<i>Predicted Negative</i>	0.021	0.029
User	<i>Predicted Positive</i>	0.500	0.457
C	<i>Predicted Negative</i>	0.000	0.043
User	<i>Predicted Positive</i>	0.455	0.393
D	<i>Predicted Negative</i>	0.045	0.107
User	<i>Predicted Positive</i>	0.478	0.463
E	<i>Predicted Negative</i>	0.022	0.036
User	<i>Predicted Positive</i>	0.439	0.454
F	<i>Predicted Negative</i>	0.061	0.046
User	<i>Predicted Positive</i>	0.471	0.308
G	<i>Predicted Negative</i>	0.029	0.192
User	<i>Predicted Positive</i>	0.458	0.479
H	<i>Predicted Negative</i>	0.042	0.021
User	<i>Predicted Positive</i>	0.490	0.480
I	<i>Predicted Negative</i>	0.010	0.020
User	<i>Predicted Positive</i>	0.445	0.418
J	<i>Predicted Negative</i>	0.056	0.082
User	<i>Predicted Positive</i>	0.500	0.477
K	<i>Predicted Negative</i>	0.000	0.023
User	<i>Predicted Positive</i>	0.453	0.105
L	<i>Predicted Negative</i>	0.047	0.395
User	<i>Predicted Positive</i>	0.493	0.433
M	<i>Predicted Negative</i>	0.007	0.067
User	<i>Predicted Positive</i>	0.480	0.021
N	<i>Predicted Negative</i>	0.020	0.479
User	<i>Predicted Positive</i>	0.490	0.428
O	<i>Predicted Negative</i>	0.010	0.072
User	<i>Predicted Positive</i>	0.500	0.500
P	<i>Predicted Negative</i>	0.000	0.000
User	<i>Predicted Positive</i>	0.500	0.496
Q	<i>Predicted Negative</i>	0.000	0.004
User	<i>Predicted Positive</i>	0.500	0.500
R	<i>Predicted Negative</i>	0.000	0.000
User	<i>Predicted Positive</i>	0.500	0.449
S	<i>Predicted Negative</i>	0.000	0.051
User	<i>Predicted Positive</i>	0.462	0.445
T	<i>Predicted Negative</i>	0.038	0.055
User	<i>Predicted Positive</i>	0.480	0.473
U	<i>Predicted Negative</i>	0.020	0.030
User	<i>Predicted Positive</i>	0.448	0.450
V	<i>Predicted Negative</i>	0.052	0.050
User	<i>Predicted Positive</i>	0.443	0.368
W	<i>Predicted Negative</i>	0.057	0.132
User	<i>Predicted Positive</i>	0.474	0.471
X	<i>Predicted Negative</i>	0.026	0.029
User	<i>Predicted Positive</i>	0.493	0.439
Y	<i>Predicted Negative</i>	0.007	0.060
User	<i>Predicted Positive</i>	0.482	0.401
Z	<i>Predicted Negative</i>	0.019	0.099
User	<i>Predicted Positive</i>	0.468	0.480
Θ	<i>Predicted Negative</i>	0.032	0.020
User	<i>Predicted Positive</i>	0.484	0.470
Ψ	<i>Predicted Negative</i>	0.016	0.030
User	<i>Predicted Positive</i>	0.472	0.366
Ω	<i>Predicted Negative</i>	0.028	0.134
User	<i>Predicted Positive</i>	0.500	0.481
Φ	<i>Predicted Negative</i>	0.000	0.019
Mean	<i>Predicted Positive</i>	0.478	0.422
	<i>Predicted Negative</i>	0.022	0.077

oversampling of the validation and test sets of users. Oversampling the positive instances in the training set was absolutely necessary in order for these models to learn what constituted a positive instance. However, by not oversampling the positive instances in the validation and test sets, these models could instead be validated and tested on a “real-life” scenario to see how they would perform in the realistic situation of having a large number of discussions in which only a small number would be contributed to by the user. Though there is certainly sense to this approach, one shortcoming of this methodology is that even if models were developed for it that perfectly identified which discussion a user would contribute to, the model would miss out on the vast array of content available in the online forum which are potentially of interest to the user but not being classified as such by the model. Therefore, oversampling the positive instances with added noise in the validation set to validate how accurately these models were capturing users interests was appropriate, as was oversampling the positive instances in the test set to keep the ratio of positive and negatives instances in the validation and test sets consistent. However, the evaluation metric used to validate and test these models may not have been appropriate.

The rationale for using the F2-Score as a performance metric in the first place was to develop models that risked having more false positive errors to try and mitigate the number of false negative errors, but not to the extreme disparity that is appearing in these results. At first glance it would therefore seem that the obvious solution is to tweak the β value in the evaluation metric to lower this disparity. However, this begs the question, at what ratio of positive to negative predictions would be satisfactory. Would a ratio of 66% positive predictions to 33% negative predictions be better? Or perhaps 60% to 40%, or 51% to 49% positive to negative predictions would be ideal. Regardless of whichever ratio was deemed satisfactory, it would seem to be a relatively arbitrary decision. Therefore, it was decided that just using accuracy as a performance metric would be a suitable way in which to evaluate these machine learning models, by simply assessing how well a given model was correctly identifying user interests. A new experiment was conducted, similar in approach

to this experiment, using accuracy as the performance metric instead of the F2-Score. The methodology of this second experiment was outlined in Chapter 3.6. With this approach, not only could models be produced that more accurately captured user interests, but these models could also hopefully be useful in serving as a recommender system for a given user. The results and analysis of this approach are discussed in Chapter 4.2.

4.2 Final Experiment Results and Analysis

Running the second experiment which used accuracy as a performance metric, detailed in Chapter 3.5, the accuracy scores obtained by each user’s selected model on their corresponding test set can be seen in Fig. 14. As one can see in Fig. 14, this new approach achieved a mean test accuracy of 66.1% and a median test accuracy of 63.4%.

Figure 14: Accuracy scores obtained for each of the 30 sample users' test sets in relation to their corresponding selected neural network architectures, represented by the number of hidden layers they contained.

The frequency of the selected neural network architectures for the sample set of users can be seen in Fig. 15. As evident by some of these models having only one hidden layer, clearly this approach led to a little more diversity in the model architectures selected for these users.

Figure 15: Bar chart of selected neural network architectures for the sample set of users, with the neural network architecture being represented by the number of hidden layers it contained. Due to no users having a selected model with zero hidden layers, zero is not present on the x-axis.

Viewing the confusion matrices obtained from each sample user’s model on their corresponding test set in Table 5, one can see that there is much more balance between the positive and negative predictions made by these models as well, with none outputting all positive predictions like was had in the previous experiment.

The evolution of the validation accuracy score achieved by each user’s selected model during its training procedure can also be seen in Fig. 16. As compared with the results obtained in the previous experiment, one can see that these scores did not plateau almost immediately, still steadily increasing for the entire training procedure. This indicates that training these deep learning models for more epochs could further improve their accuracy

Table 5: Normalized confusion matrices for the test set of each user in the sample set. The values in a given user’s confusion matrix were normalized by dividing all 4 values by their sum. The mean confusion matrix was calculated as simply the average values of these normalized confusion matrices.

		<i>Actual Positive</i>	<i>Actual Negative</i>
User	<i>Predicted Positive</i>	0.016	0.033
A	<i>Predicted Negative</i>	0.484	0.467
User	<i>Predicted Positive</i>	0.221	0.125
B	<i>Predicted Negative</i>	0.279	0.375
User	<i>Predicted Positive</i>	0.292	0.106
C	<i>Predicted Negative</i>	0.208	0.394
User	<i>Predicted Positive</i>	0.355	0.142
D	<i>Predicted Negative</i>	0.145	0.358
User	<i>Predicted Positive</i>	0.246	0.196
E	<i>Predicted Negative</i>	0.254	0.303
User	<i>Predicted Positive</i>	0.167	0.072
F	<i>Predicted Negative</i>	0.333	0.428
User	<i>Predicted Positive</i>	0.391	0.010
G	<i>Predicted Negative</i>	0.109	0.490
User	<i>Predicted Positive</i>	0.268	0.233
H	<i>Predicted Negative</i>	0.232	0.267
User	<i>Predicted Positive</i>	0.363	0.325
I	<i>Predicted Negative</i>	0.137	0.175
User	<i>Predicted Positive</i>	0.206	0.135
J	<i>Predicted Negative</i>	0.294	0.365
User	<i>Predicted Positive</i>	0.268	0.114
K	<i>Predicted Negative</i>	0.232	0.386
User	<i>Predicted Positive</i>	0.453	0.115
L	<i>Predicted Negative</i>	0.047	0.385
User	<i>Predicted Positive</i>	0.368	0.147
M	<i>Predicted Negative</i>	0.132	0.353
User	<i>Predicted Positive</i>	0.480	0.024
N	<i>Predicted Negative</i>	0.020	0.476
User	<i>Predicted Positive</i>	0.382	0.250
O	<i>Predicted Negative</i>	0.118	0.250
User	<i>Predicted Positive</i>	0.354	0.161
P	<i>Predicted Negative</i>	0.146	0.339
User	<i>Predicted Positive</i>	0.253	0.142
Q	<i>Predicted Negative</i>	0.247	0.358
User	<i>Predicted Positive</i>	0.375	0.107
R	<i>Predicted Negative</i>	0.125	0.393
User	<i>Predicted Positive</i>	0.336	0.291
S	<i>Predicted Negative</i>	0.164	0.208
User	<i>Predicted Positive</i>	0.317	0.067
T	<i>Predicted Negative</i>	0.183	0.433
User	<i>Predicted Positive</i>	0.245	0.171
U	<i>Predicted Negative</i>	0.255	0.329
User	<i>Predicted Positive</i>	0.302	0.165
V	<i>Predicted Negative</i>	0.198	0.335
User	<i>Predicted Positive</i>	0.284	0.112
W	<i>Predicted Negative</i>	0.216	0.388
User	<i>Predicted Positive</i>	0.214	0.150
X	<i>Predicted Negative</i>	0.286	0.350
User	<i>Predicted Positive</i>	0.472	0.171
Y	<i>Predicted Negative</i>	0.029	0.329
User	<i>Predicted Positive</i>	0.389	0.063
Z	<i>Predicted Negative</i>	0.111	0.439
User	<i>Predicted Positive</i>	0.253	0.141
Θ	<i>Predicted Negative</i>	0.247	0.358
User	<i>Predicted Positive</i>	0.175	0.096
Ψ	<i>Predicted Negative</i>	0.325	0.403
User	<i>Predicted Positive</i>	0.264	0.110
Ω	<i>Predicted Negative</i>	0.236	0.390
User	<i>Predicted Positive</i>	0.242	0.140
Φ	<i>Predicted Negative</i>	0.258	0.359
Mean	<i>Predicted Positive</i>	0.298	0.137
	<i>Predicted Negative</i>	0.202	0.363

scores.

Figure 16: Validation accuracy score for each user’s selected model at each epoch of training.

A KDE plot comparing the accuracy results obtained by this approach with those achieved with the simpler machine learning approaches, and the 25-Topic and 50-Topic deep learning models can be viewed in Fig. 17. A line graph comparing the accuracy results achieved for each individual user in the sample set using each of these approaches can be found in Fig. 18. As one can see in both Fig. 17 and Fig. 18, the SVM performed noticeably worse than all other methods, similar to the previous experiment, with the user SVMs almost all scoring a validation accuracy of around 50%. The deep learning approaches also achieved generally superior results to the naive Bayes classifier. Surprisingly however, both the 25-Topic and 50-Topic deep learning models generally performed slightly better than the 100-Topic LDA model used throughout this work. This could be possibly due to the deep learning models

using the 25-Topic and 50-Topic LDA models being less complicated than those associated with the 100-Topic LDA model, having less values fed into their input nodes. This could have resulted in the 25-Topic and 50-Topic deep learning models being able to approach convergence faster and seemingly perform better since there was a limit of each deep learning model being trained for a maximum of 30 epochs. This could also just indicate that the deep learning models achieve better results with an LDA model with around 25 to 50 topics. This certainly calls for future analysis. For consistency, the 100-Topic LDA model and the associated deep learning models will continue to be used for the remainder of this study.

Figure 17: Kernel Density Estimation (KDE) plots of test accuracy results obtained for the sample set of users using several different methods of classification. A KDE plot estimates the density of given data based on a set function or kernel, which was Gaussian for this visualization. *ANN* is an abbreviation for artificial neural network, *NB* is an abbreviation for the naive Bayes classifier, and *SVM* is an abbreviation for support vector machine.

Figure 18: Line graph of test accuracy results obtained for each individual user in the sample set using several different methods of classification, sorted by test accuracy score for the 100-Topic deep learning approach in decreasing order. *ANN* is an abbreviation for artificial neural network, *NB* is an abbreviation for the naive Bayes classifier, and *SVM* is an abbreviation for support vector machine.

It is in the view of the author that these results are satisfactory given the context, though these accuracy scores and classification results may seem underwhelming at first glance. Upon some reflection, one can understand that there are many factors unrelated to an online discussion’s themes that can influence whether or not a given user contributes to it. The mood a user is in when considering a discussion and its associated web content must surely have an impact on how they respond to it. A user could possibly also be in a rush and not have time to comment in a discussion they would have liked to contribute to otherwise. In the work of Li et al. [17], they took into consideration whether or not other news articles in their dataset were simply duplicates of each other. Due to the ability of every user on Reddit to contribute content to its platform, it’s not too difficult to imagine that situations

would arise where multiple users would post a link to the same new movie trailer, or post a headline with the same breaking news. Users who contribute to one of these potentially many related discussions may not see the need to contribute to any of the others. Given these constraints on knowing what a user truly may be interested in, and considering that this was done using deep learning, it is the opinion of this author that achieving a mean accuracy of around 66% for these sample users' test sets validates the thesis statement that user interests can be accurately identified in an online discussion forum using deep learning. Due to this being done using a dataset with hundreds of thousands of users and discussions in each of the training, validation, and test sets, these results also substantiate the claim that user interests can be identified in a forum of hundreds of thousands of users and discussion threads.

With these neural networks therefore reasonably modelling user interests, the attempt was made to use these models to make some recommendations for users on discussions for them to contribute. Implementing the two approaches discussed in Chapter 3.4 to recommend content to users, the results of these user recommender systems can be seen in Fig. 19 and Fig. 20.

Figure 19: Bar graph showing the number of users who had a given number of recommendations that matched discussion threads they had actually been seen to contribute to, out of 20 recommendations made using each user’s given test set. The 20 recommendations made to each user in this approach corresponded to the 20 discussion threads in their corresponding test set that had the highest probability of being of interest to them as deemed by their selected interest predictor.

Figure 20: Bar graph showing the number of users who had a given number of recommendations that matched discussion threads they had actually been seen to contribute to, out of 20 recommendations made using each user's given test set. The 20 recommendations made to each user in this approach were discussion threads randomly selected from their corresponding test set, using their user interest probabilities as deemed by their selected interest predictor as selection weights.

Viewing Fig. 19, one can see that this approach yielded poor results for the majority of users in the sample set, recommending discussions for each user that they had not been seen to contribute to. Though this recommendation approach was successful for a few users, the poor results achieved for the majority of users in the sample set is cause for concern. Viewing Fig. 20 on the other hand, one can see that for all users in the sample set around half the discussion threads recommended to them were ones they had actually been seen to contribute to. This recommendation approach of probabilistically recommending discussions using their interest probabilities as selection weights therefore seems to be the more robust

method, and one that will be used going forward.

One can discern further information of the results achieved by this probabilistic recommendation approach by comparing the topic distributions of the recommendations with that of each corresponding user’s interests. Plots comparing the mean topic distribution of all discussions a given user contributed to in their training set with the mean topic distribution of all discussions recommended to them in their corresponding test set can be seen for three users in our sample set, one in each of Fig. 21, Fig. 22, and Fig. 23.

Figure 21: Bar graph comparing the mean topic distribution of all the discussions User C contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User C in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 22: Bar graph comparing the mean topic distribution of all the discussions User Ω contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Ω in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 23: Bar graph comparing the mean topic distribution of all the discussions User H contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User H in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Viewing Fig. 21, Fig. 22, and Fig. 23, one can see immediately how varied the interest of users can be and how varied the themes of discourse of discussions recommended to them are. One can also see that the most prominent topics of interest for a given user tend to be exhibited with a pronounced weight on average in the discussions recommended to them, though this correlation is much more or less pronounced depending on the user. These comparison plots for every single user in the sample set can be viewed in Appendix E.

Despite these results, it is difficult to really evaluate the usefulness of these recommendations without deploying these recommender systems online to actual users and getting their feedback on the recommendations made by these models. Though recommending discussions that a user had been seen to contribute to is comforting, it is difficult to deem the recommendation of a discussion that a user had not contributed to as a failure since these could indeed be in fact discussions that they would have been interested in contributing to had

they been aware of it. Recommending discussions that they end up contributing to anyways may also be not as useful as first thought, since these threads may be popular discussions and/or be associated with very popular web content that users would have been aware of anyway, and would have no need of a recommender system to present it to them. Though seeing similarities between the themes of discussions users have contributed to and the topics of recommended discussions is encouraging, how successful these recommendations are and ways in which they can possibly be refined call for future study. However, it is the opinion of this author that this approach endorses the thesis statement that these deep learning models can be used as the basis for discussion recommender systems for users in an online discussion forum.

5 Conclusion and Future Work

Using a dataset consisting of millions of comments made in an online discussion forum, the themes of discourse of these comments were determined using the probabilistic topic model LDA. This LDA model was further used to represent the discussion threads in which these comments appeared in as topic vectors to showcase the subject matter of these discussions. In order to develop machine learning models that could identify the interests of a given user, deep learning was used to predict which threads a given user would or would not contribute to. Using two primary methodologies in which to accomplish this task, it was found that using the evaluation metric accuracy was an appropriate means in which to assess the quality of these deep learning models. Training several different neural network architectures for each user in a sample set of 30 users, the framework that achieved the highest validation accuracy score was selected to identify the interests for each given user. Comparing the classifications made by these deep learning models with the classifications made by naive Bayes classifiers and SVMs for the users, it was found that this deep learning approach achieved a higher accuracy than both of these approaches. This deep learning approach with a 100-Topic LDA model scored a mean accuracy of 66.1% on test data.

It is the view of the author that these results validate the thesis statement that user interests in an online discussion forum can be identified using deep learning. Due to this study being carried out using a dataset consisting of comments made by hundreds of thousands of users in hundreds of thousands of discussion threads, it is further the author's view that the claim made in the thesis statement that this can be done is warranted. These deep learning models were further used to make discussion recommendations to users in the sample set based on their predicted probability of contribution, finding a probabilistic recommendation method to be the more robust approach. Though the results achieved by this approach are inconclusive, it is the author's view that this endorses the thesis proposition that these deep learning models can be used as the basis for a discussion recommender system for a user in an online forum.

This work leaves the door open for plenty of future work to be conducted. Many facets of this work can be further studied to better understand how they impact and could possibly improve the results achieved in this study. The influence that the LDA model, especially its number of topics, has on this work calls for extensive study, especially since there were indications that lowering the number of topics may improve the accuracy of predicting user interests. Using a different method in which to represent discussion threads altogether could also be implemented, such as Latent Semantic Analysis (LSA) [54] or doc2vec [55]. The NLP techniques used to preprocess the data could also be further refined. To minimize the presence of ambiguous words in the data, perhaps dropping all words that were not nouns would be advisable. In the work of Blei [22], only nouns seemed to be used to train their LDA model, with stemming also being absent. Perhaps following this rationale would achieve better results. Only six neural network architectures were used to model each user in the sample set. Adding further deep learning architectures as well as fine-tuning many of these neural network hyperparameters for a given user would most likely lead to more accurate results, and it would be interesting to study how much these models' performances could be improved. The use of more data could also help to achieve better results, likely making these models more robust. As opposed to labelling discussions in a binary format as either of interest or not of interest for a given user, labelling them using a continuum of degrees of interest based on how many comments they made for that discussion could also perhaps be worth implementing. Other factors, ignored in this work, could be implemented to further improve results, such as taking the scores given to web content and comments in to account, as well as user reputation, the actual web content itself, the subcommunity the web content was submitted to, etc. Though these factors are more specific to features present in Reddit and perhaps not in other online discussion forums, taking these factors into consideration could potentially improve results if the goal is to identify user interests in discussions and recommend some of them to him/her specifically for Reddit. This work also only scratched the surface on how these deep learning models could be used as recommender systems.

Deploying these recommender systems online to real users to evaluate their recommendations would certainly be a justifiable endeavor, as well as studying how to further improve these recommendations so as to always recommend interesting content and discussions to users.

Applying NLP techniques to social media data in the form of user comments in an online discussion forum posed some unique challenges as opposed to its use in more traditional texts such as news articles [56]. As opposed to more traditional textual media, social media data contains text that tends to be written in a more informal, conversational tone, with a higher chance of there being incorrect spelling, punctuation, and grammar, as well as the higher chance of the presence of emoticons and abbreviations [56]. Social media data also tends to be more subjective than traditional texts, as well as exhibiting multiple topics of conversation due to the informal tone of online discussions and the presence of multiple authors of its content [56]. It would definitely be interesting to see what results could be achieved on using the approach of this work on more traditional textual data. Though this work was done in the context of an online discussion forum and consisted of social media data collected from Reddit, not too much additional work is necessary to extend this methodology to other domains. Since discussion threads were just treated as documents in a corpus, this work can be applied to really any domain involving users and a corpus of text, as long as there is some way in which to identify user interest in a given text. Whether or not the results achieved by this approach in a different domain would be better or worse from what was found in this study would be fascinating, and further calls for future work. It is the author's hope that the work presented here not only validates the use of deep learning in predicting contributor interests in the setting of online discourse, but serves as the foundation for further research in this exciting and expanding domain.

A LDA Model Topics: 100-Topic Model

The following are the topics identified by the 100-Topic LDA model on the training corpus of Reddit comments. Each topic is listed with their ten most popular tokens.

Topic	Ten Most Popular Tokens
0	<i>peopl, don, think, becaus, say, just, like, understand, thing, whi</i>
1	<i>american, america, histori, taken, ignor, unfortun, travel, christian, religion, relev</i>
2	<i>true, list, switch, lost, onlin, round, window, server, screen, load</i>
3	<i>anim, imagin, board, bore, cast, shape, comic, wont, held, discov</i>
4	<i>chang, liter, ok, human, climat, land, natur, scienc, shut, affect</i>
5	<i>face, arm, wife, hasn, lucki, hole, blow, finger, monitor, injuri</i>
6	<i>com, reddit, amp, https, comment, www, messag, http, wiki, bot</i>
7	<i>wall, shop, idiot, bag, rich, anywher, na, bitcoin, queen, reject</i>
8	<i>review, discuss, topic, relat, askreddit, special, request, 3a, icon, talent</i>
9	<i>hit, water, notic, crazi, speed, worst, spot, lock, excit, sport</i>
10	<i>com, http, https, www, imgur, jpg, 2017, org, html, png</i>

11 *trade, test, offer, doubt, engin, magic, defens, biggest,*
jesus, armor
 12 *win, lose, okay, dick, loss, tough, won, 300, hope, sister*
 13 *sound, like, order, fit, appar, gain, popular, lmao, brand,*
chat
 14 *thank, price, product, ll, appreci, core, memori, repli,*
advic, help
 15 *post, pleas, messag, subreddit, question, remov, ani,*
rule, moder, action
 16 *like, feel, just, realli, someth, charact, don, think, thing,*
want
 17 *yes, en, org, wikipedia, 32, pm, welcom, downvot, wiki,*
https
 18 *school, late, direct, delet, subject, park, imguralbumbot,*
twitter, messag, https
 19 *tv, forward, nobodi, search, brother, driver, deep, button,*
mom, a@#h\$l
 20 *pay, buy, cost, tax, account, servic, free, expens, paid,*
health
 21 *day, time, week, month, everi, ago, went, coupl, took,*
just
 22 *number, refer, key, heart, plenti, instal, met, breath, im-*
poss, softwar
 23 *rate, clean, critic, averag, competit, complain, stat, film,*
90, amazon
 24 *watch, video, youtub, https, com, www, song, bar, youtu,*
channel

25 *stupid, train, studi, degre, weak, rise, mass, beer,*
growth, porn
 26 *heard, space, piec, ball, appli, club, engag, movement,*
stack, element
 27 *option, music, bike, describ, sentenc, doesnt, vehicl,*
trap, laptop, drunk
 28 *plus, la, le, est, et, il, si, je, les, en*
 29 *dead, step, fail, member, middl, involv, touch, dream,*
street, ahead
 30 *googl, articl, app, communiti, social, access, join, tho,*
connect, media
 31 *guy, damn, saw, like, imo, mix, solid, album, just, band*
 32 *high, level, low, skill, qualiti, higher, size, use, differ,*
lower
 33 *link, sourc, content, imag, note, site, mod, file, post,*
websit
 34 *report, map, random, lie, bug, boy, return, kick, battl,*
odd
 35 *light, hot, air, 000, ya, heavi, materi, planet, sun, dress*
 36 *white, black, event, beat, busi, straight, gay, insan, mur-*
der, secret
 37 *life, care, don, kid, want, famili, live, just, know, becaus*
 38 *season, ad, end, univers, jump, rank, student, frame,*
3rd, danger
 39 *pull, share, sad, ship, wear, hair, weapon, function, lo-*
cat, yep

40 *dure, area, fix, walk, room, away, gun, carri, stage,*
south
 41 *fan, happi, wish, im, vs, score, dumb, ridicul, slow, de-*
serv
 42 *match, episod, blue, self, green, abil, target, ruin, paint,*
besid
 43 *movi, releas, scene, readi, catch, patch, flat, crash,*
tonight, brought
 44 *love, absolut, draw, impress, p#@s, tomorrow, alright,*
press, champion, german
 45 *trump, op, presid, news, elect, russia, fake, republican,*
campaign, obama
 46 *man, die, fart, er, der, min, und, al, das, ist*
 47 *woman, colleg, negat, femal, male, judg, court, comfort,*
genuin, garbag
 48 *ve, talk, friend, ask, just, seen, know, haven, don, time*
 49 *god, que, son, se, em, el, la, en, da, te*
 50 *gonna, cut, field, technic, cop, stone, shift, cook, wild,*
chain
 51 *second, car, drive, pass, confirm, town, burn, lane, cu-*
rious, mile
 52 *question, word, vote, answer, pictur, simpl, ask, mark,*
cap, budget
 53 *exact, damag, deal, glad, dark, rare, organ, anti,*
counter, run
 54 *game, play, team, player, just, like, time, don, becaus,*
onli

55 *babi, ban, blame, pic, machin, fault, ex, mission, wrote,*
pen
 56 *s@#t, suppos, wow, bu#l@h\$t, holi, weren, classic,*
artist, metal, theme
 57 *yea, hang, master, pure, cold, rang, suit, educ, buddi,*
smile
 58 *money, compani, sell, dont, market, spend, save, invest,*
cat, want
 59 *eat, bodi, food, fast, fat, daili, oil, iron, like, just*
 60 *speak, project, safe, languag, english, mistak, boss, writ-*
ten, translat, japanes
 61 *gt, lol, lt, smell, ain, rocket, embarrass, insert, loser,*
heaven
 62 *sign, track, confus, ground, risk, king, legal, floor, split,*
det
 63 *work, job, posit, write, just, block, time, experi, shitti,*
incred
 64 *parti, hous, energi, agreement, deck, lead, accord, pari,*
global, produc
 65 *good, ass, luck, correct, look, pretti, realli, bad, veri,*
perhap
 66 *pc, weight, steam, consol, egg, uniqu, wheel, ps4, lift,*
lean
 67 *favorit, drink, gold, combat, platform, alcohol, sa, elit,*
union, ed
 68 *yeah, oh, wait, hell, just, sleep, grow, zero, right, like*

69 *make, open, close, sens, sure, cover, leav, just, common,*
door

70 *sex, kept, hook, attract, sexual, marri, partner, commu-*
nic, terrorist, ladi

71 *hand, fight, hold, protect, fuel, shock, 1st, fighter, sum-*
mon, weed

72 *sorri, book, figur, anymor, star, ah, gotten, rape, cultur,*
yesterday

73 *age, tank, meme, tire, minor, use, pool, 21, chase, celebr*

74 *cool, miss, dog, send, date, forget, somewher, sent, men-*
tal, ill

75 *wonder, perfect, surpris, seri, knew, goal, stick, exist,*
whi, wast

76 *idea, hear, dude, awesom, phone, weird, listen, explain,*
just, meant

77 *limit, model, potenti, success, inform, physic, proper,*
immedi, cheap, guarante

78 *add, eye, haha, beauti, bet, generat, aw, camera, trans-*
fer, somebodi

79 *girl, hour, women, men, color, road, collect, half, child,*
glass

80 *shoot, pop, lebron, scale, defend, mess, bigger, blood,*
tree, refus

81 *citi, suck, finish, funni, ice, gotta, doctor, upvot, dragon,*
mr

82 *pick, bitch, initi, trigger, disappoint, screw, pair, im-*
pact, solo, exclus

83 *hate, polit, result, remind, user, troll, archiv, click,*
speech, auto

84 *year, old, red, annoy, young, older, bother, new, teach,*
ride

85 *long, power, term, short, replac, time, reach, progress,*
use, research

86 *stori, fun, sub, version, joke, updat, worri, just, skin,*
like

87 *read, nice, art, veri, paper, post, cancer, style, look, re-*
aliti

88 *10, 20, 100, 30, minut, 50, 15, 12, doubl, 40*

89 *box, attent, leg, announc, rais, spoiler, strength, rout,*
use, mini

90 *wrong, https, com, imagesofnetwork, reddit, www, wiki,*
origin, faqandinstruct, learn

91 *build, worth, soon, hero, unit, extra, mode, 60, smart,*
built

92 *control, fine, use, boot, ram, debt, hardwar, cpu, layer,*
gpu

93 *use, class, design, set, data, avail, program, custom,*
code, creat

94 *problem, support, state, issu, govern, major, law, public,*
polic, pro

95 *card, type, amaz, page, info, wtf, net, gather, wizard,*
ring

96 *plan, shot, item, roll, effort, upgrad, 13, lord, closer,*
need

97	<i>kill, attack, chanc, war, death, enemi, use, heal, fli, valu</i>
98	<i>world, countri, china, eu, uk, nation, europ, popul,</i> <i>canada, usa</i>
99	<i>f#\$k, night, pack, nope, armi, just, giant, empti, sake,</i> <i>like</i>

B Neural Network Architectures

The following are visualizations of each neural network architecture used in this work.

Figure 1: Simplest neural network architecture used in this work, with 0 hidden layers.

Figure 2: Neural network architecture used in this work, with 1 hidden layer.

Figure 3: Neural network architecture used in this work, with 2 hidden layers.

Figure 4: Neural network architecture used in this work, with 3 hidden layers.

Figure 5: Neural network architecture used in this work, with 4 hidden layers.

Figure 6: Most complex neural network architecture used in this work, with 5 hidden layers.

C LDA Model Topics: 50-Topic Model

The following are the topics identified by the 50-Topic LDA model on the training corpus of Reddit comments. Each topic is listed with their ten most popular tokens.

Topic	Ten Most Popular Tokens
0	<i>trump, countri, world, vote, chang, parti, climat, support, peopl, american</i>
1	<i>babi, today, pictur, sad, stupid, wors, p@#s, anymor, gone, shame</i>
2	<i>game, play, fun, releas, player, deck, just, new, lmao, time</i>
3	<i>realli, like, feel, end, season, just, cool, watch, enjoy, stori</i>
4	<i>human, dream, death, mark, yep, brain, sick, fart, tast, rape</i>
5	<i>trade, happi, languag, english, speak, draw, hole, join, readi, travel</i>
6	<i>amp, http, com, www, 2017, https, reddit, 32, comment, org</i>
7	<i>add, ad, list, number, site, content, search, id, new, date</i>
8	<i>sub, ban, thread, user, mod, remind, offer, product, result, hate</i>
9	<i>die, bet, pic, der, score, und, das, jack, ist, ich</i>
10	<i>com, https, http, imgur, imag, link, reddit, jpg, messag, amp</i>
11	<i>sound, like, make, music, word, true, listen, song, hear, sens</i>

12 *anim, design, step, art, voic, generat, consol, devic,*
ruin, theme
 13 *dont, water, im, run, pull, pack, forget, board, forgot,*
camera
 14 *thank, ll, lol, yeah, tri, sorri, dude, help, guess, check*
 15 *post, pleas, question, messag, ani, subreddit, rule, re-*
mov, moder, contact
 16 *level, charact, idea, gold, collect, impress, tier, suit,*
raid, quest
 17 *yes, space, size, park, 90, wonder, perfect, street, fish,*
safe
 18 *nice, exact, dead, ice, walk, door, batteri, brand, hide,*
bear
 19 *miss, amaz, test, drug, hair, cat, model, smoke, doctor,*
medic
 20 *problem, light, bodi, caus, drive, face, cut, hand, area,*
head
 21 *10, 20, damn, hour, 30, minut, wait, 15, week, 50*
 22 *use, car, work, phone, set, updat, app, just, need, win-*
dow
 23 *hot, color, blue, hous, green, clean, plant, b\$l@s#@#t,*
rich, shut
 24 *power, high, switch, count, control, low, fast, speed, bar,*
core
 25 *team, player, win, fan, best, final, ball, leagu, lose, play*
 26 *la, le, ya, est, et, cop, il, je, en, ca*

27 *land, ride, paint, bike, command, wild, buddi, shoe,*
john, flat
 28 *https, wiki, com, imagesofnetwork, www, reddit, wrong,*
stop, faqandinstruct, origin
 29 *state, right, war, peopl, law, group, govern, public, forc,*
legal
 30 *gt, card, net, http, lt, page, type, wtf, credit, gather*
 31 *money, buy, pay, price, cost, sell, make, market, com-*
pani, free
 32 *hit, damag, shot, attack, just, kill, skill, weapon, gun,*
tank
 33 *com, https, www, comment, reddit, watch, video, youtub,*
http, googl
 34 *use, hero, just, onli, dog, need, unit, make, mode, block*
 35 *ok, meme, kill, upvot, nah, er, men, det, femal, glass*
 36 *version, beat, confirm, special, photo, lock, pop, leg,*
appl, giant
 37 *don, peopl, just, know, like, think, want, becaus, say,*
make
 38 *read, book, write, articl, uk, stream, paper, patch, his-*
tori, note
 39 *f\$@k, wear, twitter, yea, mad, status, shitti, short, hi-*
lari, doesnt
 40 *build, red, item, box, map, pick, event, rank, wall, spot*
 41 *good, look, like, pretti, sure, bad, thought, just, hope,*
realli
 42 *love, s@#t, man, oh, god, white, black, haha, beauti, eye*

43	<i>movi, fix, break, bug, heart, welcom, report, battl, annoy,</i>
	<i>broken</i>
44	<i>did, guy, kid, didn, gonna, girl, wow, ass, awesom, sur-</i>
	<i>pris</i>
45	<i>op, news, said, key, lie, fake, evid, email, russian, claim</i>
46	<i>differ, veri, point, base, thing, think, way, ani, exampl,</i>
	<i>certain</i>
47	<i>year, work, live, old, job, school, life, month, time, age</i>
48	<i>day, ve, time, just, friend, befor, got, week, everi, seen</i>
49	<i>que, eu, se, na, em, la, el, en, da, um</i>

D LDA Model Topics: 25-Topic Model

The following are the topics identified by the 25-Topic LDA model on the training corpus of Reddit comments. Each topic is listed with their ten most popular tokens.

Topic	Ten Most Popular Tokens
0	<i>trump, countri, gt, world, vote, chang, state, support, climat, peopl</i>
1	<i>peopl, say, believ, white, live, becaus, whi, law, right, gt</i>
2	<i>game, play, player, team, win, match, fun, deck, best, leagu</i>
3	<i>gt, love, movi, read, charact, stori, fan, book, season, like</i>
4	<i>f@\$k, s#@t, damn, ass, god, fight, gonna, holi, hell, war</i>
5	<i>red, light, use, blue, black, unit, spell, magic, build, box</i>
6	<i>com, https, amp, reddit, comment, www, messag, np, http, link</i>
7	<i>watch, video, https, youtub, com, www, music, song, org, en</i>
8	<i>use, just, work, new, set, onli, need, phone, item, updat</i>
9	<i>la, que, en, se, le, est, youtu, eu, lmao, et</i>
10	<i>http, com, https, www, imgur, card, jpg, imag, news, net</i>
11	<i>yes, ban, true, sure, ok, credit, remind, hate, troll, re-port</i>
12	<i>just, don, like, think, know, peopl, realli, want, make, thing</i>
13	<i>water, air, heat, bag, nope, pm, el, ship, wait, fli</i>
14	<i>thank, lol, oh, yeah, ll, man, nice, great, awesom, hope</i>

15 *post, pleas, question, messag, ani, subreddit, rule, re-*
mov, contact, moder

16 *like, look, good, sound, pretti, just, realli, littl, bit, pictur*

17 *https, com, www, reddit, imagesofnetwork, wiki,*
faqandinstruct, post, 2017, origin

18 *damag, hit, kill, shot, use, attack, weapon, tank, enemii,*
onli

19 *time, just, friend, got, kid, didn, day, did, befor, went*

20 *differ, veri, use, make, point, work, way, ani, think,*
time

21 *year, day, 10, week, month, hour, 20, ago, time, 30*

22 *money, pay, buy, car, price, cost, compani, job, sell,*
work

23 *life, women, girl, men, guy, sex, man, woman, like, per-*
son

24 *sport, twitter, status, core, bar, tbh, bet, boy, champion,*
german

E User Recommendations Topic Comparison

The following are plots comparing the mean topic distribution for the discussions that a user had contributed to in the training set with the the mean topic distribution for the discussions recommended to that user in their given test set, for every user in the sample set.

Figure 7: Bar graph comparing the mean topic distribution of all the discussions User A contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User A in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 8: Bar graph comparing the mean topic distribution of all the discussions User B contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User B in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 9: Bar graph comparing the mean topic distribution of all the discussions User C contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User C in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 10: Bar graph comparing the mean topic distribution of all the discussions User D contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User D in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 11: Bar graph comparing the mean topic distribution of all the discussions User E contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User E in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 12: Bar graph comparing the mean topic distribution of all the discussions User F contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User F in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 13: Bar graph comparing the mean topic distribution of all the discussions User G contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User G in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 14: Bar graph comparing the mean topic distribution of all the discussions User H contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User H in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 15: Bar graph comparing the mean topic distribution of all the discussions User I contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User I in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 16: Bar graph comparing the mean topic distribution of all the discussions User J contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User J in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 17: Bar graph comparing the mean topic distribution of all the discussions User K contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User K in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 18: Bar graph comparing the mean topic distribution of all the discussions User L contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User L in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 19: Bar graph comparing the mean topic distribution of all the discussions User M contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User M in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 20: Bar graph comparing the mean topic distribution of all the discussions User N contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User N in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 21: Bar graph comparing the mean topic distribution of all the discussions User O contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User O in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 22: Bar graph comparing the mean topic distribution of all the discussions User P contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User P in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 23: Bar graph comparing the mean topic distribution of all the discussions User Q contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Q in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 24: Bar graph comparing the mean topic distribution of all the discussions User R contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User R in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 25: Bar graph comparing the mean topic distribution of all the discussions User S contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User S in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 26: Bar graph comparing the mean topic distribution of all the discussions User T contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User T in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 27: Bar graph comparing the mean topic distribution of all the discussions User U contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User U in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 28: Bar graph comparing the mean topic distribution of all the discussions User V contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User V in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 29: Bar graph comparing the mean topic distribution of all the discussions User W contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User W in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 30: Bar graph comparing the mean topic distribution of all the discussions User X contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User X in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 31: Bar graph comparing the mean topic distribution of all the discussions User Y contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Y in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 32: Bar graph comparing the mean topic distribution of all the discussions User Z contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Z in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 33: Bar graph comparing the mean topic distribution of all the discussions User Θ contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Θ in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 34: Bar graph comparing the mean topic distribution of all the discussions User Ψ contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Ψ in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 35: Bar graph comparing the mean topic distribution of all the discussions User Ω contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Ω in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Figure 36: Bar graph comparing the mean topic distribution of all the discussions User Φ contributed to in the training set (labelled as *Mean User Interests*) with the mean topic distribution of all the discussions recommended to User Φ in their given test set (labelled as *Mean Recommendations*) sorted by *Mean User Interests* topic weight in decreasing order.

Bibliography

- [1] reddit: the front page of the internet. <https://www.reddit.com/>, . Accessed: 2018-06-02.
- [2] reddit.com: api documentation. <https://www.reddit.com/dev/api/>. Accessed: 2018-05-29.
- [3] Jason Baumgartner. pushshift.io - Learn about Big Data and Social Media Ingest and Analysis. <https://pushshift.io/>. Accessed: 2017-08-28.
- [4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.
- [5] Jake VanderPlas. In-Depth: Support Vector Machines | Python Data Science Handbook. <https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html>. Accessed: 2018-08-24.
- [6] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*, pages 337–355. Springer Publishing Company, Incorporated, 2014. ISBN 1461471370, 9781461471370.
- [7] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*, page 307. Cambridge University Press, New York, NY, USA, 2011. ISBN 1107015359, 9781107015357.
- [8] Homepage - Reddit. <https://www.redditinc.com/>, . Accessed: 2018-05-16.
- [9] faq - reddit.com. <https://www.reddit.com/wiki/faq>, . Accessed: 2016-12-31.
- [10] faq - help. <https://www.reddit.com/r/help/wiki/faq>. Accessed: 2017-01-03.
- [11] listofsubreddits - ListOfSubreddits. <https://www.reddit.com/r/ListOfSubreddits/wiki/listofsubreddits>. Accessed: 2016-12-31.

- [12] Reddit in 2015. <https://redditblog.com/2015/12/31/reddit-in-2015/>. Accessed: 2017-06-09.
- [13] Yalin Baştanlar and Mustafa Özuysal. Introduction to machine learning. In *miRNomics: MicroRNA Biology and Computational Analysis*, pages 105–128. Springer, 2014.
- [14] Carlos Castro-Herrera, Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In *International Requirements Engineering, 2008. RE’08. 16th IEEE*, pages 165–168. IEEE, 2008.
- [15] Fabian Abel, Ig Ibert Bittencourt, Evandro Costa, Nicola Henze, Daniel Krause, and Julita Vassileva. Recommendations in online discussion forums for e-learning systems. *IEEE transactions on learning technologies*, 3(2):165–176, 2010.
- [16] Diyi Yang, Mario Piergallini, Iris Howley, and Carolyn Rose. Forum thread recommendation for massive open online courses. In *Educational Data Mining 2014*, 2014.
- [17] Qing Li, Jia Wang, Yuanzhu Peter Chen, and Zhangxi Lin. User comments for news recommendation in forum-based social media. *Information Sciences*, 180(24):4929–4939, 2010.
- [18] Andrew Messenger and Jon Whittle. Recommendations based on user-generated comments in social media. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 505–508. IEEE, 2011.
- [19] Robert M Bell, Yehuda Koren, and Chris Volinsky. All together now: A perspective on the Netflix prize. *Chance*, 23(1):24–29, 2010.
- [20] B Everett. *An introduction to latent variable models*, pages 4–5. Springer Science & Business Media, 2013. ISBN 0412253100.

- [21] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.
- [22] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.
- [23] Rasmus E Madsen, David Kauchak, and Charles Elkan. Modeling word burstiness using the Dirichlet distribution. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 545–552. ACM, 2005.
- [24] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent Dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.
- [25] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
- [26] Trang Tran and Mari Ostendorf. Characterizing the language of online communities and its relation to community reception. *arXiv preprint arXiv:1609.04779*, 2016.
- [27] Tim Weninger, Xihao Avi Zhu, and Jiawei Han. An exploration of discussion threads in social news sites: A case study of the Reddit community. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 579–583. ACM, 2013.
- [28] Tae Yano, William W Cohen, and Noah A Smith. Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 477–485. Association for Computational Linguistics, 2009.

- [29] Aysu Ezen-Can, Kristy Elizabeth Boyer, Shaun Kellogg, and Sherry Booth. Unsupervised modeling for understanding MOOC discussion forums: a learning analytics approach. In *Proceedings of the fifth international conference on learning analytics and knowledge*, pages 146–150. ACM, 2015.
- [30] Steven Bird, Edward Loper, and Ewan Klein. *Natural language processing with Python*, pages ix, 60, 107, 203, 245–250. O’Reilly Media, Inc., 2009.
- [31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [32] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, pages 17–18, 47, 88. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 2009. ISBN 0131873210.
- [33] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, pages 80, 243, 275–276. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [35] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL <http://arxiv.org/abs/1605.02688>.
- [36] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [37] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for Youtube rec-

- ommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.
- [38] Christina Christakou, Spyros Vrettos, and Andreas Stafylopatis. A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools*, 16(05):771–792, 2007.
- [39] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.
- [40] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.
- [41] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.
- [42] TC Wong, Hing Kai Chan, and Ewelina Lacka. An ANN-based approach of interpreting user-generated comments from social media. *Applied Soft Computing*, 52:1169–1180, 2017.
- [43] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
- [44] Martin Porter. Snowball. <http://snowballstem.org/>. Accessed: 2018-06-07.
- [45] Andreas Mueller. Wordcloud. https://github.com/amueller/word_cloud, 2016. Accessed: 2018-06-07.

- [46] Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015. ISBN 151730007X, 9781517300074.
- [47] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.
- [48] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, pages 142–145. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.
- [49] Eric W. Weisstein. Harmonic mean. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/HarmonicMean.html>. Accessed: 2018-06-03.
- [50] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015. Accessed: 2018-06-07.
- [51] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [52] Jeff Heaton. *Introduction to Neural Networks for Java*, pages 158–159. Heaton Research, Inc., 2nd edition, 2008. ISBN 1604390085, 9781604390087.
- [53] Nicholas Buhagiar, Bahram Zahir, and Abdolreza Abhari. Using deep learning to recommend discussion threads to users in an online forum. In *Proceedings of the 2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018. In press.
- [54] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.
- [55] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

- [56] Atefeh Farzindar and Diana Inkpen. Natural language processing for social media. *Synthesis Lectures on Human Language Technologies*, 8(2):1–166, 2015.