CONTENT RECOMMENDATION ON SMALL SCALE SPARSE SOCIAL NETWORK

by

Bilal Khan

BSc. Ryerson University (2011)

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

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 othermeans, 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

CONTENT RECOMMENDATION ON SMALL SCALE SPARSE SOCIAL NETWORK

Bilal Khan

2019

Master of Applied Science
Electrical and Computer Engineering

Abstract

Small scale social network platforms suffer from increased sparsity, both in the size of text posted by users and the number of posts, as well as the specific nature of kurtosis that affects all platforms, yet more so on an emerging platform. In this work we examine a dataset from such a platform, where the majority of the activity is in the form of user generated text; both posted content and comments left on those posts. We inquire into what techniques would present suitable recommendations for a platform with a similar characteristic dataset. We evaluate leading textual analysis techniques and show how topic-model based techniques present a viable means for recommendation on such a platform as compared to other simpler, or more advanced techniques.

Contents

Abstract **iii**

List of Figures **vii**

List of Tables **x**

Acronyms **xi**

1 Introduction **1**

1.1 Motivation 3

1.2 Contribution 5

2 Literature Review **7**

2.1 Recommender Systems 7

2.1.1 Collaborative Filtering Systems 8

2.1.2 User Based Similarity 10

2.1.3 Item Based Similarity 13

2.1.4 Matrix Factorization 14

2.1.5 Content Based Recommender Systems 18

2.2 Application in Social Networks 20

2.3 Data Preprocessing 29

2.4 Evaluating Recommender Systems 34

3 Methodology **38**

3.1 Introduction 38

3.2 Approach Overview 39

3.2.1 Preprocessing 41

3.2.2 Comments 42

3.3 Term Frequency Based Model 42

3.3.1 Matrix Factorization 45

3.4 Topic Model 48

3.5 Neural Network Word Embeddings 51

3.6 Summary 55

4 Evaluation **57**

4.1 Dataset 57

4.2 Evaluation Methodology 65

4.3 Platform 67

4.4 Experiment Results 68

4.4.1 Term Frequency Model Results 68

4.4.2 Topic Model Results 72

4.4.3 Word Embedding Model Results 79

4.5 Discussion 81

5 Conclusion **84**

Bibliography **86**

List of Figures

2-1 Matrixes (mxf) and (fxn) in latent space representing users and items' latent feature relations. 15

2-2 The general components of a Content-Based Recommender System [27]. . 18

2-3 Plate notation for LDA [9]. 24

2-4 Two word2vec training models; CBOW and Skip-gram [59]. 27

2-5 Data preprocessing pipeline. 29

2-6 Data Reduction Strategies. 30

2-7 Two types of objective functions for FS evaluation; Filter and Wrapper. Credit: Ricardo Gutierrez-Osuna, Texas A&M. 31

3-1 Our evaluation pipeline gives us the flexibility to configure it at any stage. 39

3-2 Cascading Hybrid Recommender System Topology. 41

3-3 Terms are mapped to a corpus-level vocabulary index. 44

3-4 CBOW algorithm [75]. 52

3-5 Distributed Memory Paragraph Vector Extension to CBOW as described in [51]. 55

4-1 Data Table Layout. 58

4-2 Distribution of the number of posts of discretized lengths. 59

4-3 Sample post content from dataset: This post embeds an image the user shared. 60

4-4 Sample post content from dataset: This post shares a scenic route and warns of perils on the road. 60

4-5 Distribution of number of posts by the number of users. 60

4-6 Distribution of number of posts by the number of users, zoomed to show post bins between 0 and 200. 61

4-7 Frequency showing concentration of posts in top 56 users with post count of >= 50. 62

4-8 Top users post behaviour bisected between comments and non-comments. 63

4-9 Distribution of the number of comments of discretized lengths. 64

4-10 Comment exchange behaviour between sample users; showing frequent exchanges between top active users of the platform and very little external (other) participation, suggesting a small, tight knit community of top active users. 65

4-11 2D projection of the TF-IDF vectors. 70

4-12 Topic spread as shown after a 2D projection. 74

4-13 Topics as observed from topic modeling of wordy sources such as newspaper [24] - Example of a corpus with topic-richness. 75

4-14 Graphs showing RMSE and MAP for various sized Topic Models. 77

4-15 T-SNE of Post vectors of different sizes for Word Embedding models. . . 79

4-16 Graphs showing RMSE and MAP for various sized vector sizes for Word Embedding model. 80

4-17 Graphs showing RMSE and MAP for the leading variants from all evaluated models. 81

List of Tables

4.1 Random sample of comments from the dataset. 63

4.2 Sample from the ground-truth rating file accompanying the dataset. . . . 66

4.3 TF-IDF Prediction Results: Column MF is the Matrix Factorization results, MF-Comment is the technique where we substitute ratings if user left comment on a post, CB/COS is the cosine similarity model and CB/COS -Comment is, like the MF-Comment, where we substitute the similarity with a high similarity value 71

4.4 Table of topics for the MOTOR corpus at 10-topic. 73

4.5 Topic Model Based Recommendation Errors. 76

4.6 Word Embedding Based Prediction Results for Vector Size of 300. 80

Acronyms

A-UUTM At User-User Topic Translation Model. 26

ALS Alternating Least Squares. 17, 40, 45–47, 78

ARPANET Advanced Research Projects Agency Network. 1, 2

BBS Bulletin Board Service. 2

CB Content Based. x, 7, 8, 18–20, 40, 47, 69, 71

CBOW Continuous Bag of Words. vii, viii, 27, 51, 52, 54–56

CF Collaborative Filtering. 7–9, 11, 13, 20, 21, 26, 40, 47, 69, 78

CTR Collaborative Topic Regression. 25

DCG Discounted Cumulative Gain. 36, 37

FS Feature Selection. vii, 31–33

IR Information Retrieval. 8

IRC Internet Relay Chats. 2

kNN K-Nearest Neighbour. 12, 28

LDA Latent Dirichlet Algorithm. vii, 24–26, 49, 74, 82

MAE Mean Absolute Error. 34, 35

MAP Mean Average Precision. ix, 36, 67, 70, 71, 75–78, 80–82

MCC Mean-Centered Cosine. 10

MF Matrix Factorization. x, 14, 15, 17, 28, 43, 48, 51, 69, 71, 72, 75, 76, 78, 80, 81

PCC Pearson Correlation Coefficient. 10

POI Points of Interest. 26

RMSE Root Mean Squared Error. ix, 34, 35, 66, 67, 69, 71, 75–78, 80–82

RS Recommender Systems. 7–9, 12, 14, 18–21, 29, 30, 34, 35, 40, 47

SBS Sequential Backward Search. 32, 33

SFS Sequential Forward Search. 32, 33

SVD Singular Value Decomposition. 16–18, 28, 45

TF-IDF Term Frequency Inverse Document Frequency. ix, x, 22–24, 43, 48, 69–72, 81

UGC User Generated Content. 3, 5

Chapter 1

Introduction

The depth and prevalence of social network integration in human lives means it is how people share everything from breaking news to precious personal moments of their lives and because of this it can present opportunity to understand human behaviour and from there present opportunities for personal and commercial gains. Indeed social network can be used to sway voters [33], change perceptions [86], engage customers to increase satisfaction [72], as well as be used for passive analysis for things such as surveillance [85] or commercial customer churn [63].

Social media is an integral part of internet usage in today's world, yet it's not a new phenomenon. As far back as 1973, inventions to open the channels for digital communications for academics and personal applications were being carried out in research labs at universities around the world. Parallel experiments to the Advanced

Research Projects Agency Network (ARPANET) (the basis for today's internet) in communication protocols saw the creation of the PLATO system on which the concept of the 'internet chatroom' took shape in the form of the *Talkomatic* program [41]. The idea was powerful and there was no going back, soon after, Usenet came onto the scene in 1979 facilitating sharing of news, stories and discussion in a distributed means. Around the same time, Bulletin Board Service (BBS) began being run by hobbyists that users could connect to to share software, pictures, news and other content. As the pace of technology increased and personal computing started becoming accessible to more people, Internet Relay Chats (IRC) in 1988 and direct communication programs such as ICQ in 1996 [76] were launched and adopted to continue the personal communication revolution. In 1997 *SixDegrees.com*, being touted as the first social network network on the internet [29], came onto the scene, but due to the timing and people's distrust of the internet (users were not willing to divulge information about themselves and not willing to meet strangers online), it was shuttered in 2000. However, after just three years, the social network boom occurred, with MySpace, YouTube, Facebook and Twitter all coming onto the scene shortly after one another.

In today's world, social networks are ubiquitous. At the time of writing of this work, according to Alexa rankings[1], out of the 20 most visited websites daily by internet users, 10 of them are social networks. These networks include Youtube, Facebook, Twitter, Reddit, Instagram, and Weibo Sina. One thing common to all of these platforms is

[1]Alexa Rankings (https://www.alexa.com/topsites), made available by Amazon's Alexa division, tracks and monitors daily website visitor trends to millions of websites. Our observations of these rankings are from April 2019

that bulk of their operations are driven by User Generated content [82]. User Generated Content (UGC) on social network sites has experienced explosive growth [60], affecting and disrupting modes of communication, brand perceptions, journalism, social causes among other landscape shifts [20]. Due to the concept known as *Social Contagion*, as in the analog life, so can clusters of users on social networks influence decisions of their peers; decisions such as those affecting fashion trends, travel destinations, purchasing advice, and even political leanings to name a few. Due to this level of sway, social networks are prime hubs of influence and information exchanges and as such the central tenet of modern marketing; the work in [2] describes the various roles of social work during all stages of product development life cycle that can drive sales for a company.

1.1 Motivation

We have made the case so far for the ubiquity of social network in the lives of connected people, as well as how such networks are mined to drive sales and sway influences. This level of interaction and transaction we know works for large scale networks such as Facebook and Twitter where billions of users are engaged in active transactions. However, what can an emerging, mini-network, that is focused on a specific niche do to become a viable hub of activity? As *sharing* is one of the core principles of social network [42], more specifically, sharing of user generated content, a new novel network must then posses the ability to allow its users to disseminate information through its system to

other users. It is essential for social network platforms to foster growth by compelling user participation and driving user-generated content. Launching and incubating a new social network platform is remarkably difficult as the platform needs to be novel, attract and retain users. To retain users or drive growth, networks like Facebook, Twitter and LinkedIn recommend other users, news articles, jobs, advertisements, and products the target user might be interested in. This in effect keeps users engaged and staying on the platform, possibly even recommending their friends and social circles to participate in the social network site, as well. While numerous literature exists on consuming, mining and analysing established major social networks such as Weibo, Facebook, Twitter and LinkedIn, there are very few studies [60, 42] and literature on niche start-up social networks. Small social networks need to establish themselves, and recommendation of content to their user base is crucial to breeding user engagement on a user-generated social network site. Such networks face the same concerns as larger, more established networks (such as Facebook) of long-tail participation characteristics (kurtosis), sparsity of ratings and feedback, and cold start problems. And, due to their considerably smaller size, those effects are heightened. Models representing those datasets also face problems of over-fitting during training due to the lack of data.

1.2 Contribution

By building on the *sharing* tenets of social network requisite, an emerging network can retain and engage users. In this work we explore the techniques which can provide performant UGC recommendations to the users of the network; we describe a dataset (herein referred interchangeably as the MOTOR dataset) obtained from an emerging social network website that is geared towards a niche of motoring enthusiast. The site, and therefore the data, is relatively sparse, and contains mainly text-only user-generated features in the form of 'mini-blogs' and comments. We describe the technical characteristics in details in Chapter 4. This work also presents our findings on which recommender strategies work well for a dataset of such characteristics. As such, our contribution in this work is the determination of strategies that produce performant recommendations given the constraints of nature of the dataset. We evaluate the techniques for prediction of similarity using absolute error metrics, as well as a Mean Average Precision for ranked recommendations.

The rest of this thesis is organized as follows:

- In Chapter 2, we review the types of recommender systems, the modelling strategies used, especially is micro-blogging environment, recommender evaluations and wrapping up with data preprocessing techniques.

- In Chapter 3, we outline our methodologies for the experiments; our repeatable evaluation pipeline and the three models that yielded better results.

- In Chapter 4, we explore the technical characteristics of the dataset, analyse and discuss the results of the experiments.

- Finally, in Chapter 5, we draw to close our work in a report conclusion.

Chapter 2

Literature Review

This chapter explores some of the existing literature and techniques related to the task of recommendation.

2.1 Recommender Systems

The popularity of social media in recent years has given rise to numerous Recommender Systems (RS). Most of these systems, however, can be classified into a few 'classes' [90]. The two predominantly large classes of recommender systems are Collaborative Filtering (CF), and Content Based (CB), another class of recommender systems, referred to as Hybrid Recommender Systems combines them together into a system that attempts to resolve the weakness of one with the strength of the other.

2.1.1 Collaborative Filtering Systems

Collaborative filtering is driven by the idea that effective recommendation stems from people who share similar interests [77], and aims to predict a rating for an item based on what other users with similar interests have rated on this item [78]. In CF RS, ratings are captured from the users in generally three ways, a rating range where a user may rate an item on a continuous scale between a low value and a high value such as a 5-stars rating scale. A boolean discrete rating system, known as a binary rating scale, captures a like-dislike style rating. And lastly, a unary rating is one that captures interaction with an item such as opening and reading a news article on a news website would indicate a user's interest in that item. All three types of ratings are captured as a direct response from the user or the user's interaction, in this respect, ratings depict clean, non-inferred data about users' interests (the two former, scalar and binary are known as explicit feedback, where as unary interaction is considered implicit feedback). In contrast, the Content-Based filtering RS, as the named implies, harnesses a user profile and item features to make recommendations [74] based on items the user may have liked previously. This approach derives from the field of Information Retrieval (IR) [4] and as such strives to model the document into appropriate representations. Through IR influence, the items can be represented by structured data and the item features mentioned earlier become part of this structured data as properties or attributes. An IR query is then executed and the results matched and ranked. Generally, CB RS rely on the user's own provided implicit (e.g. ontological semantics from comments left by the user on an item such as

news, can indicate positive or negative impressions) or explicit ratings (similar to CF's scalar and binary ratings) to build a user profile [1].

Collaborative Filtering utilizes in-memory methods, such as neighbourhood computations, as well as model-based methods which are based on factorization or require training to generate a representative model [44]. One of dominant methods for CF RS is the neighbourhood-based approach which works under the assumptions that similar users have similar interests, and similar items are likely to be interesting to similar users. The concept of similarity is based on the idea that *closeness* of two points in a domain can be expressed by a distance metric [17]. One of the predominantly used methods to determine similarity is the Cosine similarity, which gives similarity value between two non-zero vectors. In the case of determining similarity between two users, the two vectors being compared represent individual user's rating on every item being considered. More formally, let r_{ui} be a the rating given to item i by user u in the vector r_u. Then cosine similarity of two users, u and v, is calculated between vectors r_u and r_v as shown in equation 2.1.

$$similarity = cos(\mathbf{r}_u, \mathbf{r}_v) = \frac{\sum_i r_{ui} r_{vi}}{\sqrt{\sum_i {r_{ui}}^2 \sum_i {r_{vi}}^2}} \tag{2.1}$$

2.1.2 User Based Similarity

When users assign a rating to an item, they do so according to an internal scale, regardless of whether an existing scalar constraint exists (e.g. user asked to rate item between 1 and 5) [64]. For an item that they absolutely like, they may give it a rating of just 4 instead of 5. Conversely, a user may give a rating of 3 despite finding the item uninteresting. For this reason, internal biases (often referred to as *taste*) in rating either too low or too high can be softened by the use of a normalization technique, one of the widely used method of which is the Mean-Centered Cosine (MCC) [81]. MCC finds the rating-mean of a user and classifies a rating as either positive or negative to that mean; this is done by subtracting a rating r_{ui} from the mean $\overline{r}_u$ to get the mean-centered rating $h(r_{ui}) = r_{ui} - \overline{r}_u$. A popular similarity measure used that incorporates this mean-centering is known as Pearson Correlation Coefficient (PCC). In PCC, as the coefficient approaches a value of 1, the correlation between the two vectors is considered to be strong. A value of 0 signifies vector independence and no correlation. PCC, shown in equation 2.2, also can approach a negative value of -1, meaning a strong negative correlation [7]. MCC is one of the methods of normalization, but as [26] shows, it is a robust and competitive method, outperforming others such as the Spearman method.

$$\text{userSim}(r_u, r_v) = \frac{\sum_i h(r_{ui}) h(r_{vi})}{\sqrt{\sum_i (h(r_{ui}))^2 \sum_i (h(r_{vi}))^2}} \tag{2.2}$$

While user-user predictions can lead to good recommendations for users, there

are some concerns. Sparsity of rating and size of the dataset have serious consequences for this particular technique. In the case of sparse ratings, finding good quality, unbiased neighbours become challenging. Pearson similarity measure suffers from a bias where if two people agree on a universally acclaimed movie, that should be of little importance as opposed to two people agreeing on a non-acclaimed movie. Yet the Peason correlation accounts such coratings as high weighted similarities. Scalability is another problem with user-user CF; as the number of users and items grows in a dataset, the nearest-neighbour calculations become increasingly exhaustive. Running such calculations take considerable amount of time, rendering near-real-time recommendation unachievable unless optimizations are employed [78].

Equation 2.2 shows the similarity formula for establishing user-similarities, a similar Equation (eq. 2.4) can be used to establish relevance between the item vectors, as well. First described in [54], the Item-Item similarity method discovers items that are similar to one-another based on their feature descriptors. Item based recommendation attempts to solve the concerns of user-based similarity, specifically, an increase in performance and handling of larger datasets (scalability). In item-based similarity computation (explained below), higher accuracy is also obtained by having a more robust neighbourhood. In terms of performance and scalability, item-item similarity is also more efficient in the use of memory and can perform neighbourhood discovery computations offline.

Once the similarities are determined, classification is the next step to determining a recommendation. Classifiers are either supervised or unsupervised, where the

former requires a labelled training set to generate a model. One of the prevalent supervised method is the K-Nearest Neighbour (kNN) classifier [17] which is trained on a set of n, (p, l) tuple where p is a point and l is the associated label into k clusters, and upon query of a new point attempts to find the label of the closet cluster. Other supervised examples include Neural Nets, Decision trees and Rule-based, Support Vector Machines and Probabilistic Bayesian based classifiers [27]. Among the unsupervised classifier methods is the k-means partitioning which clusters the points into k partitions [3]. The k-means algorithm is trivial to implement and indeed one that is efficient as well. Algorithm 1 shows the simplicity of the algorithm. Yet it has certain limitations [3]. As with kNN method, k-means also suffers from the need to know the number of clusters k prior to running the algorithm. The selection of a proper k requires knowledge of the data being processed. Furthermore, initial selection of centroids has significant bearings on the outcome of the final clusters, and Mehdi et al. [27] suggest that k and neighbourhood selections are a delicate task that requires fine tuning on the data itself.

Algorithm 1 k-means algorithm.

procedure K-MEANS(k, D) ▷ k: # of partitions, D: dataset
  randomly assign k points from D as initial centers
  repeat
    1. for all points in D, reassign to closest centroid
    2. update cluster centroid to the new mean
  until convergence; no change in memberships
  return $\overline{D}$ ▷ k clusters and memberships
end procedure

For an RS the end-goal is to predict the item worthiness for a user and then rank unseen items for a user based on that worthiness score. In neighbourhood-based

prediction models, once neighbourhoods are determined, the task of prediction can start. Equation 2.3 [78, 37] shows the user-based prediction function that accepts a user, u (target user to whom a prediction is being made), and the item being evaluated i and determines, based on neighbour's preferences, whether that item will be found useful to user u. The neighbour's ratings for items are averaged and scaled by similarity weight (calculated in 2.2), and the highest rated predictions are recommended. The $userSim$ weights may need to be scaled to ensure normalization, so it is divided by the sum of all similarities. The prediction is calculated as deviation from the user's mean, $\bar{r}_u$, then added back to the mean to produce the final predicted rating, a process known as *average adjustment*.

$$prediction(u,i) = \bar{r}_u + \frac{\sum_{n \subset neighbours(u)} userSim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \subset neighbours(u)} userSim(u,n)} \tag{2.3}$$

2.1.3 Item Based Similarity

In item-based methods, *similarity* is computed for co-ranked items; items rated by the target user, u, and other users v, but where the other users have also rated the target item i. The similarity algorithm used in item-item CF, eq. 2.4, is almost the same as Equation 2.2, with the difference being that the similarity is computed along the item-axis in the User-Item matrix and the mean centering occurs for the item and not the user. The set of users being iterated in the sum is also different, where $RB_{i,j}$ being the

set of users who have rated both items i and j (known as being co-rated).

$$itemSim(i,j) = \frac{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{u \subset RB_{i,j}} (r_{uj} - \bar{r}_u)^2}} \tag{2.4}$$

For item-based *prediction*, the equation is similar to 2.3, except average adjustment is no longer required in Equation 2.5 because all item ratings are from the same user u. So the prediction equation can be rewritten as such

$$prediction(u,i) = \frac{\sum_{j \subset ratedItems(u)} itemSim(i,j) \cdot r_{ui}}{\sum_{j \subset ratedItems(u)} itemSim(i,j)} \tag{2.5}$$

2.1.4 Matrix Factorization

Neighbourhood based predictions are one of several methods RS use to recommend content. Here we describe techniques that exploit hidden relations within the data, known as latent factors. One of the most commonly used methods of latent factoring is the Matrix Factorization (MF) method [48]. MF techniques assumes data, in the case of RS, usually a set of user-item ratings, can be mapped to in a latent factor space. This means that the ratings in the user-item matrix are assumed to be the dot-product of the vectors representing user interest for latent factors (see Equation 2.6), and the item vectors representing the affinity towards those latent factors. Figure 2-1 shows an $M \times N$ user-item matrix whose rating value, r_{ui}, is determined by the Equation 2.6, where $\hat{r}_{ui}$

is the rating, q_i is the latent feature descriptor vector for the item and p_u is the latent feature descriptor for the user. The concept behind the latent factor model lends to a further assumption that the ratings can be decomposed into their factored matrixes in the latent space, then those matrixes can act as user profiles and item profiles. A product of those matrices will result in the user-item matrix being recreated and the unknown ratings being generated due to those latent factors.

$$\hat{r}_{ui} = q_i^T p_u \tag{2.6}$$

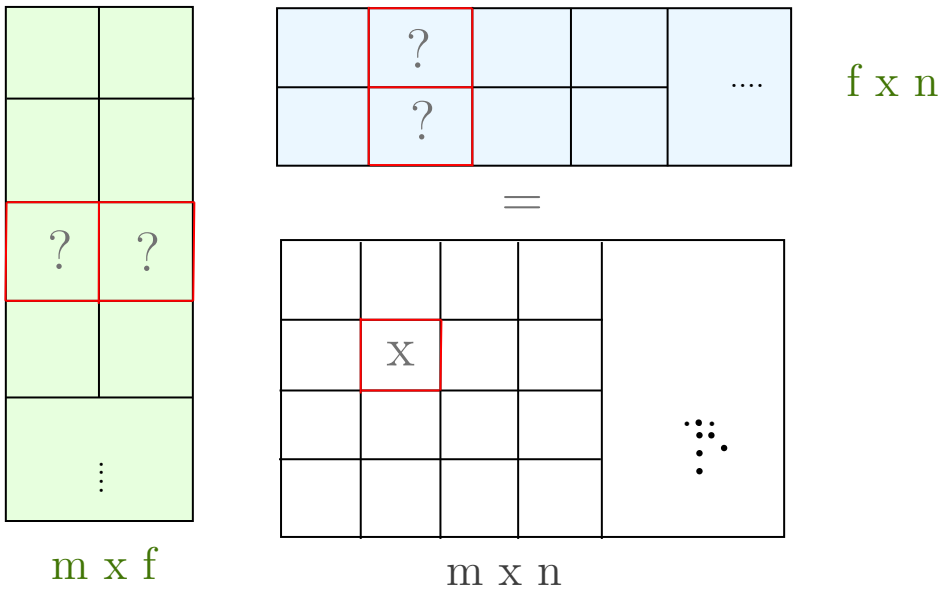


Figure 2-1: Matrixes (mxf) and (fxn) in latent space representing users and items' latent feature relations.

As with nearest neighbour algorithms using mean-centering to reduce the effects of user's rating biases, MF attempts to capture rating essence, without including distortions brought in by individual users rating biases. This method, known as *baseline predictors*,

is used to make the baseline prediction for an unknown rating and is shown in the Equation 2.7. In the Equation, b_{ui} is the intended prediction baseline, μ is the mean rating of all the items in the dataset, b_u is the deviation of the user's rating from the average, while b_i is the deviation of the overall rating of item i from μ.

$$b_{ui} = \mu + b_u + b_i \tag{2.7}$$

A well known algorithm used for decomposing the user-item matrix into their latent space models is the Singular Value Decomposition (SVD) method [43]. SVD factorization, a form of polar decomposition (where the resulting decomposed components stretch along orthogonal axes), of a matrix M amounts to $M = \mathbf{U}\Sigma\mathbf{V}^T$, where M is the $m \times n$ matrix being factorized, U is a $m \times m$ matrix, σ is a diagonal $m \times n$ matrix and V is a $n \times n$ matrix. SVD proves that, using the property of symmetric matrix (in this case U and V) and their orthonormal basis of their eigenvectors, it is possible to decompose any matrix M into its components. In order to generalize a model and make a prediction, we could minimize the squared difference between the rating r_{ui} and the prediction $\hat{r}_{ui}$ from Equation 2.6. However, that equation does not take into account the baseline predictors we specified in Equation 2.7. So a more refined approach, is to redefine $\hat{r}_{ui}$ as shown in eq. 2.8.

$$\hat{r}_{ui} = b_{ui} + q_i^T p_u = \mu + b_u + b_i + q_i^T p_u \tag{2.8}$$

At this point, minimization of the squared difference between $\hat{r}_{ui}$ and r_{ui} can take place, however, due to over-fitting concerns, regularization is used such that the learning is performed on Equation 2.9. Where λ_4 is the regularization damper weight, determined via cross validation [48].

$$\min_{b_*,q_*,p_*} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_i - b_u - q_i^T p_u\right)^2 + \lambda_4 \left(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2\right) \tag{2.9}$$

Two popular algorithms for minimization are the Alternating Least Squares (ALS) and the stochastic gradient descent by Simon Funk [8]. In ALS, the goal is to discover the unknown q and p variates by using least squares algorithms and solving two different cost functions; switching between fixing q_i and solving for p_u and vice versa. Limiting derivation to one variable at a time makes the problem convex, and thus ensure a minimal solution will be found. This technique is slower than gradient descent, however can be effective under massively parallel conditions [48] or when facing dense matrices. Funk SVD uses stochastic gradient descent, $\min_{p_u \perp p_v} \sum_{r_{ui}\in R} \left(r_{ui} - p_u \cdot q_i\right)^2$, by iterating through all known ratings and partially derivate the squared of difference equation with respect to both q_i and p_u, updating both at the end of every iteration. The entire process is iterated several times, as is the nature of stochastic gradient descent algorithms. After the latent components q_i and p_u are generated, Equation 2.7 can be used to make prediction for queries. Not all user-item matrices can be used for decomposition through SVD, however, as MF methods treat missing ratings as undefined behaviour. This is particularly a problem in very large datasets, so one solution is to assign a default value, such as 0, in place of missing ratings. This practice obviously then turns sparse matrices

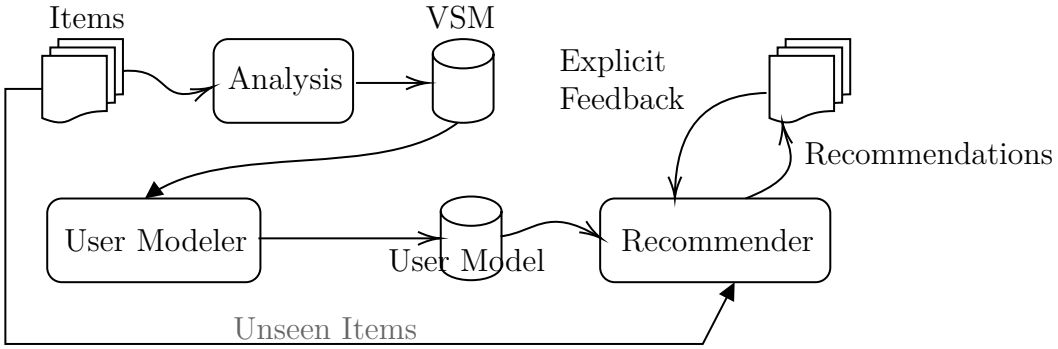


Figure 2-2: The general components of a Content-Based Recommender System [27].

into dense ones, leading to severe performance issues, as well introducing large biases as everything is assumed to be explicitly rated by the users as the default value. [84] suggests that users only rate a small subset of items, demonstrating that majority of the user-item rating is sparse and undefined. They test and see positive correlation between using a denser subset (removing and pruning inactive users and users who have not rated many items) of the user-rating matrix for SVD purposes. Furthermore, algorithms such as SVD++ attempt to improve accuracy by reducing undefined ratings through mixing explicit and implicit ratings.

2.1.5 Content Based Recommender Systems

As briefly described above, another popular RS is the Content-Based Filtering RS which utilizes structured representation, usually feature vectors, to describe the items in the system. Figure 2-2 shows a typical CB RS and its various stages. Usually, items are analysed from which an item representation structure emerges, typically a multidimensional

vector. A user preference model is also a key concept of a CB RS, which is generated from items the user has liked in the past. In addition to the explicit ratings, a CB RS may also depend on implicit rating data, usually derived from user's interaction with the site and items. Additionally, attributes of items play a substantial role in creating the feature descriptor of the item. As previously mentioned, CB RS process unstructured information and produce structured representation. For this task both supervised and unsupervised machine learning modellers can be used [52]. More specifically the problem of learning from a user's historical behaviour for the purposes of CB RS is generally handled by a classification learner [66]. Using a training set, items are labelled as either interesting or not-interesting based on user's explicit feedback. This data is the basis of the user model. Alternatively, rule inference and the similar Decision Tree algorithm are examples of supervised learning that can be used to model the user's preferences as well. As the name suggest a decision tree is a graph whose nodes perform tests on the attributes of a query data and the edges lead the resolver to possible states based on the outcome of those tests [5]. In a decision classifier, the leaf nodes are the classes, so that when a test on data's attributes leads to a leaf node, that class is then considered as the label for that data. Aside from rule induction and decision tree induction, other methods of user profile generation include the Nearest Neighbour algorithm, explained earlier, considered popular in the field of RS classification due to their simplicity. Still other methods include probabilistic approaches such as naive Bayes classifiers which are known to have effective classification for text documents [23], Neural Networks and Support Vector Machines.

CB RS face limitations in the form of requiring larger amounts of rating data on individual users in order to generate their accurate user profile. This creates a problem for recommending items to new users who are likely to not have many interactions with the systems, if any at all. The other limitation of CB RS is known as the *serendipity* problem; an issue where recommendations only span the breadth of the user's own interests as they have specified it to the RS (in the form of ratings). In CF RS, this behaviour is not as extreme due to users-user and item-item methods ability to discover related, but novel content. However, despite these limitations, a CB RF offers tangible solution to the problem facing CF RS where, without availability of coratings from neighbouring users, an item cannot be predicted. As CB RS rely primarily on the user's own preferences, a lack of rating from other users is less of an issue.

2.2 Application in Social Networks

Ekstrand [26] describes MovieLens, a social network for recommending movies to its users, as it was in July 2014. As MovieLens the recommender (as opposed to the dataset) is an ever-evolving research effort, the complete state of the system has not since been published. MovieLens combines several CF methods such as User-User, and Item-Item recommendations. In User-User CF, similar users are discovered using k-nearest neighbours algorithm and the movies they liked are recommended to the active user. MovieLens also employs Item-Item collaborative filtering. Item-based predictions

use item similarities determined based on the ratings of items that users have provided [77]. Item-Item collaborative filtering has many computational benefits over user-user collaborative filtering [77] such as being more scalable without optimizations and handling sparsity more robustly the user-user CF.

Linden et al. [54] provide details for a recommender system used at Amazon.com to recommend books and other items for purchase at the online retailer. In a traditional collaborative filtering RS, Users and Items are arranged in an $M \times N$ matrix. For a retailer like Amazon, which boasts a user base in the tens of millions and an item catalogue in the millions the resulting data structures required to produce similarity computations such as nearest-neighbour become computationally taxing, not to mention with that level of sparsity, simply inaccurate. One of the ways Amazon reduces the data complexity is by using Instance Selection and Feature Selection techniques (described in more detail in Section 2.3); by randomly sampling users they reduce the size of the set but maintain relative accuracy [30]. Removing underrated products and inactive users from the matrix further increases computation efficiencies. Further still, dimensionality reduction is employed to reduce dimensions even more - employing Principal Component Analysis and unsupervised clustering to reduce the users down to those who may already be similar, resulting in a User-item matrix where users are all similar for a given similarity measure. This similarity class can be discovered by the clustering algorithm or maybe manually specified. The contribution from Amazon to RS field is the Item-Item similarity (as previously described in Equation 2.5), it processes larger datasets than the

User-User similarity measure can, as well as provides near-realtime recommendations. This is because the item-item method maintains an item-similarity table that doesn't change unless an item is added or removed or a customer either buys or rates a product [55]. The computation is done offline and so the online segment of recommendation is a simple look up of all similar items to the one the user has already bought or rated, then ranking the items. While the item-item computation is heavy, the online lookup is relatively quick.

Content Based Recommendation involves item analysis and user-profile generation, as shown in Figure 2-2. In social networks, in particular, micro-blogging systems, recommendations involve ingesting content posted by users or influential users in the network.

When it comes to text, vocabulary term frequency statistics can be an effective modelling technique. Ramos [73] proposed an algorithm, Term Frequency Inverse Document Frequency (TF-IDF), to identify terms in a document that may have high relevance. It does this by scaling term-frequency with how common or rare the term is in the corpus, usually scaled logarithmically. Formally defined in Equation 2.10, where $|D|$ is the document count in the corpus and $|\{d \in D : t \in d\}|$ is the number of documents where t is observed, combining the first two equations, we arrive at the final Equation

2.10.

$$\begin{aligned} \mathrm{tf(t,d)} &= \frac{f_{t,d}}{|\text{words in d}|} \\ \mathrm{idf}(t, D) &= \log \frac{|D|}{|\{d \in D : t \in d\}|} \\ \mathrm{tfidf}(t, d, D) &= \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D) \end{aligned} \tag{2.10}$$

TF-IDF penalizes words that appear more often and in more documents. Words such as 'is', 'the', 'and' appear frequently in documents and are of little significance. Yet if the word 'swimming' appears in few documents, then it will have the greatest weight in that document in which it appears with greater frequency. TF-IDF algorithm is successful and is widely used for text feature selection due to its simplicity, performance and robustness [93]. [57] use TF-IDF to retrieve 'tags' from the user's posted corpus in addition to any tags that user may have explicitly specified in the post. A vector is created for every user, containing n tag weights. Tags provided by the user and those retrieved from the user's post (highly weighted TD-IDF terms) have their weights set in the vector, other values default to 0. These vectors combine to form a user-tag matrix, which is then further processed for tag-tag similarity correlations. Resulting data is then used to build a user-user similarity matrix, from which user recommendation can be made. Hannon et al. [34] recommend users to follow on the Twitter social media platform. Using TF-IDF, the authors explore relations within the tweet's textual context. Using the open-source Lucene platform, authors index the tweet corpora of all the twitter users in their dataset. They then query the text-search using the terms of user corpora. What is yielded by the engine are *documents* with TF-IDF-based scores, where *Documents* represent individual twitter users from the dataset. Chen et al. [18] publish a hybrid

technique to recommend users to follow by combining the content of the tweet text and the social graph of the users. The user's interest are represented by a vector of tf-idf of terms from his/her tweets. From this a user-user matrix recommends which users are most similar. A further ranking stage ranks the recommendations based on the social structure features such as the ratio between the number of followers and followees, and the numbers of shared followers. Tajbakhsh and Bagherzadeh [83] propose a method to recommend tweets by using TF-IDF to weight terms, then using a semantic technique to create another vector for every term. A cosine similarity of these semantic weights is used to produce a Top N ranked recommendation of tweets.

One of the most deployed techniques are to determine unsupervised latent factors discovery methods. These methods use statical and probabilistic approaches to develop models to predict the hidden factors that have led to the production of a document. Chief among these techniques is the Latent Dirichlet Algorithm (LDA). Blei et al. [9] published the Latent Dirichlet Algorithm (LDA) and at its core, LDA is a generative probabilistic model which posits that the words in a documents are the result of a topic distribution which underpins every document, the latent distribution.

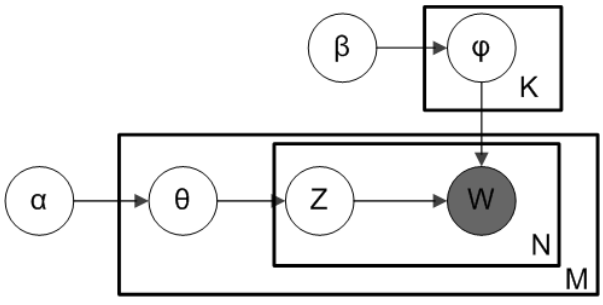


Figure 2-3: Plate notation for LDA [9].

Figure 2-3 shows the plate notation for LDA. A plate notation [14] is a common representation tool to explain the probability dependence in a Bayesian inference model. Each variable is represented as a circular 'plate'; showing the influence of two parameters α and β on topic-document distribution θ and word-topic distribution φ, respectively. In practical implementation, θ distribution is represented as M x K matrix, where M is the number of documents in the corpus and K is the number of topics defined by the user of the LDA algorithm. Similarly, the φ mixture is implemented as a K x Z matrix where Z is the number of words in the corpus vocabulary. The W plate, representing words, is greyed out indicating that words are the only observable element of this model. The generative process of LDA attempts to 'learn' the distributions θ and Z, given the observable w and parameters α and β through variational inference. The results of the training are distribution mixtures, the most useful of which is the document and topics, showing the strengths (probabilities) of each of the k-topics for each document in the corpus. Researchers from Yahoo, Pennacchiotti et al. [67] publish a comparative study for recommending tweets to users by grouping similar users based on topic distributions. Wang and Blei [87] authored literature on recommending scientific articles using topic modeling. They present a hybrid approach where collaborative filtering and probabilistic content analysis are combined. As research papers are wordy articles on which topic modeling algorithms like Latent Semantic Analysis (LSI) and Latent Dirichlet Allocation (LDA) work well [9], this research extends the LDA modeler to include the collaborative filtering algorithm. Authors introduce the Collaborative Topic Regression (CTR) concept in this research. In CTR, the main concept is to generate the rating of every user

and document pair, this is done by exploiting topic distribution similarity via a regression step. This work claims significant improvement over LDA-based recommendation and marginal improvement over CF similarity-based setup. Using LDA in a less wordy environment, Gong et al [31] extend LDA to represent topic, tweet and user. They designed and evaluated LDA-extensions that considered users as part of latent factors and separately, users being mentioned as part of the latent factor space, too. The LDA extension, At User-User Topic Translation Model (A-UUTM) which took user mentions into consideration performed the best out of all other LDA models. In the pursuit of recommending geographical Points of Interests as tags and images, Jiang et al. [40] have published their approach to a hybrid system; combining both collaborative filtering and author-topic model. They propose to alleviate the cold-start problem which inhibits the performance of CF approach by extracting information from the description of the image as provided by the users of a social media site. Texts from all the images, including descriptions and comments, are concatenated together and fed through an probabilistic topic model, where all texts from one user comprises of one document. This produces a topic distribution centred around travel destinations for each user. This topic distribution is used to produce a map of similar users via cosine similarity. This information is then further combined with the ratings and travel history of the user to produce a Points of Interest (POI) recommendation. Chen et al. [19] propose a technique that combines topic modelling and topic graphs to produce a research paper and researcher recommendation. Researcher's profiles are crawled on a social network, plus their contributions put through LDA to obtain topics. These topics then are treated as tags and are rep-

resented in a vector. From there tag-tag similarity is obtained through cosine similarity and an edge is created between the tags with the distance been akin to the similarity value. This creates a graph with neighbouring nodes, allowing for recommendation of research articles and researcher themselves.

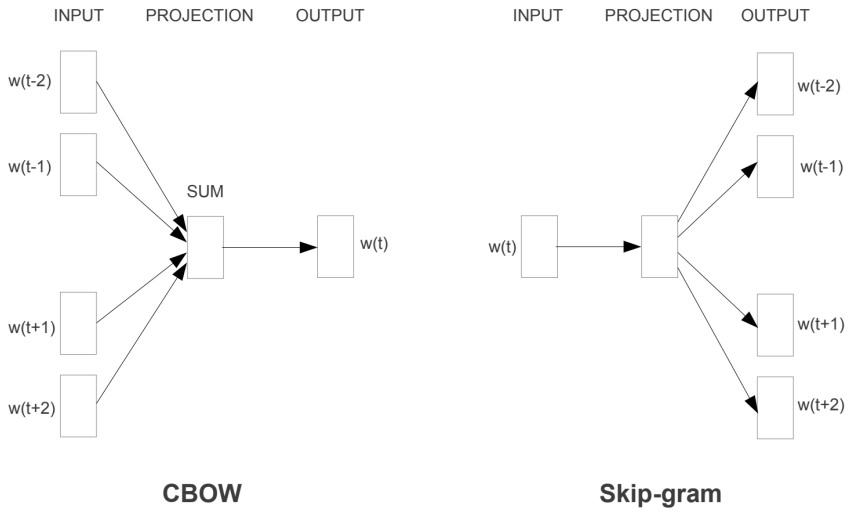


Figure 2-4: Two word2vec training models; CBOW and Skip-gram [59].

Elsafty et al. [28] propose a method for recommending job postings on a website. Using an unsupervised machine learning algorithm, Distributed representations of sentences and documents (Doc2Vec) [51] (which itself is derived from Word2Vec [59]), neural embeddings are used to build a model of the corpus into vectors. These vectors can then be used to perform recommendation based on cosine similarities. In the Continuous Bag of Words (CBOW) [59] model used in Word2Vec algorithm, the surrounding words generate the target word. A neural network (NN) (shown in Figure 2-4) is trained on a corpus to predict the word in the centre of a word-window; for example, in the window of *The Quick Fox Jump Over* the NN is trained to predict the vocabulary

vector for the term *Fox* based on the vectors of the words *The Quick Jump Over*. This allows clustering of words that have similar context, such as basketball players in the same team, since they're usually mentioned in the context of the team they belong to. In Doc2Vec algorithm, Word2Vec is extended to also include the document vector in addition to the neighbouring word vectors. The weight vectors of the neural network become the vectors for each word, as well as a document vector which identifies each document. Authors of [28] use the full-description of the job as well as the title of the posting to feed into Doc2vec as documents. In other literature, [61] compares several techniques, including Word2Vec for recommendation of movies using the MovieLens dataset. It evaluates Latent Semantic Index (LSI), which itself uses SVD to reduce the dimensionality of the Document-Word matrix into a matrix of word-concepts, and it also evaluates Random Indexing - an incremental method for learning a low rank representation of a document. Using wikipedia entries for each movie (boosting), the algorithms produced their respect vector state representations. Three recommendation algorithms, user-user kNN, item-item kNN and a Matrix Factorization (MF). The resulting recommendations' F1 generally favour the Word2Vec algorithm. Ozsoy [65] explores applying word embeddings to non-textual features from users in a recommender setting. Using data from FourSquare social network, it uses the check-in locations of a user as vocabulary, instead of text. Word2Vec produces a vector state representation of each user, using which a kNN algorithm is used to find the nearest neighbours for both items and users. The system can predict, given user's past checkin locations, where the user would like to checkin next.

2.3 Data Preprocessing

By their nature, RS deal with data that may not be in ideal form to produce effective pattern recognition or lend to statistical analysis. Cooley et al. [22] emphasise the need to preprocess the data prior to subjecting it to further analysis in order to increase statistical effectiveness.

Figure 2-5: Data preprocessing pipeline.

Figure 2-5 depicts the flow of the data preprocessing stage. Generally unstructured data, especially in the context of content-based filtering, is fed into the preprocessing system and the output of this stage is then mined by the algorithms. The term *data* can be a reference to various representation of information, though it's generally used to refer to a *record* or an *instance*. Still included under the same umbrella of *data* maybe attributes of that data; whether in the form of *field* in the event of a *record* or a *variable* in the event of an *instance*. Before algorithms in the *Data Mining* step can examine the information, the *preprocessing* stage ensures the data is clean from artifacts and influences that may affect the machine learning process. Data may be cleaned using numerous algorithms and rules and may involve filtering and transformation. Much of this cleansing step involves direct knowledge of not only how the data is stored and in what form is made available to the system, but also what algorithms are being employed

in the mining step as well as what pattern are to be recognized in the *interpretation step*. Cooley et al. [22] describe a data preparation technique for analysing web server access logs. The literature aims to track users as they browse a website from the website's access logs. They establish *User Sessions* from raw logs by cleaning out access logs for irrelevant content. Content relevancy is determined based on whether the url points to a page or a static asset (such as style sheets or images). A priori site topology is generated by crawling the site and this data is then used to facilitate the preprocessing stage. The reduction of non-relevant items from the log, as part of the data cleaning strategy, increased model effectiveness.

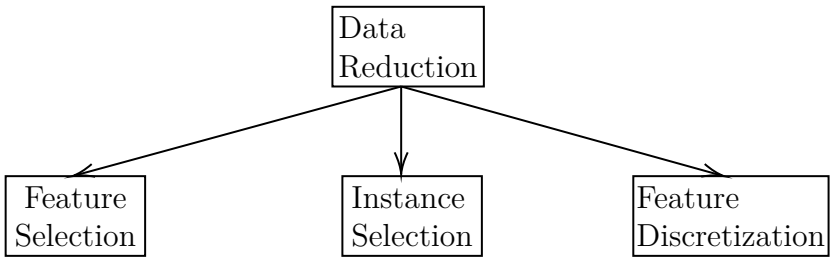


Figure 2-6: Data Reduction Strategies.

Kotsiantis et al. [49] published a study observing various strategies for pre-processing data for different applications. In the field of machine learning, of which RS is arguably a branch, data preprocessing aims to increase not only the effectiveness of the algorithm but also the performance as well. It does so by targeting redundancy, noise and incomplete data, as well as data reduction techniques to enable algorithms to ingest large datasets. Jankowski et al. [39] published a study to investigate the effects of *Instance Selection* algorithms that clean the dataset by means of noise filtration, data condensing and/or data representation (known as prototying). The general idea is to

reduce the dataset by smartly selecting instances of data to trim. Algorithms may be as straight forward as scanning variables for outliers; such as boundary and range of a field, to cardinality or even artifacts such as misspelt terms [49]. For multivariate/multidimensional datasets, distance based techniques may be used to discover outliers as is done in [45] to detect fraudulent patterns in spending activities. A similar application of e-commerce fraud prevention [13] uses a novel technique of *local outlier factor* that posits that for some applications cluster membership is better given as a degree, rather than a boolean inlier-outlier value. While Instance Selection can improve performance of machine learning algorithm, [30] concludes that random sampling and Stratified sampling are other ways to reduce very large datasets but still maintain accuracy. Figure 2-6 shows

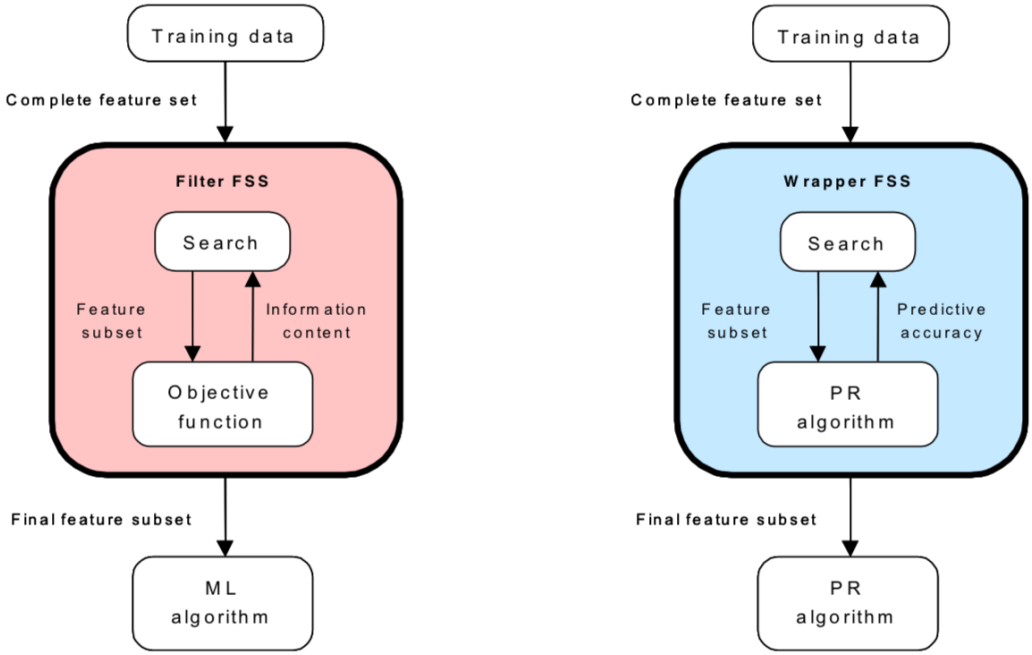


Figure 2-7: Two types of objective functions for FS evaluation; Filter and Wrapper. Credit: Ricardo Gutierrez-Osuna, Texas A&M.

the three subfields of data reduction. After Instance Selection we explore Feature Selec-

tion (FS), which is the act of purging as many features that may have little bearing on the end goal. Features are classified, in general, under three classes; *Relevant*, *Irrelevant* and *Redundant*. The goal of the feature selection process is to generate a reduced feature set from the original feature set (known as a search strategy) and evaluate whether the generated set is a better set (known as an objective function). Figure 2-7 highlights the two general types of objective functions, Filter and Wrapper. Filter objective functions depend on the data's metrics such as Euclidean distances to determine candidacy of a feature. Meanwhile the Wrapper objective function uses classifiers to run the feature set on a sample data to determine its efficiency. The act of discovering the perfect reduced feature set would ideally be exhaustive, but that would exhibit prohibitive computation costs [49]. Large amounts of literature exists on FS using non-exhaustive strategies. [21] and others suggest that an effective feature set is one that has a better overall mixture of relevant features rather than the best features. Using regression, Stepwise Multiple Regression (SMR) [92] incrementally (stepwise) adds a feature to the set and calculates the correlation coefficient (R^2) with its stopping strategy being when a newly added feature does not significantly contribute to the coefficient. Despite all the FS strategies, Piramuthu [68] have published a comparative study evaluating various FS algorithms and concluding that no one particular algorithm exhibits leading results, however the use of a FS strategy does entail generally favourable results.

Some of the most deployed FS techniques [68] in the field of machine learning are the greedy algorithms of Sequential Forward Search (SFS) and the Sequential

Backward Search (SBS) [70]. In SFS, regression is used to steer the feature selection by maximizing the cost function $J(Y_k + x^+)$, where Y is the generated feature set and x^+ is the candidate feature that maximizes the cost function. The sequence starts from an empty candidate set $Y_0 = \{\emptyset\}$. SBS is the opposite of SFS, starting from a complete feature set and removing a feature that has the least affect on the cost function.

Beyond FS, continuous numerical features can be 'discretizated' to improve computational performance, especially for inductive machine learning. The problem of discretization is one of discovering effective number of intervals and boundaries [47]. Naive routines of discretization of continuous values include uniform-sized binning, and uniform number of values in a bin, and they generally lend to poor result in ML algorithms. Treated as a possible knowledge discovery problem, most discretization algorithms are classified as either *supervised* or *unsupervised*. Additionally, they can further be classed as *global* or *local*, and *static* or *dynamic* [25]. Global and local algorithms create bins that spans across the entire $k-dimensions$ of the value or in specific regions of the value, respectively. Static and dynamic methods create partitions based on the feature solely or consult other features in the class to determine the partitions, respectively. In the case of supervised and unsupervised methods, supervised routines work by observing provided labels for the classes and partitioning based on further knowledge of those labels.

2.4 Evaluating Recommender Systems

We have reviewed literature related to the most common types of Recommender Systems, and the methods and algorithms they employ to make useful predictions of items to users. In this section we review offline methods for evaluating the worthiness of a recommendation. In our work, we cannot perform A/B tests with the users, nor generate implicit feedback by observing user's interaction with the recommendations, under such circumstance we will discuss here relevant offline processes. The end goal of an RS is to yield the best possible recommendation. One way to evaluate the recommendation outputted by system is to inspect the predictions made by the RS. Accuracy in prediction is a measure most sought for RS recommendations [79] as it can be performed offline without requiring user input. Mean Absolute Error (MAE) is a measure commonly observed in literature to compare the accuracy of various recommendations. MAE produces a single scalar value from the mean of the absolute difference between predicted value and the actual value and is expressed as MAE $= \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{N}$. The value is a simple metric to express the error rate of an RS prediction. Similarly, Root Mean Squared Error (RMSE) produces a non-negative value from the differences in actual and predicted figures. A value between 0 and $+\infty$, with 0 being a sample with no error, and hence a perfect recommendation. Described by the equation RMSE $= \sqrt{\frac{\sum_{k=1}^{N} (\hat{y}_k - y_k)^2}{N}}$ (where N is the number of samples compared, $\hat{y}$ being the predicted rating and y the actual), RMSE of a single recommendation on its own is of little use, but can provide a robust measure to compare various recommendations. Of the two, RMSE is particular more sensitive to

higher magnitude of errors and thus is useful when larger error magnitudes are particularly undesirable. Both MAE and RMSE can be calculated per user to determine the error on a per-user basis instead of the entire sample.

Prediction accuracy is useful when evaluating the correctness of a prediction. For Recommender Systems, the tasks go beyond assigning a rating to an item, a common task for RS is to produce a list of recommendations that the user may find interesting. Precision and Recall measures provide an insight into recommended list, especially with respect to answering questions of *How many instance are relevant out of the total that were recommended?* (precision) and *How many instances are recommended out of all those that should be considered relevant?* (recall), respectively. More formerly, $precision = \frac{tp}{tp+fp}$, where tp is true positive, the instances correctly recommended, and fp are the false positive, the instances wrongly recommended as interesting. Recall is defined as $recall = \frac{tp}{tp+fn}$, where fn is false negative, the instances that should have been recommended, but were not. RS strive for results where ideally both their recall and precision ratios are high, as it suggests that the list was as relevant as possible and that as many relevant results as possible were recommended, however it is observed that precision and recall are usually inversely related [79], so an effective RS would attempt to balance the two. A common trait of RS is to return a ranked list of recommendations, typically known as *Top N*, where N is the length of the list. A precision and recall of this list is generally a more meaningful metric to evaluate the recommendations [79]. Known as Precision at N, the measure evaluates a ranked list of length N and

usually includes calculating the recall ratio in addition to the precision. Additionally a graphical curve, known as the precision-recall curve depicts the relationship between recall and precision. The curve plots are the result of a binary classification probability distribution extracted from the precision/recall ratios. The threshold for the probability is changed and more curve points are generated until enough points are accumulated to connect them together into a graph. The role of this curve is to visually locate a balance point between precision and recall for the evaluated model. Multiple models' curves can be overlaid onto a graph from which the performant model would emerge. The area under this curve acts as another measure to compare the models with [12], essentially encoding the precision-recall ratio in a numerical value. Measuring the accuracy of the rank of the recommendation can be done by obtaining the Average Precision, $\text{AveP} = \frac{1}{11} \sum_{r \in \{0,0,1,\ldots,1.0\}} p_{\text{interp}}(r)$, where $p_{\text{interp}}(r)$ is a smoothing interpolation and is defined as $p_{\text{interp}}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$. Iterating over 11 evenly spaced r recall values from $r = 0$ to $r = 1.0$, $p_{\text{interp}}(r)$ takes the largest precision value up to the r recall in the ranked list. When performing multiple queries, such as producing ranked recommendation for multiple users, a Mean Average Precision (MAP) measure is used to gauge the performance and is defined as $\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$. Another metric to evaluate a ranked list is the Discounted Cumulative Gain (DCG), which works on the principal that item usefulness means relevant items should higher up in the list and that a *gain* should be reduced at a logarithmic scale as the rank of relevant document slips down the list. DCG is defined by the formula $\text{DCG}_\text{p} = \sum_{i=1}^{p} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$, where p is DCG calculated for a rank placement p, rel_i is a binary indication of relevancy (0 = non-relevant item, 1

= relevant item). DCG produces a scalar value for rank performance of a model based solely on the list and rank of the item. When comparing different recommendations of non-fixed length lists, DCG cannot be used because the cumulative nature will mean that longer lists can potentially have larger values despite being worse overall. For this reason, Normalized DCG is used to remove the effects of cumulative gains from irregular length of lists. This is done by dividing the DCG by the ideal DCG. Ideal DCG is defined as $\mathrm{IDCG_p} = \sum_{i=1}^{|REL|} \frac{2^{rel_i}-1}{\log_2(i+1)}$, where $|REL|$ is the ranked list of length p containing only the relevant items. Putting it all together, NDCG is defined as $\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$.

Chapter 3

Methodology

3.1 Introduction

Recommender Systems, as the name suggests, have the job of recommending content to users that they will find interesting. Recommender systems are employed in various applications, and one of the common application is a Social Networking site. Datasets scraped from sites like Twitter and Weibo can be dense as there's no shortage of user participation in these popular social media sites [11]. Reference literature from Section 2 shows that feature selection has a positive impact on the final recommendation, so a dataset with multiple features is ideal. While our dataset contains various features such as 'follows' (where one user can opt to keep being notified of another user's new posts) and 'likes' (where one user likes the post of another user), the data for these features

is unreliable as it contains a lot of noise, is heavily sparse and for the purposes of our investigation unusable, hence we decide to use two features that we consider reliable - the posted microblog text along with comments that users may have exchanged on those posts. As the dataset is mainly composed of text features, the amount of activity in text posts far outweigh the amount of *likes* activity, making it very sparse. Using comments to in lieu is more reliable as, not only is it more dense, but also writing a response suggests interest more reliably.

In this work, we compare competitive techniques on a mid-sized, relatively sparse novel social media dataset and analyse whether traditional recommender systems can produce reasonable predictions and TopN recommendations. In the next chapter, we analyse the results from these experiments.

3.2 Approach Overview

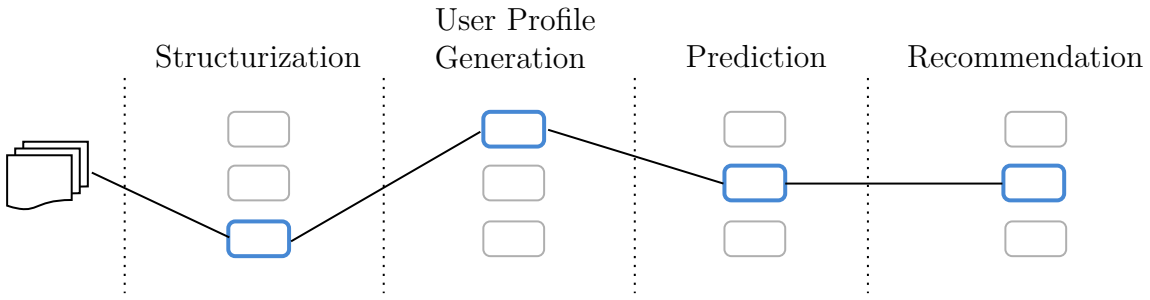


Figure 3-1: Our evaluation pipeline gives us the flexibility to configure it at any stage.

Using a novel dataset, as well as a gold-standard set of ratings for a small subset of that dataset, our goal is to propose post recommender systems and analyse the

efficiencies and deficiencies. Accommodating for a dataset with two features, we compose recommendation systems using proven techniques. Unlike Twitter, the MOTOR social media site is in its infancy and does not yet posses (in adequate quantities) rich, explicitly defined data that aids in building a social graph - aspects such as *likes* and *follows* [91]. With textual posts as the main reliable features to consider, our methods are designed as hybrid recommendation systems that rely on the post content to extract representational models. The steps in our evaluation pipeline are thus defined as:

- **Structuration** step defines how each model will process the item texts and represent it in vector space.
- **Profile Generation** step defines how each model uses item representations to generate user representation. This is always mean all of posts (vector) from the user.
- **Prediction** step outlines how CB (Cosine Similarity) and CF (ALS) variants of the model predict ratings. Figure 3-2 shows the cascading RS used in our experiment; cosine similarity between user-post vectors from CB RS are treated as ratings in the CF RS. Details are given in section 3.3. As another variant of experiment, we also substitute maximum rating of 3 if a user commented on a post the system is predicting similarity/rating for.
- **Recommendation** step is sorting and filtering to obtain TopN recommendation for each user

3.2.1 Preprocessing

Prior to the pipeline, however, Posts' (interchangeably referred to as *content* and *item*) text is filtered to remove any non-English syntax such as a hypertext link and Unicode emoticons. In this data pre-processing step, we perform feature selection and first select active users - this process is in line with [84] where qualifiers are used to reduce the initial dataset to predict more accurate ratings. For our qualifiers, we define active users as having decent participation rate in the social media site, more specifically a post count of 30 or above. The arbitrary value of 30 is found by tuning through trial and error, as lower values produced larger yet sparser datasets which negatively impacted the recommendation results, while larger values did improve performance and accuracy, the gains were almost negligible. Many users of the MOTOR platform share hyperlinks to images in the form of albums or hyperlinks to external articles, they also use emoticons; Unicode that modern GUI interfaces substitute with a render of a "smiley face" graphics. These artefacts are removed from the post's text. Stemming [71] is then used to substitute term variants down to their root terms, eg. ⟨Playing, Plays, Played⟩ → ⟨ Play ⟩. Furthermore, any remaining punctuation marks are removed and all terms are lower

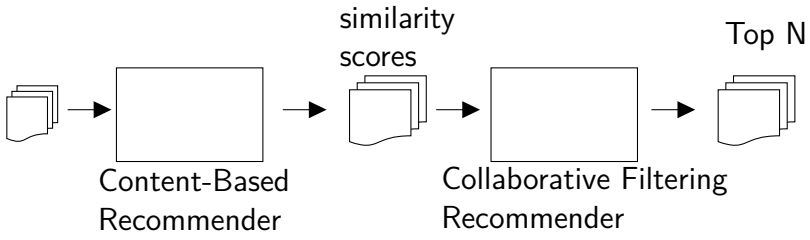


Figure 3-2: Cascading Hybrid Recommender System Topology.

cased to remove case sensitivity. Following this process the post's text is considered 'sanitized'.

3.2.2 Comments

Previously we stated that the dataset has two reliable features: post text and comment text. As part of implicit feedback of content-based recommendation systems, comments can be used to indicate a level of engagement between a user and a post. The fact that the user took the initiative to leave a comment on a post suggests the user is interested in the content. As such we run parallel experiment for each model below where we substitute a positive rating in any instance where a user has left a comment on a post. For example, if user A has commented on post #288, then when finding similarity score between User A and post #288, we would short-circuit a positive rating (of 3) to suggest the post is interesting to the user.

Starting in Section 3.3 we set the pipeline's structuration to our term frequency based model, following that we evaluate topic modelling in Section 3.4 by changing the structuration segment to topic modelling and lastly we wrap up in Section 3.5 by analysing the text using word embeddings.

3.3 Term Frequency Based Model

We start with exploring the first of three techniques evaluated. Our input, as already explained, consists of post identification number, user identification number and the

textual content. The evaluation setup is as follows:

- Preprocessing: Data is sanitized and features selected to reduce the overall size
- Content analysis: Structured item representation based on term frequencies
- User profile generation: Based on item vectors
- Prediction: (1) Use similarity measures, and (2) Use MF to predict rating

In this method, we create a baseline recommender system by using the vocabulary terms' significance as the features describing every post. Once the preprocessing is complete, post's text is transformed into structured representation. We do this using the term weighting technique described in [73], and as described in section 2.2, the TD-IDF algorithm's main goal is to associate a weight to a term with respect to a post. When we can describe a post using the weight of its member terms, we can then group similar posts together. So in our system, terms are built into a corpus-wide vocabulary and the term frequency per post is counted using the following expression $tf(t,d) = \frac{f_{t,d}}{|\text{words in d}|}$. This expression takes in the term and the post and returns its relative frequency - more frequent words will have a higher value than those that occur less, but naturally very common words that occur in many posts may have the highest tf values. To counter that, TF-IDF, is a product of TF and the inverse-document-frequency, which aims to apply a penalty to words occurring throughout the corpus, as those are likely not specific to a subset of post in the corpus. The entire expression is shown in Equation 2.10. TF·IDF function takes a post and a term as parameters and returns a value between 0 and 1, this value describes the weight of the term in the post at hand. A common technique (as

described in [18]) is to create a feature-descriptor of the post using the term weights as vector values. The length of the vector is equal to the size of the vocabulary, such that if

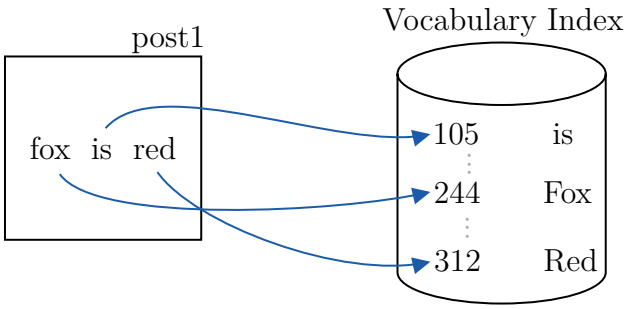


Figure 3-3: Terms are mapped to a corpus-level vocabulary index.

the vocabulary dictionary has 500 words, then the length of the feature descriptor vector $|\vec{fd}|$ will be 500. For every term in the post, the tfidf value is determined and inserted into the appropriate vector dimension (which is the index of the term in the dictionary as shown in Figure 3-3). In the example from Figure 3-3, the tfidf value maybe as such: $tdidf(fox, post1) = 0.80$, $tdidf(is, post1) = 0.02$, $tdidf(red, post1) = 0.30$. From this we can build the following feature descriptor, assigning a value of 0 as default for any term not found in the post yet part of the overall corpus vocabulary:

$$\vec{fd} = \left[\cdots, \quad 0.02, \quad \cdots, \quad 0.80, \quad \cdots, \quad 0.30, \quad \cdots\right]$$

At this point in the pipeline, the items (posts) are represented as vectors. Since our aim is to recommend posts to users, we need to know user preferences/tastes [66]. In microblogging social networks, text posted by the user can be mined to determine useful information such as user preferences, as discussed in [34]. In the next step, we create a user model to describe the user preferences. As the item representation was a term-

weight vector, it's simple and effective to represent that user-model in vector-space as well. The user vector $\vec{u_k}$ is a normalized sum of all the feature-descriptors of the posts the user has published: $\vec{u_k} = norm(\sum_{i \in D_k} \vec{fd_i})$ where k is the user, D_k are the posts published by user k and $\vec{fd_i}$ is the feature-descriptor for post i. Normalization ensures the term-weights are between 0 and 1.

3.3.1 Matrix Factorization

As explained in chapter 2, Matrix Factorization decomposes a matrix into its components in the latent space, which is usually of lower dimensions. The product of the decomposed matrix, $\hat{A}$ is an approximation of the original matrix A, $\mathbf{A} \approx \hat{\mathbf{A}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$. While this equation is for SVD, we utilize ALS method [6] due to its computational efficiencies in parallel environments, to decompose into two components $A \approx R = \mathbf{H}^T\mathbf{W}$. If A is a user-item matrix, then the component H can be thought of as the decomposition for user latent factors, while the W component can be thought of as the latent factors describing items. Starting with a parameter d as the dimension of the factors, H is a matrix of dimension $M \times d$ while matrix W is of dimensions $N \times d$, where M is the number of users from A and N is the number of posts in A. ALS is regularized regression algorithm that attempts to fit H and W to the data by minimizing the expression $\min_{x_*,y_*} \sum_{r_{u,i}} \left(r_{ui} - h_u^T w_i\right)^2 + \lambda \left(\|h_u\|^2 + \|w_i\|^2\right)$. Yet the expression is not convex, a minimum is difficult to obtain. As such, ALS algorithm solves for one component at a time, effectively making a convex problem. ALS algorithm repeats until convergence

Algorithm 2 ALS algorithm [1].

procedure ALS(d, R) ▷ d: # of factors, R: user-item matrix to decompose
  Initialize H, W
  repeat
    for $u = 1...M$ **do**

$$h_u = \left(\sum_{r_{ui} \in r_{u*}} w_i w_i^\top + \lambda I_d \right)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} w_i$$

    end for
    for $i = 1...N$ **do**

$$w_i = \left(\sum_{r_{ui} \in r_{*i}} h_u h_u^\top + \lambda I_d \right)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} h_u$$

    end for
  until convergence
  return X, Y
end procedure

occurs, and in it, user component is solved for, while items are held fixed, and then users are held fixed while items are solved for, algorithm 2(Reza Zadeh, Standford, 2015) shows this in action. The value of λ is a regularization parameter and determined by cross validation [36]. The components optimized by the algorithm contain the latent factors describing each user and each item. To obtain missing ratings from components, they can simply be multiplied with each other $R \approx \mathbf{H}^T\mathbf{W}$ or, for more performant operation targeting individual users, $r_{ui} \approx h_u^T w_i$.

In order to generate a recommendation for a user of this social network, we configure our pipeline for various methods. Our first technique is to find cosine similarity between the user preference vector and the post items not submitted by the active user, $simScore(\vec{u_{active}}, \vec{fd_{u_{active} \notin u}}) = cosineSim(\vec{u}, \vec{fd})$. Posts made by user themselves are

automatically given the maximum similarity score of 1.0 (which is eventually mapped to a rating of 3.0). When this is run for every user against every post in the system, we are left with a product of $M \times N$ scores, where M is the number of users and N number of posts. In recommending for user u, we can filter out user's own posts, and any post the user has already seen, rank the items by the scores and return the top N. To evaluate the recommendations, we treat the gold-standard ratings as the test set. Masking the system to return the scores for a randomized subset but also including the items in the test set, we can compare the recommender system performance against a known gold rating. Alternatively, we also evaluate the performance of the Matrix Factorization technique in a cascading-hybrid approach by combining CB and CF RS [58] (topology shown in Figure 3-2, where the CB similarity scores form the rating for CF ALS based recommender). As there are no provision for user-specified explicit ratings for posts in the MOTOR social network, we assume a straight linear naive mode for translating from similarity domain to rating domain; as the ratings are between [0, 3], we multiply similarity score with 3 to obtain the rating, $\hat{r_{ui}} = sim_{ui} \cdot 3$, where $\hat{r_{ui}}$ is the rating for the cell in the user-item matrix for user u and item i. Using CF RS in combination with CB RS (a form of hybrid RS [15, 53]) is known to mitigate to some-degree the weaknesses (such as the specialization/serendipity problem with CB RS and the sparsity/scale problem with CF RS) of either RS. In our case, we combine the similarities obtained from the CB RS as ratings into the CF RS and then do item-item recommendations, in doing so, we hope to address the cold-start and sparsity problems affecting CF RS, as well as tackling the serendipity problem affecting CB RS. A complication of using CB RS

to populate the user-item matrix is that it results in a dense matrix. We use a basic sample algorithm defined in [46] to sample the dense matrix, while retaining similar level of accuracy, while improving computation. The algorithm randomly samples a subset of the rating matrix, applies the MF and repeats the process several times. The mean of the predicted values form the result.

3.4 Topic Model

The next technique uses topic modeling to discover hidden relations within the microblog posts. The inputs remain the same as they were in section 3.3: consisting of item identification, user identification and the textual content. The evaluation setup is as follows:

- Preprocessing: Data is sanitized and feature selected to reduce the overall size
- Content analysis: Structured item representation based on topic *strengths*
- User profile generation: Based on item vectors
- Prediction: (1) Use similarity measures, and (2) Use MF to predict rating

Preprocessing step stays consistent with section 3.3 where data is sanitized to ensure removal of inactive users as well as purging of redundant and unnecessary words.

For content analysis, Section 3.3 used TF-IDF to create vector representation of the item. In this section, we also create a vector representation of the post, however

using Topic Modeling to generate post-topic mixtures. The Latent Dirichlet Algorithm, as described in section 2.2, works on the assumption that the observable words in a post, are the result of a latent topic distribution. The aim of LDA is to train on the words (interchangeably: terms) on a post text (the generalized concept is to treat words as dictionary indexes so that the algorithm can be applied to any numerically represented system), and indeed the words at large in a corpus. The algorithm is first provided with k (number of topics assumed for the corpus), α (prior parameter to designate topic weight for each post, usually the same for all posts and topics), β (another parameter of prior weights for topic weight onto every word, usually the same value for each topic and word), M posts in the corpus and N_d the words in post d. The training process estimates three probability distributions; θ post topic distribution (a k sized vector for each post arranged in a $M \times K$ matrix) this mixture learns the topic strengths for each post, φ word topic distribution registering topic affinities for each word in the corpus (a k sized vector for each word arranged in a $V \times K$ matrix, where V is the number of words in the corpus) and Z identifies the topic for each word in each post (a $M \times N$ matrix where each element range is between $1..K$). When viewing the plate notation showing the forward generative process in Figure 2-3, we can build the joint probability as

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P\left(W_{j,t}|\varphi_{Z_{jt}}\right) \quad [9] \quad (3.1)$$

Equation 3.1 models the joint probabilities observed in Figure 2-3, yet as the calculation for $p(x)$ in $p(y|x) = \frac{p(x,y)}{p(x)}$ is intractable, [9] use variational inference to estimate it. This

is done by treating it as an optimization problem and minimizing the Kullback-Leibler (KL) divergence (which takes two distributions and measures their divergence), defined as $D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$, where $Q(x)$ is the estimate and$P(x)$ is the actual observed. More concisely, through statistical properties and KL divergence derivations, we are able to find an expression that acts as a lower bound on $p(x)$, known as the Evidence Lower Bound (ELBO). This means that the expression can be maximized to get as close to the value of $p(x)$ as possible by using $q(y)$. The model $q(\theta, z, \varphi|\gamma, \phi, \lambda)$ is used to estimate the distributions observed in $p(\theta, \mathbf{Z}, \varphi|w; \alpha, \beta)$. From here an iterative algorithm that updates the distributions $q(\gamma)$, $q(\phi)$, $q(\lambda)$ and $p(\beta)$ is used to solve until convergence. The result of this algorithm are the various distributions, however, we're interested in the post-topic distribution which assigns every post with k length vector, with every element in the vector being topic strength for that post. Similar to the way we represent the post using tf-idf weight in a vector space, here we represent the post in a vector space using the topic strength.

$$\vec{fd}_{T=1..k,d=1..M} = [topicStrength_{1,d}, topicStrength_{2,d}, \cdots, topicStrength_{k,d}]$$

The user model generation process is similar to the previous section. When it comes to tastes, a user model with the post vectors all summed up and normalized will produce a vector that summarizes the user's interest based on the topics they've posted about. The remaining pipeline is configured similar to how it is configured for Frequency based model in section 3.3: in the prediction stage, we can continue employing cosine similarity measure to find posts similar to the user, as well as using Matrix Factorization

to predict missing ratings for recommendations in a similar way as described in Section 3.3.1.

3.5 Neural Network Word Embeddings

We have used Term Frequency techniques to find similarities between posts based on term weights. We also used topic modelling to find latent topics with which we could find similarities between posts. Next we employ another widely used method [61, 65, 59, 28], word embeddings, for mining the posted content. This technique trains a neural network using the data from the corpus, specifically, embeddings - a word and its textual context. The resulting neural weights becomes the vector that identifies a word in vector space. Related words are observed to be *nearby* (in Euclidean distance), paving way for mining similar contextual and semantic words.

- Preprocessing: Data is sanitized and feature selected to reduce the overall size
- Content analysis: Structured item representation based on VSM from NN weights
- User profile generation: Based on item vectors
- Prediction: (1) Use similarity measures, and (2) Use MF to predict rating

For data preprocessing step, lemmatization reduces sparsity (as there are fewer number of terms), so we'll continue to process the data as we have done in the previous two sections. Next, to generate item representations, we use word embeddings context described by Mikolov [59], in particular the model known as Continuous Bag of Words

(CBOW) shown in Figure 2-4. CBOW is a supervised learning NN algorithm which uses the target-word and context-words pairs $(w_t, [w_{t-\frac{k}{2}}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+\frac{k}{2}}])$ to train, where target-word w_t is the expectant result of the network, for the *context-words* input.

In CBOW, every word is represented by a *One-Hot Encoded* vector of size V, the total

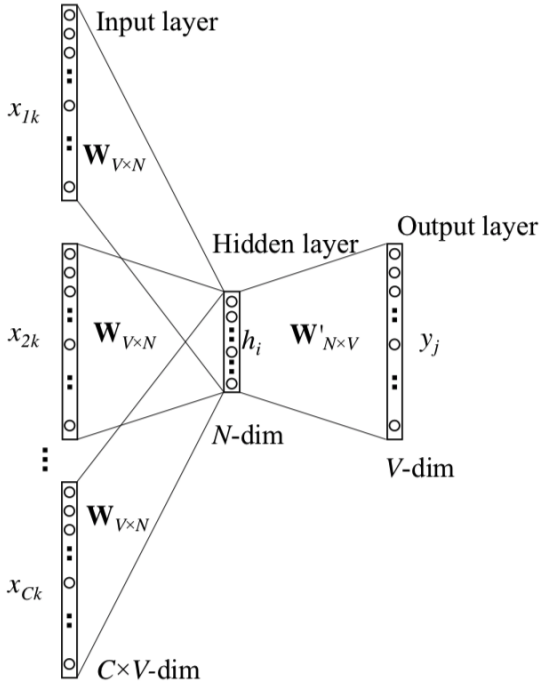


Figure 3-4: CBOW algorithm [75].

number of terms in the vocabulary. Its position in the dictionary marks the index in the vector where value is set to 1, else 0. For example, if the word is *red* and $V = 10$ and the position of the word is 4th in the dictionary, then the vector representing the word will be $\vec{red} = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$. For the NN topology, the input layer is of size $C \times V$ (where C is the number of context words) and the hidden layer of size N (arbitrarily selected to represent the word vector) and the output layer of size V. The activation function used in the hidden layer is the linear identity activation function $f(x) = x$,

while the output layer has the non-linear softmax activation (commonly used for output layer in NN classifiers). The usually larger input layer maps to the hidden layer through an averaging process where the vectors for all the context words are summed and then divided by the number of context words. The concept of the topology is to yield the target word as the output, when given content words. For training purpose, the error of predicted and expected output have to be minimized through solution of a convex optimization through gradient descent.

$$\begin{aligned}
\overline{\mathbf{x}} &= \frac{\sum_{c=1}^{C} \mathbf{x}_c}{C} \\
\mathbf{h} &= \frac{1}{C}\mathbf{W}^T \left(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C\right) = \frac{1}{C}\left(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C}\right)^T \\
\mathbf{u} &= W'^T \mathbf{h} \\
\mathbf{y} &= \mathbb{S}\,\mathrm{oftmax}(\mathbf{u})
\end{aligned} \tag{3.2}$$

In equation 3.2, $\mathbf{h}$ is a product of the weight matrix $\mathbf{W_{N\times V}}$ and the average vector, but can be simplified to just the average of all C context-words ($\mathbf{v_{w_{1..C}}}$), since the activation function is linear, each weight between the input layer and the hidden layer is just the vector unit, hence all $\mathbf{C}$ can be averaged and copied into the hidden layer. Matrix $\mathbf{W'_{N\times V}}$ represents the weights between the hidden layer and the output layer. Using these weights, the input to the output layer can be calculated as shown for $\mathbf{u}$, which then is fed into the activation function y. The activation function is $\mathbb{S}\,\mathrm{oftmax}$, which is a log-linear classification model [75] and that can be modelled as a posterior distribution $\mathbb{P}\left(w_o | w_{c,1}, w_{c,2}, \ldots, w_{c,C}\right)$. Given the context-words, what is the probability of classification of target-word being w_o? The loss function that is run through gradient

descent to optimize the weights is then

$$\mathcal{L} = -\log \mathbb{P}\left(w_o | w_{c,1}, w_{c,2}, \ldots, w_{c,C}\right) = -u_{j^*} + \log \sum_i \exp\left(u_i\right) \tag{3.3}$$

where j^* is the position j in the output layer and of value 1. Since we assume all words are represented using one-hot encoding, the output word we're expecting will have all units be 0, except j^*. To minimize the loss function, NN will use the back propagation technique to learn from the data. As this is a multivariate probability, a partial derivative is taken of the loss function with respect to both $\mathbf{W_{ij}}$ and $\mathbf{W'_{ij}}$.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W'_{ij}} &= \sum_{k=1}^{V} \frac{\partial \mathcal{L}}{\partial u_k} \frac{\partial u_k}{\partial W'_{ij}} = \left(-\delta_{jj^*} + y_j\right) \left(\sum_{k=1}^{V} W_{ki} \overline{x}_k\right) \\ \frac{\partial \mathcal{L}}{\partial W_{ij}} &= \sum_{k=1}^{V} \frac{\partial \mathcal{L}}{\partial u_k} \frac{\partial u_k}{\partial W_{ij}} = \sum_{k=1}^{V} \left(-\delta_{kk^*} + y_k\right) W'_{jk} \overline{x}_i \end{aligned} \tag{3.4}$$

Where Kronecker delta, $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$ is a value of 1 if $i = j$, else is 0. Finally the weight matrices are updated using the expressions $W_{\text{new}} = W_{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial W}$ and $W'_{\text{new}} = W'_{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial W'}$, where η is a positive value learning rate. This algorithm is repeatedly run until convergence occurs - where no significant changes are observed after a round of optimization. The CBOW embeddings algorithm optimizes the W' matrix, whose column k is the vector fd^T identifying word k location in vector space. In order to obtain post vectors, we can either sum up all the term vectors in the post and normalize, or as we have done, modify the back propagation learning model as described in [51]. A CBOW extension to create post vectors is that the word window grows by one to include a unique post/paragraph identifier, known as document embeddings. It is referenced by

the Matrix $D_{G\times N}$ (where G is the number of posts) along with the pre-hidden-layer weight matrix W. The k^{th} post vector is the k^{th} row in matrix D.

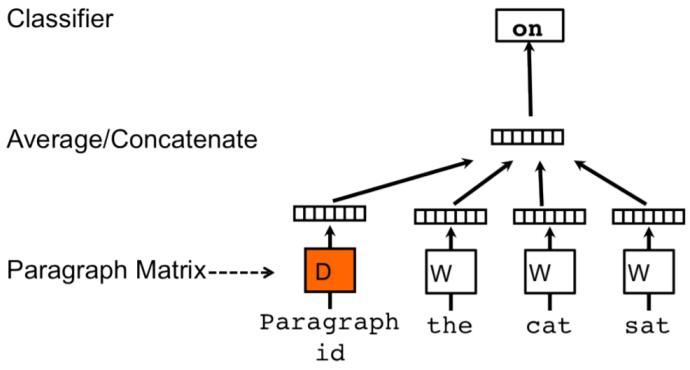


Figure 3-5: Distributed Memory Paragraph Vector Extension to CBOW as described in [51].

A post vector represents the post in the vector space and the remaining steps are also similar to the previous sections. We generate the user taste-profile as a vector, which is then used to find similarity scores between other posts. We also use Matrix factorization, similar to previous steps, to generate predictions for recommendations.

3.6 Summary

We have outlined the models and variants we employ. As our data is entirely composed of text, we have opted to use models proven to be robust against text corpus. Once posts are modelled and represented in vector space, we model the user profile. We can then find nearest neighbour or progress to matrix factorization to observe the latent

behavior between user and post. As rating boost, comments are leveraged as indicators of extreme interest in a post (after all, user took the time to engage), we expect comment boosting to improve accuracy since it is an implicit feedback of user interest. We expect term-frequency to perform well as terms describing motorcycles, events, riding routes and locations can lend robustness to vector space similarity. We expect topic modelling to introduce enough differences between topics to neatly cluster many similar posts; since its in micro-blogs' nature to generally have one topic per post, posts gravitating strongly towards a topic would likely be considered similar. Lastly, the word embedding model is considered to model latent 'aspects' of the corpus, rather than the dictionary - hence the hidden layer size can be arbitrary. Word embeddings analyse the grammar structure and the term's role in the syntax structure as the CBOW window moves down the post's text. In short text settings we expect the system to underperform, yet still maintain relatable similarities due to MOTOR blogs using same vocabulary and structure to describe similar concepts (such as motorcycles brands, conventions and locations).

Chapter 4

Evaluation

In Chapter 3 we examined the setup for the different techniques. This chapter explores the evaluation results of our experiments. In the first section we describe the dataset we used and in subsequent sections we examine the results from each technique.

4.1 Dataset

As previously mentioned, the dataset is from a social media platform, and for the purposes of this work we will refer to the dataset as MOTOR. The social media platform is focused on, and geared towards the niche of motoring community and enthusiasts. The website provides news, reviews and information on upcoming events in the world of motoring - each of these actions is considered a 'post'. Furthermore users are encouraged

to sign-up so that they can share content themselves. Users are able to post content in the form of blog posts - such as picture album from recent motoring circuits and tours or opinions on certain models of vehicles, and they can also leave comments on other posts.

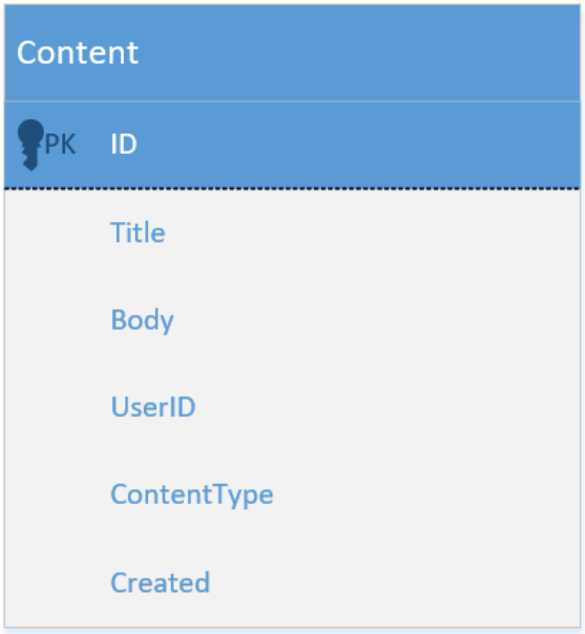


Figure 4-1: Data Table Layout.

Structurally, the data is stored in tables in the MySQL relational database, but all the user posted data is stored in one table labelled 'Content'. The UML layout of the table *Content* is shown in Figure 4-1. The table has the following columns:

- **ID** - The primary key associated with every post. Also the identifier we use to track the post in our algorithms
- **Title** - The users are asked to title their posts when submitting content
- **Body** - The column containing the textual data on which the algorithms perform

their operations

- **UserID** - The foreign key identifying the user who submitted the content. This is the identifier we use to track the poster as well.
- **ContentType** - This attribute identifies what type of post this is. Discrete options are: BlogPost, Review, Event, and Comment

Dimension wise, the total number of posts in the table is 163,500, made by 18,872 users, the largest post featuring 4,473 words while the shortest posts are of length 1, whether it is a single emoticon or an embedded hyper-link to an image or an external resource. Figure 4-2 shows the distribution of the term count over a discretized post count, for example there are 27,178 posts that are between 0 to 50 terms.

Figure 4-3 shows the content of a sample blog type post that contains a single image. Textually, this information is not relevant as it cannot lend itself as any feature

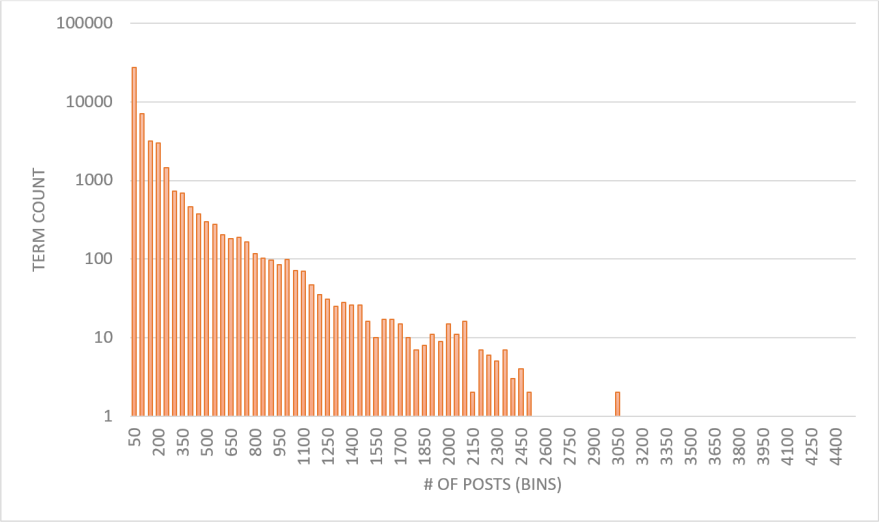


Figure 4-2: Distribution of the number of posts of discretized lengths.

[cid=38657,[IMG]dm_bbVKVNTVYl17TH8lgc4MQ1ePA6nzipdu.jpeg]

Figure 4-3: Sample post content from dataset: This post embeds an image the user shared.

This section of Route 14 from Gloucester Va. to Tappahannock Va.is really nice just watch out for the two 90 degree turns on the route!

Figure 4-4: Sample post content from dataset: This post shares a scenic route and warns of perils on the road.

of the post or user profile. If we omit such posts, the filtered post count drops to 46,498 generated by 6,240 users. Figure 4-4 shows a typical microblog-like post shared by a user, we'd like to remark on the terms "Route 14", "Va.", and "Va.is", as these create a need for disambiguity, text normalization and complex context-aware abbreviation expansion techniques [10, 56]. Figure 4-5 shows the frequency of posts against the number of users in this dataset, and from this distribution it is evident that majority of the users, 6,185, have generated between 0 and 50 posts.

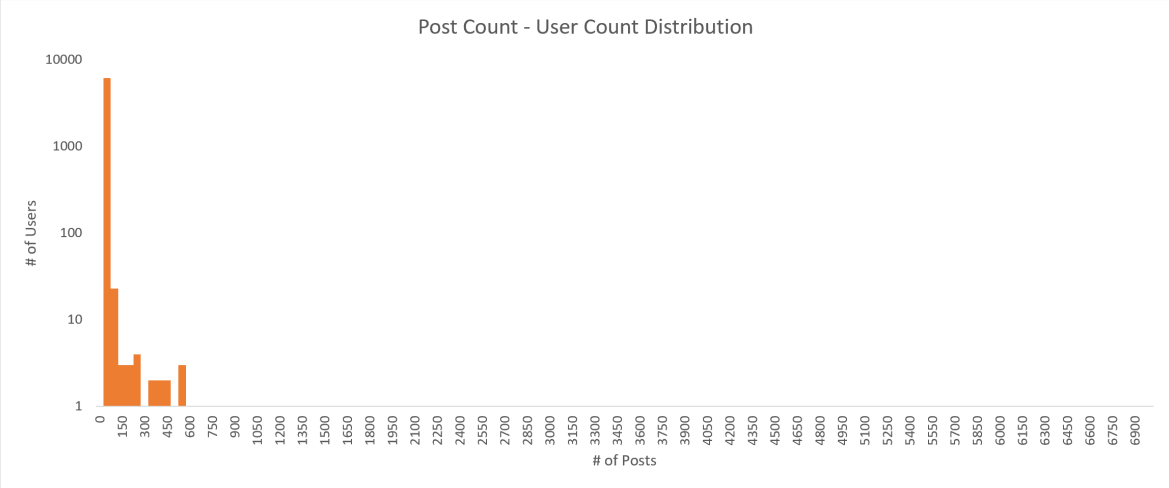


Figure 4-5: Distribution of number of posts by the number of users.

Zooming in further on the obvious hotspot between 0 and 150 posts, Figure 4-6 looks at discretizations between 0 and 200 and find that still, majority of the posting populace of the social media platform, about 5,750 users, have made between 0 and 5 posts. This lack of participation translates into sparse data and potentially inaccurate recommendations as the user profiles created from these few posts will inevitably be incomplete.

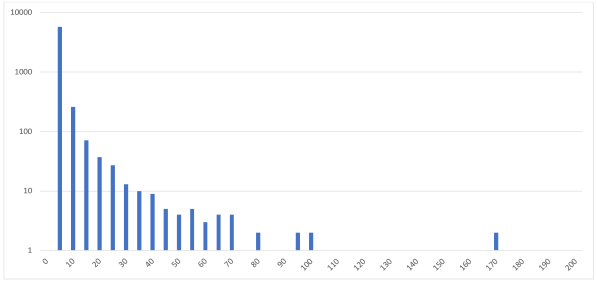


Figure 4-6: Distribution of number of posts by the number of users, zoomed to show post bins between 0 and 200.

From another angle, Figure 4-7 examines the concentration of posts by top 56 users who have made at least 50 contributions to the site. This group of users overwhelmingly dominate the site's activity space. Of remarkable note is that there is one user who has post count that more than doubles that next leading user. Together, these top 56 users have made 32,307 posts, accounting for nearly 70% of all activity on the website. This hot-spot attribute of this dataset introduce sparsity and cold-start issues for recommendations.

Exploring the dataset further, we examine the composition of the *content type* of posts for the top 56 users. Overlaying post frequency of comments and non-comment content types allows us to see the makeup of the dataset. Figure 4-8 shows the users

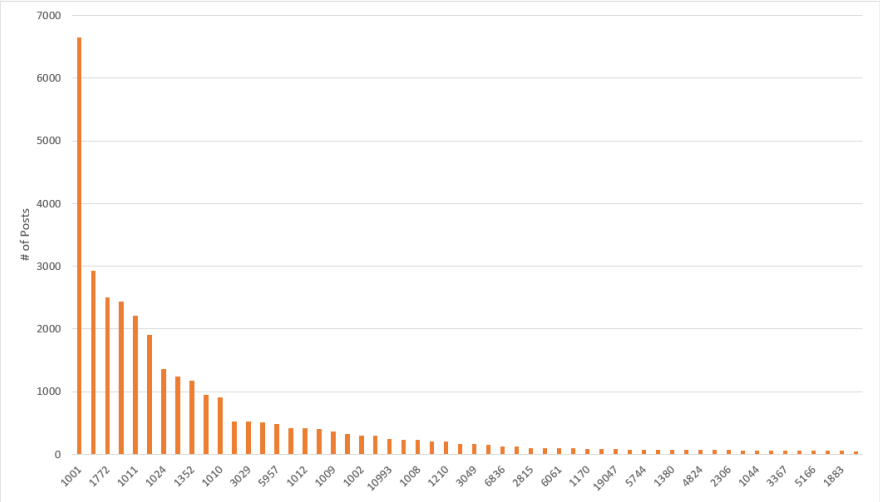


Figure 4-7: Frequency showing concentration of posts in top 56 users with post count of >= 50.

against comments and non-comment content, and immediately we see clusters of users that have never commented. Further investigation has led us to confirm that the top 4 users by post count are indeed bot accounts used by the social media platform to post official news, and event informations. Indeed in all, 12 out of the top 56 users are confirmed to be bot accounts, yet we have opted to include the posts of these bot accounts into the models. Posts made by these bots account for a significant segment of the entire site activity and as such contain posts that users may want to see. While recommendations will not be made for these bot accounts, their post representations will still be used to find similarity scores.

Comments in the dataset are between 1 word at the smallest sized, to the longest at 1012 words, with a standard deviation of 38.46 - suggesting a large spread. Indeed, Figure 4-9 shows the comment length distribution, with the largest group be-

1. Great ride. We need pics :)
2. Here's the onboard footage, which is much better than the first video! http://<redacted>
3. Hey is this mostly freeway or secondary roads?
5. Now, that is craftsmanship.
6. That is pure beauty. I want one. For my living room
7. The monkey needs a helmet!
8. Even in the banana yellow, it's a work of art

Table 4.1: Random sample of comments from the dataset.

tween nearly 1,900 comments of lengths between 5 and 10 words. This observation gives useful bounds of the nature of text content available in the form of comments. Table 4.1 shows a random sample of comments. The overlap between the gold-standard's positive ratings (rating >= 2) and comment-posted exists for 25 instances (out of the 420 gold standard ratings), suggesting that the users in this dataset might interact with comments on posts they deem interesting. This is in contrast to 3 instances of comments being left on posts where the user has given that post a rating of < 2 in the gold standard, an analysis of the comment sentiments (positive, negative, and neutral sentiments) agrees

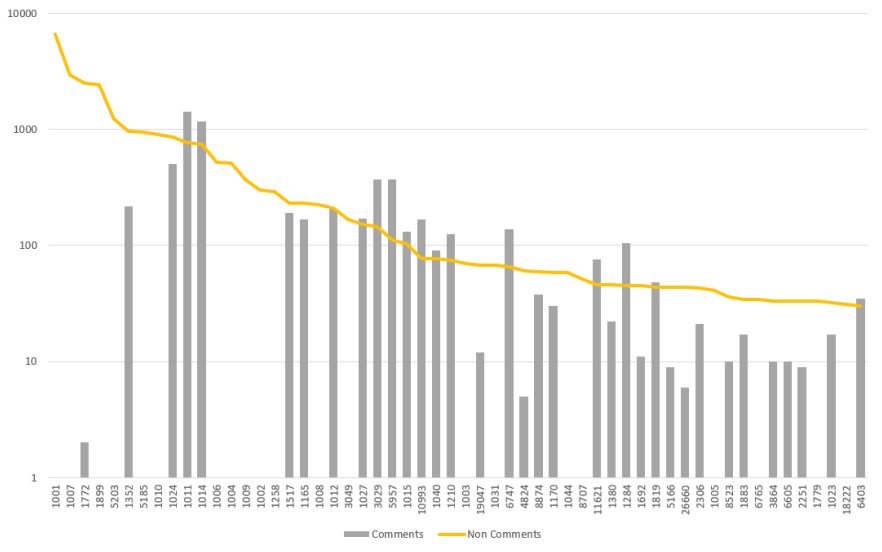


Figure 4-8: Top users post behaviour bisected between comments and non-comments.

with our findings; comments left on posts user rated high were generally positive or neutral, while the 3 comments left on posts rated low were generally neutral in nature. Figure 4-10 uses chord graphs to visualize the comments exchanged within the corpus, the key take-away being that most comments, like the posts, are made by the top active users, and minimal participation is present from users other than the top 10. This behaviour suggests that the top several users form a tight-knit community within the platform and they exchange comments with each other. In later sections we explore the comment attribute of this dataset and to what effect they can be utilized and whether that yields any benefits.

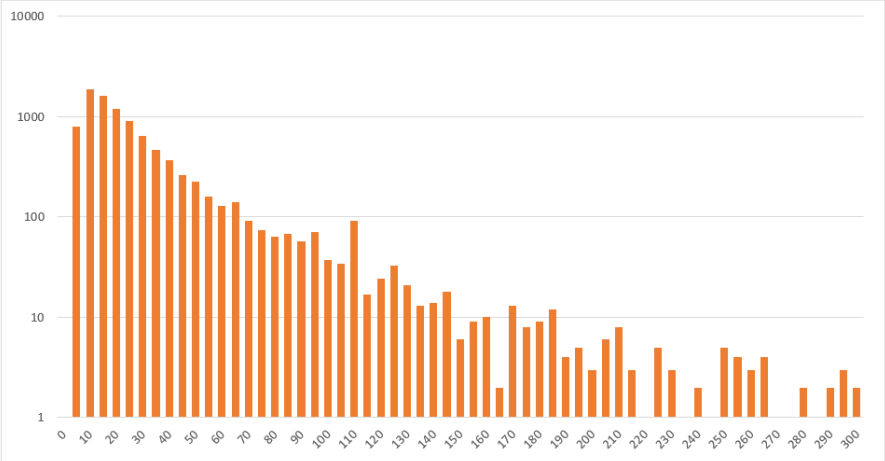
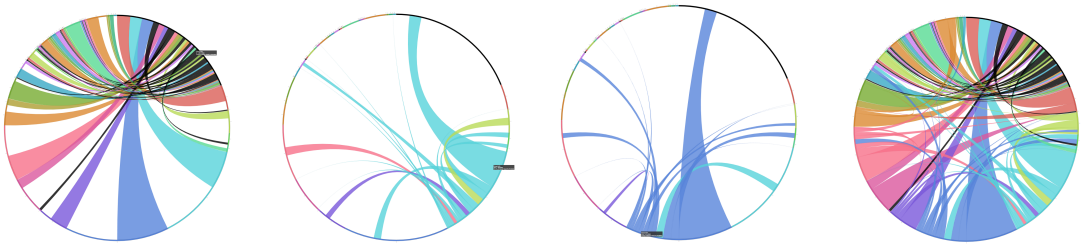


Figure 4-9: Distribution of the number of comments of discretized lengths.

We have reviewed the dataset behind the MOTOR social media platform. We examined the layout of the data structures and the type and samples of data available to us. We have seen evidence of data accumulating among handful of highly active users and some of those users are artificial. Further still we observe that the dataset is geared towards a particular niche; the motoring enthusiast and are likely to converse



(a) Comments from- and to- posts by 'other' users (black arcs show posting of comments by marginal users on posts by top users whereas color bands show top user's comments left on 'other' users' posts)

(b) Comments from and to user 1011

(c) Comments from and to user 1014

(d) Comments between all users

Figure 4-10: Comment exchange behaviour between sample users; showing frequent exchanges between top active users of the platform and very little external (other) participation, suggesting a small, tight knit community of top active users.

and interact on topics generally limited to the scope of motoring. The dataset posts by users are largely of few words, in traditional microblog sense, and would generally cover one topic [62].

4.2 Evaluation Methodology

Despite MOTOR being a social media platform, we are limited to offline evaluation of the predictions and recommendations. Optimally, the A/B paired tests, where user is shown recommendations and asked to select one that they prefer, are preferred avenues of recommendation evaluation as the user themselves mark the more interesting post [27].

Post ID	User A	User B	User C	User D	User E	User F	User G
116331	2	1	2	3	1	2	1
109561	2	2	2	2	2	2	2
...							

Table 4.2: Sample from the ground-truth rating file accompanying the dataset.

A similar observation can be made by looking at which recommendation, if any, the user engaged with through usage metrics. Yet both types of evaluations are not available to us for this experiment. As the dataset contains only text based usable attributes of the posts made by users, our evaluated methods model the text content of the posts and using these models make a prediction on which posts would be interesting to which user. In order to evaluate the performance and accuracy of the predictions, we need some ground truth of 'interterestingness'. Accompanying the dataset is a comma-separated value (CSV) file containing manual ratings by 7 users for 60 randomly sampled posts. The post ratings in this CSV file are between 0 and 3 and are normalized before being processed by our evaluation system. Table 4.2 shows a sample of the ratings provided by the users.

Having a gold standard with a rating scale between 0 and 3 means we can compare the ratings we predict for one of these 60 posts from our similarity methods and evaluate the error using squared difference techniques such as Root Mean Squared Error; these give us a picture of variance in our prediction. As discussed in Chapter 2, RMSE describes the accuracy of a model and we use it to compare the relative performance of the models against our dataset. RMSE cannot indicate a threshold of poor or favourable values by itself, however we do know that the maximum error between actual and predicted values

can be 3 (as the rating is between 0 and 3) and as such the upper bound of RMSE for our models can be 3.0 as the absolute failure scenario. Our end goal remains recommendation of content, so measuring ranking of prediction is another component of the evaluation. Using Mean Average Precision metrics, which combines both recall and precision in a single value, we can examine the ranked recommendations. We use the Mean Average Precision (MAP) metric to get an overall robustness of the recommendation. Both precision and recall are essential metrics when evaluating the accuracy or the query results (in this case query is what we ask the system to recommender for a particular user). Naturally the goal is to obtain a high precision and a high recall value for each model, yet both are inversely proportional [35].

4.3 Platform

The following describes the environment, platform and machines used in the programming, execution and result analysis of the experiments.

- Programmed using a mixture of Python 3 language and the Apache Spark framework version 2.0.1 (up to 2.4.3 most recent) on the Scala language.
- Data preservation used 'pickling' in python to serialize data at various stages in the procedure to not only start from a known last-success point, but to also reuse data in multiple experiment from known points in the process.
- Used the numpy, scipy, gensim, sklearn, matplotlib, mlmetrics, pandas, and the

nltk libraries to support the functional code.

- Post and Comment data was restructured from MySQL database in a relational form, to MongoDB in a condensed form. This allowed for a faster lookup as relational information was collapsed into a format that contained only the essential information in a single-row-per-post format.
- Machines: Desktop computer with 32 gigabytes of memory, 8-Core AMD FX 8350 processor, and 4 terabytes of hard-disk space, running Windows 10 operating system. Also a laptop with an i5-540M processor, 8 gigabytes of ram, 1 terabytes of hard-disk space, running Windows 10 operating system.
- Microsoft Excel was used to analyse the data and to generate the graphs and charts

4.4 Experiment Results

4.4.1 Term Frequency Model Results

For our first experiment, we had constructed a configurable and repeatable pipeline that would model the text of the posts using a term-frequency and inverse document frequency method. After tokenizing the entire corpus, there are 53,300+ terms, most of them are either fragments of dynamic URLs, which by their nature are unique and others are spelling variants of known words, such as *moto*, *motogp*, *motogpmatt*, *motomatt* - take note that this is after stop-words have been removed and stemming applied. In

this model, each post was represented by a vector of length the same as the size of the dictionary. Such large vectors are known to increase sparsity and decrease computational performance. However, as TF-IDF has robust performance on news and text-rich document where the full English dictionary can reach upwards of 171,476 words, we did not see 53,300 as a significant hurdle for the technique. We did employ preprocessing technique to discard terms unique enough that they were being used only 10 times or less throughout the corpus. This brought the dictionary size down to 9,800. Each of the vector units, associated with a term in the dictionary, would have the TF-IDF value in it. Figure 4-11 graphs the posts in the 2D-projected space, as visualizing dimensions over 9000 is not feasible, we project them down to 2D space to see any obvious relationships. However after such a steep projection, dimensional details are lost and any useful relational data may not be present in the graph. Yet it gives us the sense of spread of the posts in VSM and also confirms that bulk of the posts, of shorter lengths, having smaller distances are congregating near the epicentre, whereas larger posts are seen spreading away from the center.

When querying predictions for the test set (60 posts for 7 users), we evaluate two techniques, a cosine similarity (COS/CB) and a hybrid (cascading) recommender system using Matrix Factorization (MF/CF) based prediction. The runtime for the CB model took approximately 10 hours to execute, while the CF (MF) process took an additional 2 hours to run. Table 4.3 shows the results for both MF and COS based predictions. In a comparison between CB/COS and CF/MF for TF-IDF model, we observe that COS model presents overall better rating predictions as seen from the lower RMSE

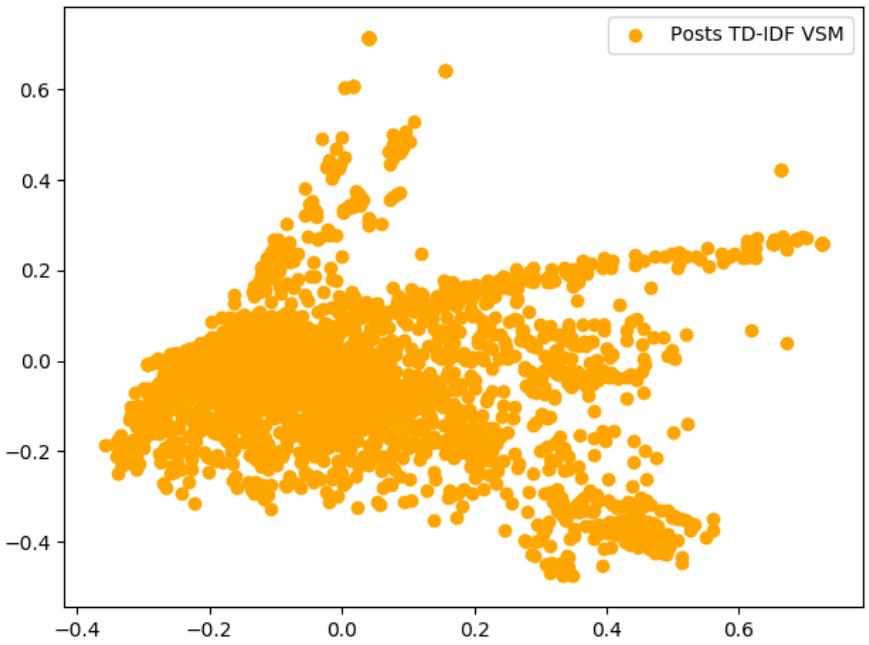
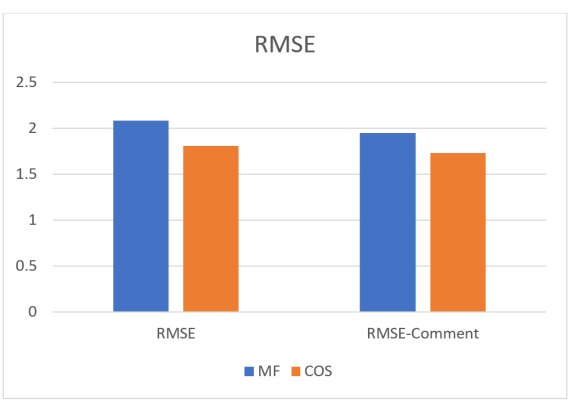


Figure 4-11: 2D projection of the TF-IDF vectors.

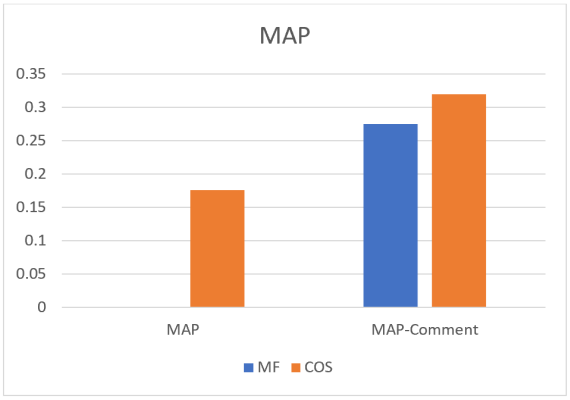
rates. Beyond prediction, we evaluate recommendations by post ranking. As previously outlined, the gold standard has a random sample of posts that have been explicitly rated by a sample of social platform users in an offline procedure. A recommendation of posts constrained to the ground truth set can use TopK rank comparisons. For this we use the Mean Average Precision metric which looks at the mean (for all 7 golden-truth users) of Average Precision (which itself is taken at 11 Recall steps).

Second row of Table 4.3 shows the MAP graph for the various model variants evaluated. MAP metric is the mean used to measure the recommendation's ranking for all the 7 users. At its core is the Average Precision metric which is essentially the area under the precision-recall curve (AUC), combining both precision and recall in a

	MF	MF-Comment	CB/COS	CB/COS-Comment
RMSE	2.0856	1.9493	1.8065	1.7320
MAP	0	0.2445	0.1764	0.3199



(a) Model RMSE



(b) Model MAP

Table 4.3: TF-IDF Prediction Results: Column MF is the Matrix Factorization results, MF-Comment is the technique where we substitute ratings if user left comment on a post, CB/COS is the cosine similarity model and CB/COS -Comment is, like the MF-Comment, where we substitute the similarity with a high similarity value

single value representation of accuracy. For this experiment, we observe that the MF based model fails to predict any correct ranking order as the mean average precision for MF is 0. The complete failure is likely due to the already poor results of the COS similarity being used as rating input to the Matrix Factorization algorithm, which in turn produces predictions that are all lower than a rating of 2.0; a rating of 2 or higher is needed to boolean-map for True-Positive predictions as required by MAP metric. In both the COS and MF tests, the comment-integration variants appear to produce lower error and higher recommendation ranking. This is likely due to the assumption that when users engage with posts through comments, they have found the content of the post *interesting*, so by using comment feature to augment predictions, we have increased the recommendation accuracy by 81% for the COS-based, as well as lowered the error

rates by 4% and 7% for COS and MF models, respectively.

4.4.2 Topic Model Results

Alongside our evaluation of a term-frequency model on our dataset, we evaluated recommendations based on topic modelling and topic vectors. Similar to TF-IDF's vector state representation of each post in the dataset, topic modelling would represent each post with its affinity to k-number of topics. Exploring the latent features of each post, Latent Dirichlet Allocation algorithm finds the common topics of the corpus. As the corpus is of considerable size (nevertheless sparse) the number of topics is a parameter best arrived at through cross validation or via dataset specific evaluations. As such we here show our recommendation evaluations over a range of k-topic values (10, 15, 20, 25, 30, 25, 40, 45); (hyperparameters of smaller and larger k values than the ones listed here produced considerably worse results). Note that where term-frequency possessed large vector dimensions, topic model has a relatively smaller representative dimensions. Topic modelling is a form of dimensionality reduction; representing large corpus with few latent descriptors. We expect it will create a dense vector model, however, it will also create topics that are very similar to one another, making posts appear very similar to one another. The runtime for each k-value model averaged 3.5 hours for the entire corpus - totalling approximately 26 hours.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	copi	combat	copi	copi	copi	copi	groov	superpol	copi	washoug
2	groov	13cal	groov	groov	entranc	absolut	copi	solut	entranc	cleveland
3	magazin	copi	kind	michel	absolut	unlock	encompass	mobil	wise	copi
4	shall	vote	motorcyg	doubli	newest	nineteen	wise	percentag	sert	michel
5	twitter	encompass	michel	concuss	pocket	mick	unlock	femal	upward	cbr600rr
6	sleepi	hidden	encompass	nobodi	necessari	upward	englishman	groov	shock	concuss
7	encompass	goat	femal	solut	upward	altern	femal	twitter	centuri	groov
8	wake	aoyama	unlock	superpol	softwar	entranc	jone	copi	groov	absolut
9	loss	groov	haggl	seem	shock	concuss	concuss	mick	shtml	adverti
10	femal	jone	82ama	lose	roll	loss	michel	kind	neck	acid

Table 4.4: Table of topics for the MOTOR corpus at 10-topic.

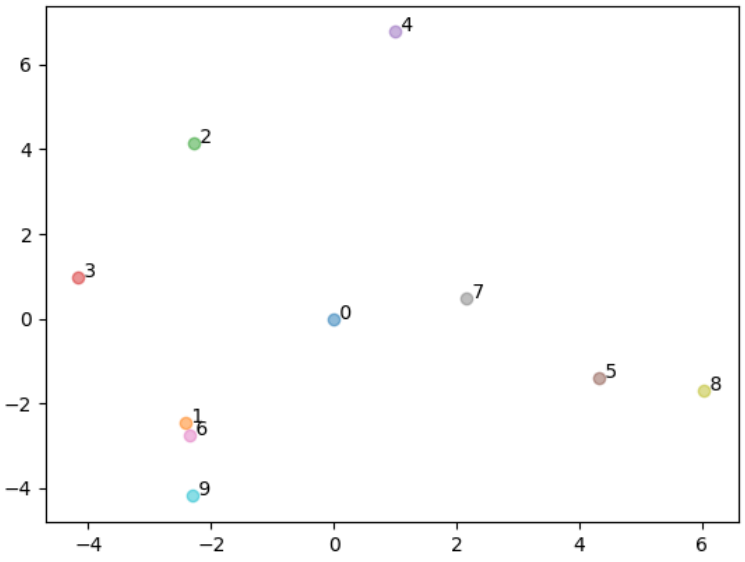


Figure 4-12: Topic spread as shown after a 2D projection.

Table 4.4 shows the topic distribution using terms from the corpus - all topics are visualized by 10 terms each. Each topic here is described as a vector $t\vec{d}^T$, where each of the vector's unit (here substituted by a term) has a weight for each of the terms listed. This visualization of the 10-topics variant of our evaluation has many of the same terms as members of each topic (observe the repeated terms in the neighbouring topics). While the topic terms in LDA may not particularly make sense at first glance, a well structured and balanced corpus will yield topic terms that present an observable theme and any outlier topic terms may be obvious [16]. An example of such cohesive topics is presented in Figure 4-13; human discernible topic list shows the various issues dealt with in wordy sources such as newspaper. The behaviour of similar terms in our corpus suggests that the corpus is centred around a topic already, as is indeed the case where MOTOR social

network is geared towards the motoring enthusiast. However, variations in topics can still be seen, take for example topic #2 which seems to be dealing with issues of combat, is remarkably different than topic #10 which seems to be dealing with geographical locations. Using principle component analysis (PCA), we can reduce and project the high dimensional representation of each topic down to 2D plot it in a scatter plot, shown in Figure 4-12. This shows the spread and confirms that the topics are not all together unique.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
city	nea	music	tv	senate
building	art	orchestra	film	house
park	endowment	jazz	show	budget
design	frohnmayer	symphony	television	congress
downtown	arts	opera	news	bill
art	artists	concert	channel	clinton
project	mapplethorpe	musicians	global	republicans
center	helms	concerts	http	appropriations
commission	grants	musical	icon	rep
public	funding	band	movie	federal
sculpture	grant	composer	night	committee
street	agency	blues	president	r
murals	congress	hall	war	republican
buildings	obscenity	classical	cbs	vote
space	chairman	piano	hollywood	spending
site	artistic	composers	star	funding
square	obscene	orchestras	pbs	gingrich

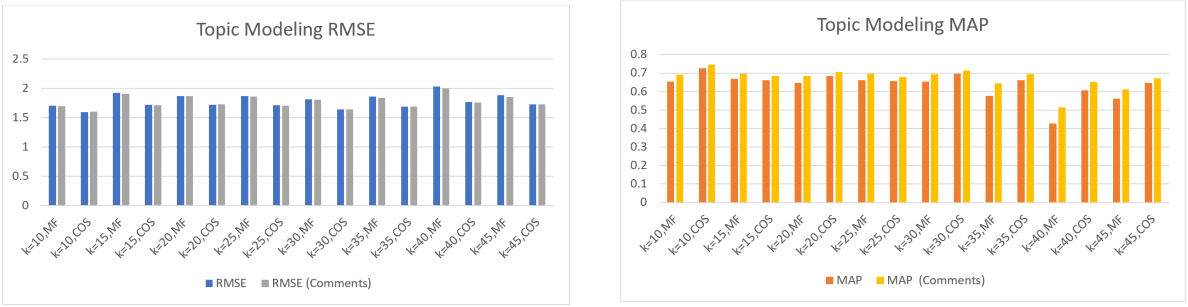
Figure 4-13: Topics as observed from topic modeling of wordy sources such as newspaper [24] - Example of a corpus with topic-richness.

For evaluating the prediction by the models, we again turn to the RMSE error metric, as it quantifies the error between what the user rated a post and what our models rated it on their behalf. Table 4.5 and Figure 4-14 show the RMSE of the predictions as well as the MAP of the recommendation. In the table, the columns are the individual metrics, and the row are the models, as an example, *k=10,MF* is the Topic Model with

	RMSE	MAP	RMSE (Comments)	MAP (Comments)
k=10,MF	1.710263	0.65489	1.695582	0.692213
k=10,COS	1.598141	0.726283	1.60447	0.745079
k=15,MF	1.923538	0.670236	1.910497	0.697287
k=15,COS	1.720308	0.661822	1.714406	0.684731
k=20,MF	1.870829	0.646716	1.870829	0.683458
k=20,COS	1.720308	0.684573	1.726189	0.705398
k=25,MF	1.870829	0.662513	1.864135	0.696373
k=25,COS	1.714406	0.65566	1.708484	0.679445
k=30,MF	1.81659	0.654553	1.809696	0.693851
k=30,COS	1.641934	0.695276	1.641934	0.713746
k=35,MF	1.864135	0.57729	1.837117	0.644879
k=35,COS	1.690594	0.661973	1.690594	0.694811
k=40,MF	2.03101	0.426893	2.00624	0.51487
k=40,COS	1.766811	0.607734	1.761065	0.652333
k=45,MF	1.884144	0.561793	1.857418	0.611555
k=45,COS	1.726189	0.646016	1.726189	0.671427

Table 4.5: Topic Model Based Recommendation Errors.

k=10 (ten topics) using Matrix Factorization for recommendation. In relative terms, the most performant variant of the topic models from the RMSE perspective was the 10-topic COS/Without-Comment model. Table 4.5 shows however, that the RMSE for k=10,COS with and without comments models being less than half a percent from each other; suggesting that addition of comments did not have a perceivable impact on the RMSE of predictions.



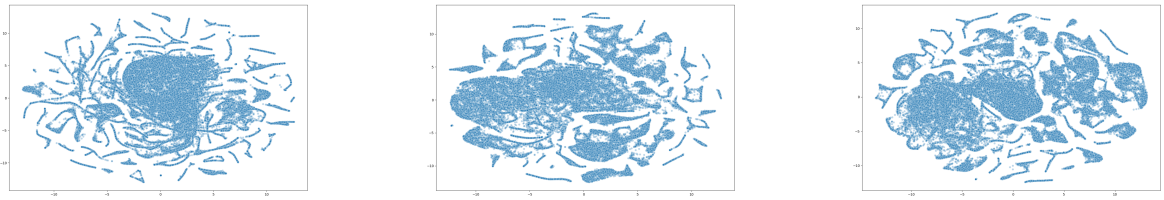
(a) Model RMSE

(b) Model MAP

Figure 4-14: Graphs showing RMSE and MAP for various sized Topic Models.

RMSE can be a measure of how well the model performed overall relative to others. However, recommendation systems have value-add when suggesting recommendations in a ranked order, so they need further analysis with respect to recommendation order. Once more we use the MAP metric to get an overall robustness of the recommendation by evaluating topic models of various topic-k variants. In Figure 4-14b the MAP value can be seen for the models evaluated, with the 10-topic, cosine similarity achieving the highest MAP within the topic model batch, an additional 2.5% gain was observed by integrating the comments feature. Topic Modeling results were indeed the most intriguing. All variants achieving nearly similar, and very high, MAP results and

the RMSE also hovering near the similar range. Despite larger topic counts being known to lead to many highly-similar topics clusters [32], we found that for the k values we were using, multiple similar-topic clusters were not an issue and the topic affinities for each user remained relatively intact as we increased k. In other words, as we increased the number of topics, the relative topic associations remained almost the same. Meaning the post representation vectors were different for each k, but maintained topic strengths that repeatedly lead to the same cosine similarities between user-vectors and post vectors. Ultimately producing similar MAP and RMSE results. An exception to this explanation was seen for $k = 40$ Matrix Factorization (MF) model, where the MAP values for with and without comment models are lower than the trend, and that extends to the RMSE as well, where the error is higher than the trend. This is regional trend particularly affecting the MF models starting from $k = 35$ and ending with $k = 45$. The Cosine models are also affected, but less so in intensity. Upon inspection of the topic-post probability mixtures, the increased number of topic count (k) affects the results such that some posts outside of our ground truth, being strongly associated with certain topics at lower k values, are then associated with many smaller (but similar) clusters (topics) at higher k. This phenomenon does not affect the COS model too drastically as most of the posts being compared for ranking are not affected by the smaller (but similar) clusters. However, in CF/MF, the ALS decomposition spreads the effect over to the ground truth posts being predicted. Hence, we see the effect for MF models between 35 and 45 k values.



(a) Post Representation Vectors at ||k||=25

(b) Post Representation Vectors at ||k||=250

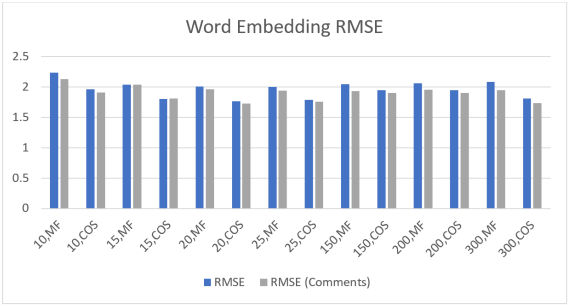
(c) Post Representation Vectors at ||k||=300, commonly used size

Figure 4-15: T-SNE of Post vectors of different sizes for Word Embedding models.

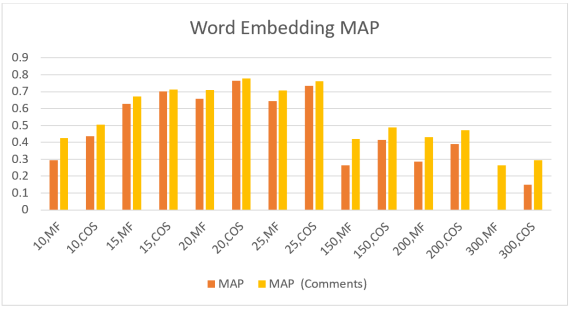
4.4.3 Word Embedding Model Results

Reviewing our third technique evaluation, we look at training Artificial Neural Nets (ANN) to predict word embeddings. From these we created a vector representation for each post in the dataset, then user profiles were created and similarity measures used to find recommendations. The ANN requires training on the corpus itself to produce the document vector representation and is an unsupervised model. This process is considered a black box as far as the intermediate variables are concerned and this data is difficult to obtain during the training phase and difficult to make sense from a human introspection point of view. As such we evaluate the errors and precision of recommendations as before, while configuring the size of the hidden layer of the neural net (as part of hyper-parameter tuning), which ultimately dictates the vector dimension of the post representation. Word Embeddings model defines k-concepts (where k is the size of the hidden layer and the post vector representations) from the entire corpus vocabulary. As we alter the dimensionality of the vector we can sparsify or densify the vector space in order to produce higher similarities between more subtle topics. As we have a niche topic set to start with, we

found the common size of 300 [50] to perform very poorly; Figure 4-15 shows the posts in VSM as represented after a 2D projection; vector size of 25 have fewer smaller cluster where as for size of 300, there are many small clusters.



(a) Model RMSE



(b) Model MAP

Figure 4-16: Graphs showing RMSE and MAP for various sized vector sizes for Word Embedding model.

	RMSE	MAP	RMSE (Comments)	MAP (Comments)
10,MF	2.236068	0.293883	2.133073	0.425871
10,COS	1.967644	0.435263	1.910144	0.505352
15,MF	2.037155	0.62902	2.037155	0.672119
15,COS	1.80652	0.702203	1.812121	0.712413
20,MF	2.00624	0.656966	1.962142	0.709139
20,COS	1.766811	0.765281	1.732051	0.777308
25,MF	2	0.643666	1.942936	0.708222
25,COS	1.78961	0.735386	1.755301	0.761007
150,MF	2.04939	0.263069	1.936492	0.420348
150,COS	1.95213	0.413588	1.899502	0.488845
200,MF	2.061553	0.285648	1.955761	0.430469
200,COS	1.95213	0.390864	1.899502	0.471455
300,MF	2.085665	0	1.949359	0.263529
300,COS	1.812121	0.150383	1.737892	0.292794

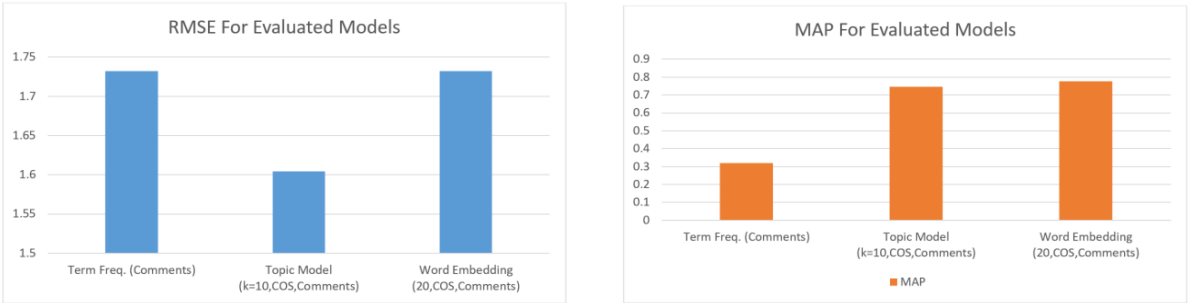
Table 4.6: Word Embedding Based Prediction Results for Vector Size of 300.

Table 4.6 lists the prediction error results for our word-embeddings model based recommendation against the ground truth, the same results are visualized in Figure 4-

16. The pipeline runtime averaged 3.5 hours for the entire corpus for each hidden layer parameter value. The cosine similarity based predictions provided the leading results for this model as well. We once again see that incorporation of the *comment* feature leading to improved MAP and lower RMSE. In our tests, vector size of 20 with the cosine similarity variant provided the best results for word-embeddings experiments. Higher vector sizes lead to higher error rates likely due to the multiple similar clusters that are formed at higher vector sizes. Indeed the 300-sized vector MF test failed altogether in a similar fashion to the TF-IDF model. No recommendation was rated 2 or higher, a requirement needed to assert a true-positive for MAP metrics.

4.5 Discussion

The MOTOR dataset has a typical *tail-long* characteristic [89], meaning activity is concentrated within its top 56 users. This is problematic for recommendation to majority



(a) Leading Variant RMSE

(b) Leading Varaint MAP

Figure 4-17: Graphs showing RMSE and MAP for the leading variants from all evaluated models.

of the users of the site as their activities are relatively limited. The dataset is also textual, with comments as implicit feedback feature. Figure 4-17 compares the results from all three model evaluations, with the topic model out performing the other two because of not only its high MAP but also its low RMSE values. In our analysis, we found the following:

- Comment feature proved a valuable integration for boosting recommendation ranking and error.
- Hyper-parameter adjustments improved overall results from each model.
- Corpus contained many variants of words; those which were misspelled or syntactically flawed (two words merged or hyphenated), leading to imperfect larger dictionaries which needed to be pre-processed.
- Topic modelling performed optimally for 10-dimensional topic descriptors. While larger topic sizes did not perform as well, they did not progressively become worse either. Deeper inspection of the topic strengths suggests LDA is maintaining the core topics intact while smaller clusters of similar topics do get created, their relative strengths to each posts remains low.
- Word Embeddings results showed a clear increase in performance as we tuned the vector-size hyper-parameter up to 20 dimensions, after which performance degrades. Normal sizes of 300 performed poorly on a dataset this small and sparse.
- Term Frequency based models performed poorly overall. This is likely due to the large dictionary size creating sparse vectors

- Cascading recommender systems did not yield any improvements in the quality and accuracy of recommendations, as such cosine similarity models proved more robust

Chapter 5

Conclusion

There exists plenty of literature on dealing with specific aspects of sparsity using commonly used and relatively richer data (such as Twitter-based or Netflix) [80], however to our best knowledge work on short-text and heavy kurtosis datasets is limited (some here: [69, 38, 88]). This is likely due to the sparse nature of the text and dataset causing model over-fitting and not being suitable for general applications. Our research question was to ask whether a new social network platform can use just the user posts to make viable recommendations and what approach would show promising results. In this work, we used a sparse, heavy-head and long-tailed data source with small gold-standard testing set to gauge predictions.

We evaluated the MOTOR dataset for recommendation feasibility using robust text based models. The dataset comes from a motoring enthusiast social network plat-

form, which has a sizeable user and posted content numbers, yet as we saw, a sizeable bulk of the posted content contribution is made by bot-accounts. Human user participation was centred around small group of frequent users, both for posts and comments. Lacking other consistent and reliable features in the dataset, the recommendations for users must come from user's explicit engagement; text content from users' posts and comments in particular. Majority of the posts made by majority of the users contain few words. In such circumstances we chose to use models that rely on terms and frequency, as well as latent factors. We used Term Frequency based models, Topic Models and Word Embeddings to represent the post in vector space. We also hoped that cascading the recommender systems would alleviate the over specialization issue present in content based modelling (known as the serendipity problem). In order to evaluate whether serendipity was addressed, the users need to be engaged in an online evaluation process, ideally. However, in lieu of an online evaluation for serendipity, we manually examined the top-k recommendations for each test-subject to determine if the list contained posts that were subjectively similar to posts the test-subject had explicitly rated as highly interesting. Our findings were inconclusive as posts are similar in broad topics (indeed the niche of the dataset is limited to reviewing and riding motorcycles). Overall, of the models we evaluated, the topic modelling technique showed the leading results. In other sets of unreported experiments we applied Natural Language Processing techniques to boost the performance, however the results were regrettably poor. Among the methods we tested were comment semantics (as weights in explicit feedback), named entity recognition (as features), content boosting (via Wikipedia), temporal values (time

of engagement as features), and aggregation of user's posted material to boost *wordiness* as well as the same for queries when making recommendation.

For future work, it would be prudent to improve ranking to obtain better recommendations. One way to do this would be to increase the term purity; we've identified vocabulary issues within the dataset, such as misspellings and term variants that could be stemmed even more rigorously. Another means to increasing similarity scores might be to use computer vision techniques to estimate entities in pictures shared by the users. As users of this platform share photos of their rides and events, understanding the subjects of the photos and incorporating those measures into the recommendation process might yield improved predictions. Greater access to online evaluations for tasks such as serendipity resolutions, or assessing the effects of regularization on recommendation, might make for improved assessments, as well.

Bibliography

[1] Mohammad Yahya H Al-Shamri. User profiling approaches for demographic recommender systems. *Knowledge-Based Systems*, 100:175–187, 2016.

[2] James "Mick" Andzulis, Nikolaos G Panagopoulos, and Adam Rapp. A review of social media and implications for the sales process. *Journal of Personal Selling & Sales Management*, 32(3):305–316, 2012.

[3] Preeti Arora, Shipra Varshney, et al. Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, 78:507–512, 2016.

[4] Ricardo Baeza-Yates, Berthier de Araújo Neto Ribeiro, et al. *Modern information retrieval*. New York: ACM Press; Harlow, England: Addison-Wesley,, 2011.

[5] Rodrigo C Barros, André CPLF De Carvalho, Alex A Freitas, et al. *Automatic design of decision-tree induction algorithms*. Springer, 2015.

[6] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SiGKDD Explorations*, 9(2):75–79, 2007.

[7] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[8] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[10] Branimir K Boguraev, Jennifer Chu-Carroll, David A Ferrucci, Anthony T Levas, and John M Prager. Context-based disambiguation of acronyms and abbreviations, April 28 2015. US Patent 9,020,805.

[11] Shelley Boulianne. Social media use and participation: A meta-analysis of current research. *Information, communication & society*, 18(5):524–538, 2015.

[12] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466. Springer, 2013.

[13] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[14] Wray L Buntine. Operations for learning with graphical models. *Journal of artificial intelligence research*, 2:159–225, 1994.

[15] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[16] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.

[17] George H Chen, Devavrat Shah, et al. Explaining the success of nearest neighbor methods in prediction. *Foundations and Trends® in Machine Learning*, 10(5-6):337–588, 2018.

[18] Hanhua Chen, Hai Jin, and Xiaolong Cui. Hybrid followee recommendation in microblogging systems. *Science China Information Sciences*, 60(1):012102, 2017.

[19] Yuyun Chen, Hang Dong, and Wei Wang. Topic-graph based recommendation on social tagging systems: a study on research gate. In *Proceedings of the 2018 International Conference on Data Science and Information Technology*, pages 138–143. ACM, 2018.

[20] George Christodoulides, Colin Jevons, and Jennifer Bonhomme. Memo to marketers: Quantitative evidence for change: How user-generated content really affects brands. *Journal of advertising research*, 52(1):53–64, 2012.

[21] III CoNcLUs1oNs. Note on optimal selection of independent binary-valued features for pattern recognition. 1971.

[22] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and information systems*, 1(1):5–32, 1999.

[23] Marco De Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*, pages 119–159. Springer, 2015.

[24] Paul DiMaggio, Manish Nag, and David Blei. Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of us government arts funding. *Poetics*, 41(6):570–606, 2013.

[25] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.

[26] Michael Ekstrand. Towards recommender engineering: tools and experiments for identifying recommender differences. 2014.

[27] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20:29–50, 2016.

[28] Ahmed Elsafty, Martin Riedl, and Chris Biemann. Document-based recommender system for job postings using dense representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, volume 3, pages 216–224, 2018.

[29] Bellarmine A Ezumah. College students' use of social media: Site preferences, uses and gratifications theory revisited. *International Journal of Business and Social Science*, 4(5), 2013.

[30] Jerome H Friedman. Data mining and statistics: What's the connection? *Computing Science and Statistics*, 29(1):3–9, 1998.

[31] Yeyun Gong, Qi Zhang, Xuyang Sun, and Xuanjing Huang. Who will you@? In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 533–542. ACM, 2015.

[32] Derek Greene, Derek O'Callaghan, and Pádraig Cunningham. How many topics? stability analysis for topic models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 498–513. Springer, 2014.

[33] Katherine Haenschen. Social pressure on social media: Using facebook status updates to increase voter turnout. *Journal of Communication*, 66(4):542–563, 2016.

[34] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[35] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[36] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, volume 8, pages 263–272. Citeseer, 2008.

[37] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.

[38] Ziwon Hyung, Kibeom Lee, and Kyogu Lee. Music recommendation using text analysis on song requests to radio stations. *Expert Systems with Applications*, 41(5):2608–2618, 2014.

[39] Norbert Jankowski and Marek Grochowski. Comparison of instances seletion algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, pages 598–603. Springer, 2004.

[40] Shuhui Jiang, Xueming Qian, Jialie Shen, Yun Fu, and Tao Mei. Author topic model-based collaborative filtering for personalized poi recommendations. *IEEE transactions on multimedia*, 17(6):907–918, 2015.

[41] Steve Jones and Guillaume Latzko-Toth. Out from the plato cave: uncovering the pre-internet history of social computing. *Internet Histories*, 1(1-2):60–69, 2017.

[42] Jan H Kietzmann, Kristopher Hermkens, Ian P McCarthy, and Bruno S Silvestre. Social media? get serious! understanding the functional building blocks of social media. *Business horizons*, 54(3):241–251, 2011.

[43] N Kishore Kumar and Jan Schneider. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, 65(11):2212–2244, 2017.

[44] Daniel Kluver, Michael D Ekstrand, and Joseph A Konstan. Rating-based collaborative filtering: algorithms and evaluation. In *Social Information Access*, pages 344–390. Springer, 2018.

[45] Edwin M Knox and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases*, pages 392–403. Citeseer, 1998.

[46] Mei Kobayashi, Georges Dupret, Oliver King, and Hikaru Samukawa. Estimation of singular values of very large matrices using random sampling. *Computers & Mathematics with Applications*, 42(10-11):1331–1352, 2001.

[47] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *KDD*, pages 114–119, 1996.

[48] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[49] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[50] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[51] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[52] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[53] Qing Li and Byeong Man Kim. An approach for combining content-based and collaborative filters. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pages 17–24. Association for Computational Linguistics, 2003.

[54] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1):76–80, 2003.

[55] Gregory D Linden, Jennifer A Jacobi, and Eric A Benson. Collaborative recommendations using item-to-item similarity mappings, July 24 2001. US Patent 6,266,649.

[56] Trieu Thi Ly Ly, Nguyen Van Quy, Ninh Khanh Duy, Huynh Huu Hung, and Dang Duy Thang. Representing context in abbreviation expansion using machine learning approach. *PROCEEDING of Publishing House for Science and Technology*, 2019.

[57] Huifang Ma, Meihuizi Jia, Di Zhang, and Xianghong Lin. Combining tag correlation and user social relation for microblog recommendation. *Information Sciences*, 385:325–337, 2017.

[58] Prem Melville, Raymod J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192. American Association for Artificial Intelligence, 2002.

[59] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[60] James William Pontes Miranda, Danilo Henrique Rodrigues Bueno, Rogéria Cristiane Gratão de Souza, Carlos Roberto Valêncio, Antônio Marcos Neves Esteca, and Geraldo Francisco Donegá Zafalon. A qualitative approach to develop niche social networks. In *Proceedings of the 18th International Conference on Enterprise Information Systems*, pages 265–272. SCITEPRESS-Science and Technology Publications, Lda, 2016.

[61] Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *RecSys Posters*, 2015.

[62] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Searching microblogs: coping with sparsity and document quality. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 183–188. ACM, 2011.

[63] Aaron David Nielsen, Qiao Pang, Ebrahim Bagheri, Mohammad-Amin Jashki, and Fattane Zarrinkalam. System and method of analyzing social media to predict the churn propensity of an individual or community of customers, September 1 2016. US Patent App. 15/052,831.

[64] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 37–76. Springer, 2015.

[65] Makbule Gulcin Ozsoy. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356*, 2016.

[66] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[67] Marco Pennacchiotti and Siva Gurumurthy. Investigating topic models for social media user recommendation. In *Proceedings of the 20th international conference companion on World wide web*, pages 101–102. ACM, 2011.

[68] Selwyn Piramuthu. Evaluating feature selection methods for learning in data mining applications. *European journal of operational research*, 156(2):483–494, 2004.

[69] Alexandrin Popescul, David M Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 437–444. Morgan Kaufmann Publishers Inc., 2001.

[70] Pavel Pudil, Francesc J Ferri, Jana Novovicova, and Josef Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 279–283. IEEE, 1994.

[71] Brajendra Singh Rajput and A NilayKhare. A survey of stemming algorithms for information retrieval. *IOSR Journal of Computer Engineering*, 17(3):76–80, 2015.

[72] Usha Ramanathan, Nachiappan Subramanian, and Guy Parrott. Role of social media in retail network operations and marketing to enhance customer satisfaction. *International Journal of Operations & Production Management*, 37(1):105–123, 2017.

[73] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[74] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[75] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[76] Karin Sajithra and Rajindra Patil. Social media–history and components. *Journal of Business and Management*, 7(1):69–74, 2013.

[77] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[78] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[79] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[80] Kent Shi and Kamal Ali. Getjar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–212. ACM, 2012.

[81] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, and Prasenjit Choudhury. A comparative study of different similarity metrics in highly sparse rating dataset. In *Data Management, Analytics and Innovation*, pages 45–60. Springer, 2019.

[82] Anjana Susarla, Jeong-Ha Oh, and Yong Tan. Social networks and the diffusion of user-generated content: Evidence from youtube. *Information Systems Research*, 23(1):23–41, 2012.

[83] Mir Saman Tajbakhsh and Jamshid Bagherzadeh. Microblogging hash tag recommendation system based on semantic tf-idf: Twitter use case. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 252–257. IEEE, 2016.

[84] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of machine learning research*, 10(Mar):623–656, 2009.

[85] Daniel Trottier. *Social media as surveillance: Rethinking visibility in a converging world*. Routledge, 2016.

[86] Maurice Vergeer. Twitter and political campaigning. *Sociology compass*, 9(9):745–760, 2015.

[87] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[88] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1069–1078. ACM, 2014.

[89] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.

[90] Markus Zanker, Markus Jessenitschnig, Dietmar Jannach, and Sergiu Gordea. Comparing recommendation strategies in a commercial context. *IEEE Intelligent Systems*, 22(3):69–73, 2007.

[91] Peng Zhang, Tun Lu, Hansu Gu, Xianghua Ding, and Ning Gu. How do you interact with your old friends on a new site: Understanding social ties among different social network sites. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pages 2–9. ACM, 2017.

[92] Jing Zhou, Dean P Foster, Robert A Stine, and Lyle H Ungar. Streamwise feature selection. *Journal of Machine Learning Research*, 7(Sep):1861–1885, 2006.

[93] Zhiliang Zhu, Jie Liang, Deyang Li, Hai Yu, and Guoqi Liu. Hot topic detection based on a refined tf-idf algorithm. *IEEE Access*, 2019.