Towards Effective Approaches for News Recommendation System

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Computational Linguistics by Research

by

Vaibhav Kumar
201325115
vaibhav.kumar@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2018
It is certified that the work contained in this thesis, titled “Towards Effective Approaches for News Recommendation System” by Vaibhav Kumar, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Vasudeva Varma
To Embark on a Journey of Research
Acknowledgments

I would like to take this opportunity to express my heartfelt gratitude towards everyone who not only helped me in completing this thesis but also made my experience at IIIT Hyderabad very memorable. As I submit this thesis, I feel indebted to a lot of people.

I would like to thank Prof. Vasudeva Varma, without whom this work would not have been possible. He has been an excellent mentor since I joined the laboratory. He has provided me with valuable insights throughout my time here and has encouraged me to think more about the problem itself rather than shoot a blind arrow towards a possible solution. I would also like to thank Dr. Manish Gupta who has helped me get a deeper understanding of the problem at hand. His ideas on various experiments have been excellent. His thorough review has helped me prepare papers in a crisp and concise manner. They have provided me tremendous support to help me embark on a journey of research.

I would like to thank my mother and father, uncle and aunt and my grandmother who have always supported me. They have given me continuous encouragement throughout my years of study and to pursue it even further. A special thanks to little brother Aadi.

I would like to extend my deepest thanks to my lab mate Dhurv Khattar, who has helped me lot in bringing out various insights and methods to solve a particular problems and conducting a wide variety of experiments. A special thanks to another lab mate Shashank Gupta for providing intriguing ideas.

I could not have accomplished it without the constant support from many of my friends. Ishaan, Kartik and Mrinal, thanks for being around and going beyond your limits to help me. The last year wouldn’t have been so wonderful without you guys. Amogh, Bhavesh, Chirag, thanks to you all for bringing out the best in me, as we would call it ”unconditional fearlessness”. Sambuddha and Aayush, thanks for doing nothing. Yash Lal, thanks for all the help and discussion and for being there perennially. Abhimanshu Nepali, I would like you to thank you for the fool that you have been, educating a lot of us about the real meaning of chilling. Eshita, thanks for always motivating me and being my constant source of support, and of course how can I forget the 24x7 masala that we have shared alongwith this string 60cb939f0cb565d87a116d442a259cf3 that only you will understand. You people have been truly amazing and have made me realize the importance of friendship.
Abstract

Deep neural networks have yielded immense success in speech recognition, computer vision and natural language processing. However, the exploration of deep neural networks for news recommendation systems has received a relatively less amount of attention. Also, different recommendation scenarios have their own issues which creates the need for different approaches for recommendation.

With news stories coming from a variety of sources, it is crucial for news aggregators to present interesting articles to the user to maximize their engagement. This creates the need to have a recommendation system which accounts for both the content of the articles and the user preferences. Methods such as Collaborative Filtering, which are well known for general recommendations, are not suitable for news because of the short life span of articles and because of the large amount of articles published each day. Typically, it is desirable that a news recommendation system be able to discriminate between and select articles from a pool to recommend to a user as soon as they are published. Apart from this, such methods do not harness the information present in the sequence in which the articles are read by the user and hence are unable to account for the various interests of the user which may keep changing with time. Alternatively, the other class of models based on Content Filtering, can handle cold start problems but in the long run tend to suffer from the problem of over specialization.

In this thesis, we address these issues in a step-by-step manner. We start off with a problem that is commonly associated with deep learning based methods i.e lack of sufficient data. In order to tackle this issue we come up with an item-based collaborative filtering approach which utilizes Markov Decision Process (MDP). We also come up with a novel semantic similarity measure which we incorporate as a reward for the MDP. This helps us gain insights about the various interests that users may have by only having prior knowledge of a few articles read by them in the past.

We then move on to the exploration of various deep learning based methods to tackle our problem. We give importance to utilizing the content of the articles, and taking into account the historical reading data of a user. We attempt to solve the cold-start problem, and make effective recommendations for users who have had little to no interaction with items. We design the model to keep learning parameters based on implicit feedback from the users. A description of how we go about it is described as follows.

We first come up with a user profiling based approach and use it in combination with a Deep Semantic Structured Model (DSSM). We then later expand the model and utilize a recurrent neural network with an attention mechanism. Such a mechanism helps us discriminate between the various interests of the user. We then come up with the Recurrent Attentive Recommendation Engine (RARE). RARE consists
of two components and utilizes the distributed representations of news articles. The first component is used to model the user’s sequential behaviour of news reading in order to understand her general interests i.e to get a summary of her interests. The second component utilizes an article level attention mechanism to understand her particular preferences. We feed the information obtained from both the components to a Siamese Network in order to make predictions which pertain to the user’s generic as well as specific interests.

We carry out extensive experiments to establish the effectiveness of our methods. We also perform experiments to prove the efficacy of our model on solving the item cold-start cases as well as making effective recommendations for users who have had very little interaction with articles.

Finally, we experiment with a novel 3D Convolutional Neural Network based model in an attempt to solve similar problems as the ones we tried to address earlier. Applying 3D convolution helped us identify both spatial (features of a particular article) as well as temporal information (features present in the sequence of articles read by the user) which are pertinent to a user’s interest. The initial results of this experimentation are also presented in this thesis.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
</tr>
<tr>
<td>1.2</td>
<td>News Aggregators, Recommendation Systems and Challenges</td>
</tr>
<tr>
<td>1.2.1</td>
<td>News Aggregators</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Functions of a News Recommendation System</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Challenges in News Recommendation</td>
</tr>
<tr>
<td>1.3</td>
<td>Summary of Thesis</td>
</tr>
<tr>
<td>1.3.1</td>
<td>The Problem</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Contributions</td>
</tr>
<tr>
<td>1.3.3</td>
<td>Thesis Workflow</td>
</tr>
<tr>
<td>2</td>
<td>Related Work</td>
</tr>
<tr>
<td>2.1</td>
<td>Common Approaches for Recommendation Systems</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Collaborative Filtering (CF)</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Content-based Filtering</td>
</tr>
<tr>
<td>2.2</td>
<td>Neural Network based Recommendation</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Restricted Boltzmann Machine for CF</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Autoencoder based Recommendation System</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Deep Semantic Structured Model (DSSM)</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Neural Matrix Factorization</td>
</tr>
<tr>
<td>3</td>
<td>Leveraging Moderate User Data for News Recommendation</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview</td>
</tr>
<tr>
<td>3.2</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Statistical Language Modeling</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Markov Decision Process (MDP)</td>
</tr>
<tr>
<td>3.3</td>
<td>Proposed Approach</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Bag of News Model</td>
</tr>
<tr>
<td>3.3.2</td>
<td>MDP based Recommendation</td>
</tr>
<tr>
<td>3.3.2.1</td>
<td>Semantic Similarity</td>
</tr>
<tr>
<td>3.3.2.2</td>
<td>MDP Description for News</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Recommending Articles</td>
</tr>
<tr>
<td>3.4</td>
<td>Data and Experiments</td>
</tr>
<tr>
<td>3.5</td>
<td>Results and Discussion</td>
</tr>
<tr>
<td>3.6</td>
<td>Summary</td>
</tr>
</tbody>
</table>
## CONTENTS

4 User Profiling based Neural Network for Temporal News Recommendation .......................... 21
  4.1 Overview .................................................................................................................. 21
  4.2 Model Architecture .................................................................................................. 22
    4.2.1 User Profiling ................................................................................................. 23
    4.2.2 Prerequisite .................................................................................................... 24
    4.2.3 Modified DSSM ......................................................................................... 24
    4.2.4 Learning ....................................................................................................... 25
  4.3 Experiments ............................................................................................................. 25
    4.3.1 Dataset ......................................................................................................... 26
    4.3.2 Experimental Settings .................................................................................. 26
      4.3.2.1 Evaluation Protocol ............................................................................. 26
      4.3.2.2 Baselines ............................................................................................. 27
      4.3.2.3 Parameter Settings ............................................................................. 27
  4.4 Results and Discussion ............................................................................................ 29
  4.5 Summary .................................................................................................................. 31

5 Deep Neural Architecture for News Recommendation ....................................................... 32
  5.1 Overview .................................................................................................................. 32
  5.2 Model Architecture .................................................................................................. 33
    5.2.1 Prerequisite .................................................................................................... 33
    5.2.2 Recurrent Attention DSSM (RA-DSSM) ......................................................... 33
    5.2.3 Learning ....................................................................................................... 36
    5.2.4 Relation with Matrix Factorization ................................................................. 36
  5.3 Experiments ............................................................................................................. 37
    5.3.1 Dataset ......................................................................................................... 37
    5.3.2 Experimental Settings .................................................................................. 38
      5.3.2.1 Evaluation Protocol ............................................................................. 38
      5.3.2.2 Baselines ............................................................................................. 38
      5.3.2.3 Parameter Settings ............................................................................. 39
  5.4 Results and Discussion ............................................................................................ 40
  5.5 Summary .................................................................................................................. 48

6 RARE : A Recurrent Attentive Recommendation Engine for News ..................................... 50
  6.1 Overview .................................................................................................................. 50
  6.2 Model Architecture .................................................................................................. 52
    6.2.1 Task Description ............................................................................................. 52
    6.2.2 Overview ....................................................................................................... 53
    6.2.3 Distributed Representation for News Articles ................................................. 53
    6.2.4 Generic Encoder ............................................................................................ 53
    6.2.5 Specific Encoder ............................................................................................ 54
    6.2.6 RARE ............................................................................................................ 54
    6.2.7 Learning ....................................................................................................... 56
  6.3 Experiments ............................................................................................................. 56
    6.3.1 Dataset ......................................................................................................... 57
    6.3.2 Baselines ....................................................................................................... 57
    6.3.3 Evaluation Protocol ..................................................................................... 57
CONTENTS

6.3.4 Parameter Learning .................................................. 58
6.4 Results and Discussion .................................................. 65
6.5 Summary ................................................................. 67

7 Word Semantics based 3-D Convolutional Neural Networks for News Recommendation ........ 68
7.1 Overview ................................................................. 68
7.2 Model Architecture ....................................................... 68
  7.2.1 3-D Tensor ............................................................. 68
  7.2.2 3-D Convolution ..................................................... 69
  7.2.3 Score Aggregation .................................................... 70
7.3 Experiments ............................................................... 70
7.4 Performance Comparison .............................................. 72
7.5 Summary ................................................................. 72

8 Conclusions and Future Work ............................................ 73
8.1 Conclusion ............................................................... 73
8.2 Future Work .............................................................. 76

Bibliography ................................................................. 79
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>News Aggregator Overview</td>
<td>3</td>
</tr>
<tr>
<td>4.1</td>
<td>Model Architecture for the Modified DSSM. ⊗ represents the operation using which the user profile is created. The left side of the model creates the user profile, while the right side of the model provides positive and negative instances for training the model parameters. $r_1 \ldots r_R$ represent articles present in the user history. Item $^+$ and item $^-$ represents the positive and negative instances respectively.</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>HR@K performance of our model vs state-of-the-art models</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>NDCG@K performance of our model vs state-of-the-art models</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>HR@K performance of our model on cold start cases</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>NDCG@K performance of our model on cold start cases</td>
<td>30</td>
</tr>
<tr>
<td>5.1</td>
<td>Model Architecture of Recurrent Attention DSSM</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>HR@10 of RA-DSSM in w.r.t User’s Reading History</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>NDCG@10 of RA-DSSM in w.r.t User’s Reading History</td>
<td>41</td>
</tr>
<tr>
<td>5.4</td>
<td>HR@K of RA-DSSM vs state-of-the-art</td>
<td>42</td>
</tr>
<tr>
<td>5.5</td>
<td>NDCG@K of RA-DSSM vs state-of-the-art</td>
<td>42</td>
</tr>
<tr>
<td>5.6</td>
<td>HR@K of RA-DSSM on Cold-Start cases</td>
<td>43</td>
</tr>
<tr>
<td>5.7</td>
<td>NDCG@K of RA-DSSM on Cold-Start cases</td>
<td>43</td>
</tr>
<tr>
<td>5.8</td>
<td>HR@K of RA-DSSM for different recurrent units</td>
<td>44</td>
</tr>
<tr>
<td>5.9</td>
<td>NDCG@K of RA-DSSM for different recurrent units</td>
<td>44</td>
</tr>
<tr>
<td>5.10</td>
<td>HR@10 performance of RA-DSSM w.r.t the number of factors</td>
<td>45</td>
</tr>
<tr>
<td>5.11</td>
<td>NDCG@10 performance of RA-DSSM w.r.t the number of factors</td>
<td>45</td>
</tr>
<tr>
<td>5.12</td>
<td>HR@10 performance of RA-DSSM w.r.t the number of negative samples</td>
<td>46</td>
</tr>
<tr>
<td>5.13</td>
<td>HR performance of RA-DSSM vs state-of-the-art models on Yahoo WebScope R6B</td>
<td>46</td>
</tr>
<tr>
<td>5.14</td>
<td>NDCG performance of RA-DSSM vs state-of-the-art models on Yahoo WebScope R6B</td>
<td>47</td>
</tr>
<tr>
<td>6.1</td>
<td>In (A), the users sequence is used to model her general interests. While in (B), the users specific interests are captured. In (C), the changing interests of the users are modelled. In all these cases, sequential reading history of a user plays an important role. Different colors represent the different topics of the article.</td>
<td>51</td>
</tr>
<tr>
<td>6.2</td>
<td>Model Overview</td>
<td>52</td>
</tr>
<tr>
<td>6.3</td>
<td>The two components of RARE</td>
<td>54</td>
</tr>
<tr>
<td>6.4</td>
<td>Complete Architecture</td>
<td>55</td>
</tr>
<tr>
<td>6.5</td>
<td>HR of RARE vs state-of-the-art on CLEF News-REEL</td>
<td>58</td>
</tr>
</tbody>
</table>
6.6 NDCG of RARE vs state-of-the-art on CLEF News-REEL ........................................ 59
6.7 HR of RARE vs state-of-the-art on Malayalam Dataset ........................................... 59
6.8 HR of RARE vs state-of-the-art on Malayalam Dataset ........................................... 60
6.9 HR of RARE vs state-of-the-art on Bahasa Dataset ................................................ 60
6.10 HR of RARE vs state-of-the-art on Bahasa Dataset ............................................... 61
6.11 HR of RARE w.r.t. the Users Reading History on NewsREEL. A similar trend was
observed for the other datasets. .................................................................................. 61
6.12 NDCG of RARE w.r.t. the Users Reading History on NewsREEL. A similar trend was
observed for the other datasets. .................................................................................. 62
6.13 HR of our RARE w.r.t recurrent unit used in RARE on NewsREEL. A similar trend was
observed for the other datasets. .................................................................................. 62
6.14 NDCG of our RARE w.r.t recurrent unit used in RARE on NewsREEL. A similar trend
was observed for the other datasets. ........................................................................... 63
6.15 HR of RARE on cold-start cases ............................................................................. 63
6.16 NDCG of RARE on cold-start cases ....................................................................... 64
6.17 HR of RARE on negative sampling ...................................................................... 64
6.18 NDCG of RARE on negative sampling .................................................................. 65

7.1 Model Architecture for 3D CNN for Recommendation ............................................. 69
7.2 HR of 3D CNN vs some baselines ........................................................................... 70
7.3 NDCG of 3D CNN vs some baselines ...................................................................... 71
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Comparative Performance of MDP vs others</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Performance of our model using different user profiling operations</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>Performance of some variations of our model</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Some examples for which our model recommended the correct article on the first hit (HR@1).</td>
<td>49</td>
</tr>
<tr>
<td>6.1</td>
<td>Performance of RARE by changing number of dense layers</td>
<td>58</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance using different Encoding Mechanism on CLEF NewsREEL</td>
<td>59</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Today’s modern world is a period of information overload. The Internet (web) provides instant access to a wide variety of content. Hence, it becomes desirable to have a system which can point a user to the most relevant content with minimal hassle. This is where a recommendation system comes into play. A formal definition would be:

"Recommendation systems are software tools and techniques providing suggestions of items to be of use to a user." [32]."

We use the web for a variety of purposes. For example: web search, online shopping, news reading etc. In case of web search i.e while using Google, Bing etc., we observe that as soon as we type our query we have a list of suggestions provided by the search engine. Many a times, the suggested list of queries actually contains what we were going to type. Taking a different example, when we shop online, we observe a list of items similar to what we have bought before which draws our attention and we often end up buying those. All of this is because of a recommendation system playing its part in the background.

Gone are the days when people blindly followed news from one source. Now, they want to read articles and stories from a variety of sources to understand all its nuances. News aggregators come into play here. They collect news articles from a variety of sources and presents it to the user in one single location. With news coming from various streams, it becomes very cumbersome for a user to select articles of her choice from a list of presented articles which pertains to a variety of subjects. Hence, it becomes crucial for such aggregators to have a recommendation system which points the user to the most relevant items and thus, maximize her engagement with the site and, in turn, minimize the time needed to find relevant content.

A popular approach to the task of recommendation is collaborative filtering [3][31][35], which uses the user’s past interaction with the item to predict the most relevant content. Another common approach is content-based recommendation, which uses common features between items and/or users to recommend new items to the users based on the similarity between features. Among the various approaches
for collaborative filtering, Matrix Factorization (MF) [20] is the most popular one. It projects users and items into a shared space, using a vector of latent features to represent the user or the item. Thereafter, a user’s interaction with an item is modelled as the inner product of their latent vectors.

Different scenarios have varying recommendation requirements which creates the need for specific recommendation systems. Methods such as Collaborative Filtering which are well known for general recommendations (movies, books etc.) are not suitable for news. Such methods suffer from three major problems:

1. It requires a considerable amount of user historical data before making good recommendations.

2. News articles are transient in nature. What might be of interest today might not be of interest to the user a few days down the line. Hence, the system should be able to recommend articles as soon as they are published. This is the typical item cold start problem.

3. Such methods do not harness information encoded in the sequential reading history of the user and hence fail to account for the generic and the specific interests of the user which may keep changing with time.

Alternatively, content-based methods can handle the problem of item cold-start but suffer from the problem of over-specialization. Hence, we need a system which is specifically tailored to handle the problems associated with news recommendation.

We posit that understanding the content of the article and the user’s sequential reading history are very important for making good predictions. In this thesis, we begin with a system which utilizes moderate user data for providing recommendations. We then come up with a series of approaches which (1) utilizes the content of the article and, (2) harnesses the information present in the sequential reading data of a user. Apart from it, these approaches were designed keeping in mind that the system should (1) be capable of recommending articles as soon as they are published and, (2) be able to make effective recommendations for users who have had very limited interactions with the website.

1.2 News Aggregators, Recommendation Systems and Challenges

In this section, we provide a comprehensive description of news aggregators and enumerate some of the functions of a desirable news recommendation system. We then detail a list of challenges associated with news recommendation.

1.2.1 News Aggregators

An overview of a news aggregator can be seen in Figure 1.1. It crawls through a list of sources in order to identify potential relevant news articles. It then aggregates the entire content into one location for easy viewing. Such a method gives the user the flexibility to read news pertaining to variety of
topics coming from different sources. A single digest may be unable to cover all the different kinds of news that a user might be looking for, but reading news sourced through an aggregator helps solve this problem. Apart from this, reading news from a variety of sources also provides multiple perspectives of a particular event, and encourages better understanding of its nuances. Altogether, there are a variety of reasons why a user would prefer a news aggregator over a single news digest.

1.2.2 Functions of a News Recommendation System

In the previous section we provided a definition of a recommendation system. Here we aim to provide a list of important functions of a recommendation system [32].

1. Increase the number of news articles read: A news aggregator collects a lot of news. This necessitates that most of the articles should be read. It should target its audience in a way that the number of news articles read are increased. Increasing news viewership is perhaps the most important function for a news recommendation system.

2. Make diverse predictions to the user: It should be able to recommend articles to a user that might be hard to find for her, without having to look for it specifically. A news aggregator presents a list of articles to the user, browsing over which would be very cumbersome. A recommendation system should be able to accurately pin point that article.

3. Increase user satisfaction: The user should find the recommended articles interesting and relevant. The user should be able to enjoy the effects of the system. A combination of effective and accurate recommendation will help increase the subjective evaluation of a news website by the user.
4. Increase user fidelity: If a news website is visited by a user again and again over time, the recommendation should be able to make better and better predictions for her over time. Consequently, the longer the user interacts with the website, the more refined her user model should be.

5. Better understanding of the needs of the user: The preferences of the user should be effectively identified. Many times a user might not want to provide information about her in an explicit manner. In that case, the system should be able to understand her preferences in an implicit manner.

In a nutshell, we would want the recommendation system to be able to (1) find interesting articles for a user, (2) recommend a ranked list of those articles, and (3) adapt to the user’s reading behaviour.

1.2.3 Challenges in News Recommendation

Different recommendation scenarios come with different challenges, none same as the other. Here we identify the set of challenges associated with that of news recommendation.

1. Cold-Start Problem: This particular problem is common for all types of recommendation system. Typically, if a recently published news articles has not been read or interacted with by any user, then it would not be recommended to any other user. Or if there is no or very limited information about a user, then it is not possible to make recommendations for her.

2. Data Sparsity: Typical methods such as collaborative filtering utilize matrices that can be very sparse when there are not enough interactions between users and items. Data sparsity causes a decrease in the performance of the system.

3. Recency: It is perhaps one of the most important challenges in news recommendation domain. Most of the users would want to read fresh news articles instead of old dated ones. The life span of an article is very short and its importance keeps decreasing over time.

4. Implicit Feedback: Users would not want to go through the tedious process of filling up forms and providing information about her interests. Hence, it becomes very crucial to utilize implicit feedback in order to understand her interests.

5. Changing user preferences: The interests of the user might keep changing over time. A topic that might be interesting to a particular user today, might not be of interest to her a few weeks later. Some topics might not be of any interest to the user but she would still read it because she finds it important.

6. Over-specialization: A news recommendation system should not be recommending the same items repeatedly. Also, there should be some amount of diversity in the recommendations. It might be possible, that the recommendation system develops a bias toward a particular topic for a particular user and only recommends articles belonging to that topic. Such a situation is also not desirable.
1.3 Summary of Thesis

In this section, we provide a brief overview of the thesis. We first mention our problem statement and give a brief description of the various kinds of issues we attempted to solve. Additionally, we also present a brief summary of our approach. We then list the contributions made by us followed by a brief workflow of the thesis.

1.3.1 The Problem

In this thesis, we propose a set of approaches to improve the effectiveness of recommendation system used by news aggregators. We tackle various problems associated with recommendation systems in the news domain and in a step by step manner come up with a robust system capable of making good predictions.

- **Improving Performance when Presented with Moderately Low User Data**

  Most effective recommendation systems require quite a number of interactions between the user and the articles before making good predictions. Other classes of deep learning models require quite a lot of data in order to learn the parameters of the model. This places the need to have a recommendation system which can work well even in the case of moderate amount of data. For news aggregators which are new to the market, such a system would come in very handy in its initial stages.

  To tackle this issue, we propose an item-based collaborative filtering method for recommending news items using Markov Decision Process (MDP). Due to the sequential nature of news reading, we choose MDP to model our recommendation system as it is based on a sequence optimization paradigm. Further, we also incorporate factors like article freshness and similarity into our system by extrinsically modelling it in terms of reward for the MDP.

- **Exploration of Profiling Strategies for capturing Temporal Changes in User Interests**

  Typical methods such as Collaborative Filtering and Content-based Filtering do not explicitly account for the temporal changes in users interests. It might be possible that a particular topic which the user finds interesting today might be of no interest to her tomorrow. Apart from this, the user might also keep developing new interests which needs to be accounted for.

  To tackle this issue, we propose a deep neural network based architecture which is based on a two level approach. We first generate document embeddings for every news article. We then use the embeddings of the previously read articles by a user to come up with her user profile. We experiment with three different profiling strategies which helps us to understand the temporal aspect of a user’s interest. We then use the generated profiles along with adequate positive and negative samples to train the parameters of the neural network. Using the content embeddings endows the model with the ability to tackle the cold start problem as well.
• Incorporating Attention Mechanism to Understand Users Interests

Most of the existing recommendation methods do not perform well in case of news due to the dynamic nature of reading the news. In the approach we mentioned previously, we tried to experiment with different user profiling strategies which would enable the system to gather a temporal sense of the users interests.

We take our previous approach a step further and utilize a recurrent neural network with an attention mechanism in order to have a better understanding of the users interests. We first generate the embeddings for each news article. We then utilize a deep neural model, where a non-linear mapping of users and items are learnt first. For learning a non-linear mapping for the users we use an attention-based recurrent layer in combination with fully connected layers. To learn the mappings for the items we use only the fully connected layers. We then use a ranking based objective function to learn the parameters of the network.

• Unification of Specific and Generic Interests of the User

Here, we aim to unify the insights obtained from our previous approaches in order to come up with a robust recommendation system. We come up with an approach which is able to identify and adapt to the temporal changes in users interests. It is based on identifying the specific as well as generic interests of the users. Apart from this, the approach also has the capability to handle the item and user cold start problem.

We come up with a model which consists of two components and utilizes the distributed representations of news articles. The first component is used to model the users sequential behaviour of news reading in order to understand her general interests i.e to get a summary of her interests. The second component utilizes an article level attention mechanism to understand her specific interests. The information obtained from both the components is then fed to a Siamese Network in order to make predictions which pertain to the users generic as well as specific interests.

• Experimenting with 3-D Convolutional Networks for Understanding User Interests

Most of our above approaches were based on Recurrent Neural Networks. Towards the end, we tried experimenting with 3-D Convolutional Networks to solve a similar problem. 3-D CNNs have proven to be effective in case of tasks such as action recognition where capturing features from both the temporal and the spatial dimensions is important. In our case, spatial dimensions would correspond to the features of an article i.e features which attracted the attention of a user in that particular article while the temporal dimensions would correspond to features across articles read by a user.

Here we propose a recommendation model which uses semantic similarity between words as input to a 3-D Convolutional Neural Network in order to make predictions. We outline our preliminary results and aim to take this work further.
1.3.2 Contributions

We make the following contribution in this thesis:

- **Leveraging Moderate User Data for News Recommendation**
  - We present an item-based collaborative filtering approach for news recommendation using Markov Decision Process.
  - We incorporate various factors like article freshness and similarity by explicitly modelling it as reward for the MDP.
  - On a moderately low amount of user data, our method outperforms the other set of approaches.

- **User Profiling based Neural Architecture for Temporal News Recommendation**
  - We use doc2vec embeddings of each news article in order to come up with user profiles for each user which encapsulates information about the changing interests of the user over time.
  - We use a deep neural component built such that both user-item interaction information and news article (item) content are used to model latent features of users and items.
  - We perform experiments to demonstrate the utility of our model for the problem of news recommendation. We then perform experiments to show the effectiveness of our model when the user has had very little interaction with items.

- **Deep Neural Architecture for News Recommendation**
  - We present a deep neural architecture for news recommendation in which we utilize the user-item interaction as well as the content of the news (items) to model the latent features of users and items.
  - In order to address the changing interests of the users and the granularity/extent of these interests over time, we incorporate LSTMs augmented with an attention mechanism to better model the latent features of the user.
  - We present results to demonstrate the efficacy and utility of our model for the problem of news recommendation. We then perform experiments to show the effectiveness of our model to solve the problems of user and item cold-start respectively.

- **RARE : A Recurrent Attentive Recommendation Engine for News Aggregators**
  - We present a neural network based architecture (RARE) with the following capabilities
    - It considers the content of the news articles giving it the ability to recommend articles as soon as they are published since it is not dependent on the article’s previous interactions with users.
It takes into account the users’ generic as well as specific interests
* It adapts to the changing interests of the user

– We carry out extensive experiments over three real world dataset to show the effectiveness of our model. The results reveal that our method outperforms the state-of-the-art.
– We also show the ability of our model for handling the cold-start cases as well.

• Word Semantics based 3-D Convolutional Neural Networks for News Recommendation

– We suggest a novel 3-D CNN based model for news recommendation in which we utilize the user-item based interaction as well as the content of the read news articles.
– We perform experiments to examine the suitability of our model for the task of news recommendation and compare it with a certain set of approaches.

1.3.3 Thesis Workflow

This thesis has been written in a way that a particular chapter can be read in a standalone manner i.e reading of prior chapters should not be required to understand the contents of that particular chapter. In order to enable such flexibility, certain parts like dataset description and baselines might be repeated across chapters. But this is done only to provide more options while reading.

The rest of the thesis is organized as follows. In Chapter 2, we provide a brief summary of the existing methods for general recommendation system. We then look at the state-of-the-art approaches and dive into details of the ones based on neural networks. In Chapter 3, we talk about the MDP based approach we used for recommendation. We discuss theory and provide the results our of experiments. In Chapter 4, we introduce our user profiling based approach for recommendation. We present its theory and efficacy. In Chapter 5, we introduce a recurrent network based approach for recommendation which is built on top of what has been showcased in Chapter 4. We cover its theory and its utility for the task. In Chapter 6, we present the final version of our news recommendation model which we call RARE. We present its theory and detail out various experiments in order to prove its suitability to the task. Finally, in Chapter 7, we lay out our preliminary and experimental work on 3D CNNs for news recommendation. In this chapter, we also provide a bit of detail on our future work related to this. Finally, we present the conclusion of the thesis and mention some future work.

In section 1.1, we presented the application of recommendation systems and its importance for news aggregators. In section 1.2, we provided a detailed description of what news aggregators are, some functions of recommender systems and tried to identify some challenges associated with that of news recommendation. We then summarized the problem statement and the contributions along with the thesis organization in the following section.
Chapter 2

Related Work

There has been extensive study on recommendation systems with a myriad of publications. However, the exploration of deep neural networks for news recommendation systems has received relatively less scrutiny. In this section, we aim at reviewing a representative set of approaches that are related to our proposed approach.

2.1 Common Approaches for Recommendation Systems

Recommendation systems in general can be divided into collaborative recommendation systems and content based recommendation systems.

2.1.1 Collaborative Filtering (CF)

In collaborative filtering based recommendations, an item is recommended to a user if similar users liked that item. Collaborative filtering can be further divided into user collaborative filtering, item collaborative filtering or a hybrid of both user and item collaborative filtering. Examples of such techniques include Bayesian matrix factorization [34], matrix completion [31], Restricted Boltzmann Machine [35], nearest neighbour modelling [3] etc. In user collaborative methods such as [3], the algorithm first computes similarity between every pair of users based on the items liked by them. Then, the scores of user-item pairs are computed by combining scores of this item given by similar users. Item based collaborative filtering [36], computes similarity between items based on the users who like both items. It then recommends items to the user based on the items she has previously liked. Finally, in user-item based collaborative filtering, both the users and the items are projected into a common vector space based on the user-item matrix and then the item and user representation are combined to find a recommendation. Matrix factorization based approaches like [31] and [34] are examples of such a technique. One of the major drawbacks of collaborative filtering is its inability to handle new users and new items, a problem which is often referred as the cold-start issue.
2.1.2 Content-based Filtering

Another common approach for recommendation is content-based recommendation. In this approach, features from user’s profile and/or item’s description are extracted and are used for recommending items to users based on these features. The underlying assumption is that the users tend to like items that they liked previously. In [23], each user is modeled by a distribution over news topics that is constructed from articles she liked with a prior distribution of topic preference computed using all users who share the same location. A major advantage of using content-based recommendation is that it can handle the problem of item cold-start as it uses item features for recommendation. For user cold-start, a variety of other features like age, location, popularity aspects could be used.

Some recent research has given rise to adoption of word embedding based techniques in content-based recommendation scenarios. Authors in [25], use Word2Vec [24] in order to create a user profile based on the learned embeddings. The experimental results of this work show that a model based on word embeddings is comparable to that of well-performing algorithms based on Collaborative Filtering and Matrix Factorization.

In the following we discuss recommendation works which use neural networks.

2.2 Neural Network based Recommendation

In this section we aim to briefly discuss some important work which utilize neural neural networks for recommendations in general.

2.2.1 Restricted Boltzmann Machine for CF

Early pioneer work which used neural network was done in [35], where a two-layer Restricted Boltzmann Machine (RBM) was used to model users explicit ratings on items. The authors presented efficient learning and inference procedures for this class of models and demonstrate that RBMs can be successfully applied to the Netflix data set, containing over 100 million user/movie ratings. They also showed that RBMs were slightly able to outperform carefully-tuned SVD models. The work has been later extended to model the ordinal nature of ratings [28].

2.2.2 Autoencoder based Recommendation System

Recently autoencoders have become a popular choice for building recommendation systems [5][38][40].

1. AutoRec : In [38] authors propose a novel autoencoder framework for collaborative filtering (CF). The authors show that empirically, it is compact and efficiently trainable and is able to outperform state-of-the-art methods on the Movielen and Netflix datasets. The idea here is to basically learn hidden structures that can reconstruct a user’s ratings given her historical ratings as inputs.
2. Stacked Denoising Autoencoders: In [40] the authors try to address the problem of using sparse inputs to a collaborative filtering model. They utilize a neural network architecture which computes a non-linear matrix factorization from sparse rating inputs. This neural network architecture consists Stacked Denoising Autoencoders. In terms of user personalization, this approach shares a similar spirit as the item-item model [26][36] that represent a user as her rated item features.

While previous work has lent support for addressing collaborative filtering, most of them have focused on observed ratings and modeled the observed data only. As a result, they can easily fail to learn users preference from the positive-only implicit data which is what is we have available in case of news.

3. CDAE [42]: This is one the works which is most relevant to ours. In [42] a collaborative denoising autoencoder (CDAE) for CF with implicit feedback is presented. In contrast to the DAE-based CF [40], CDAE additionally plugs a user node to the input of autoencoders for reconstructing the user’s ratings. As shown by the authors, CDAE is equivalent to the SVD++ model [20] when the identity function is applied to activate the hidden layers of CDAE.

Although CDAE is a collaborative filtering model, it is solely based on item-item interaction whereas the work which we present here is based on user-item interaction.

2.2.3 Deep Semantic Structured Model (DSSM)

One of the most effective approaches in projecting queries and documents into a common low-dimensional space has been shown in [17]. The model is named as Deep Semantic Structured Model (DSSM) [17] which is effective in calculating the relevance of the document given a query by computing the distance between them. We can very easily draw a parallel between the task of finding relevant documents with respect to a query and that of finding relevant articles for a user which are in accordance with her interests.

Originally DSSM was meant for the purpose of ranking, but since the problem of ranking has very close associations with that of recommendation, DSSM was later extended to recommendation scenarios in [8]. In [8], the authors used DSSM for recommendation where the first neural network contains user’s query history (and thus referred to as user view) and the second neural network contains implicit feedback of items. The resulting model is named multi-view DNN (MV-DNN) since it can incorporate item information from more than one domain and then jointly optimize all of them using the same loss function in DSSM. However, in [8], the features for the users were their search queries and features for items came from multiple sources (e.g Apps, Movies/TV etc.). This makes it less adaptable by a news website as it requires a lot of information outside the news domain.
2.2.4 Neural Matrix Factorization

This is another work which is most relevant to ours. In [13], authors have explored deep neural networks for recommender systems. They present a general framework named NCF, short for Neural Collaborative Filtering that replaces the inner product with a neural architecture that can learn an arbitrary function from the given data. It uses a multi-layer perceptron to learn the user-item interaction function. NCF is able to express and generalize matrix factorization. They then combine the linearity of matrix factorization and non-linearity of deep neural networks for modelling user-item latent structures. They call this model as NeuMF, short for Neural Matrix Factorization.
Chapter 3

Leveraging Moderate User Data for News Recommendation

3.1 Overview

In this chapter, we propose an approach for a news recommendation system which relies on moderately low amount of user data. Popular methods like Collaborative Filtering (Matrix Factorization) require a sufficient amount of user interaction before making predictions. Other neural network based model suffers from a similar problem where due to lack of training data, the parameters for the model are not effectively learnt. This prompts the need of a news recommendation system which can work well with limited amount of data. We first start off with a model which is similar to the bag of words model used in language modeling. We define a set of probability measures and use them for making predictions. We then explore a similar kind of method but with probability measures which are based on discounting. Finally, we propose a novel approach for recommending news items using Markov Decision Process (MDP). Due to the sequential nature of news reading, we choose MDP to model our recommendation system as it is based on a sequence optimization paradigm. Further, we also incorporate factors like article freshness and similarity into our system by explicitly modelling it in terms of reward for the MDP. We compare it with various other methods. On a moderately low amount of data, we see that our MDP-based approach outperforms other techniques. However, there are a few drawbacks of using such an approach. It cannot handle the item cold-start problem and the model is unable to generalize well when presented with a large amount of data.

3.2 Prerequisites

3.2.1 Statistical Language Modeling

The goal of Statistical Language Modeling is to build a statistical language model that can estimate the distribution of natural language as accurate as possible. It includes the development of probabilistic models that are able to predict the next word in the sequence given the words that precede it. Most
widely the n-gram model is used today. It is formulated as follows:

\[
P(w_1 w_2 w_3 \cdots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \cdots P(w_n|w_1w_2 \cdots w_{n-1})
\]  

(3.1)

where each \(w_i\) represents a word and \(w_1\) to \(w_n\) represent a sequence of words. Overall this can also be written as:

\[
P(w_1 w_2 w_3 \cdots w_n) = \prod_i P(w_i|w_1 w_2 \cdots w_{i-1})
\]  

(3.2)

### 3.2.2 Markov Decision Process (MDP)

An MDP is a model for sequential stochastic decision problems. As such, it is widely used in applications where an autonomous agent is influencing its surrounding environment through actions. An MDP is by definition a four-tuple: \(< S, A, Rwd, tr >\), where \(S\) is a set of states, \(A\) is a set of actions, \(Rwd\) is a reward function that assigns a real value to each state/action pair, and \(tr\) is the state-transition function, which provides the probability of a transition between every pair of states given each action. In an MDP, the decision-makers goal is to behave so that some function of its reward stream is maximized typically the average reward or the sum of discounted reward. An optimal solution to the MDP is such a maximizing behavior. Formally, a stationary policy for an MDP is a mapping from states to actions, specifying which action to perform in each state. Given such an optimal policy \(\pi\), at each stage of the decision process, the agent need only establish what state \(s\) it is in and execute the action \(a = \pi(s)\). Various exact and approximate algorithms exist for computing an optimal policy. Below we briefly review the algorithm known as policy-iteration, which we use in our implementation. A basic concept in all approaches is that of the value function. The value function of a policy \(\pi\), denoted \(V_\pi\), assigns to each state \(s\) a value which corresponds to the expected infinite-horizon discounted sum of rewards obtained when using using \(\pi\) starting from \(s\). More formally this can be represented as:

\[
V_\pi(s) = Rwd(s, \pi(s)) + \gamma \sum_{s_j \in S} tr(s, \pi(s), s_j)V_\pi(s_j)
\]  

(3.3)

Solving MDPs is known to be a polynomial problem in the number of states via a reduction to linear programming. It is usually more natural to represent the problem in terms of states variables, where each state is a possible assignment to these variables and the number of states is hence exponential in the number of state variables. This well known curse of dimensionality makes algorithms based on an explicit representation of the state-space impractical. Thus, a major research effort in the area of MDPs during the last decade has been on computing an optimal policy in a tractable manner using factored representations of the state space and other techniques.

These were the two prerequisites required for understanding the solution that we proposed. We now describe a few approaches which then leads on to our MDP based approach.
3.3 Proposed Approach

In this section, we aim to describe the overall architecture of our model. Firstly, we mention the notations used. We then move on to explain an approach which leads to the MDP based approach. We then discuss the construction of our MDP based model and its the semantic measure used for constructing its reward. Next, we discuss the semantic measure used to find out the similarity between different news articles. Finally, we explain how we recommend articles to a user based on the decisions made by the MDP.

3.3.1 Bag of News Model

In this model, we consider transitions between adjacent states of a user. Intuitively, a state corresponds to one news item. However, this representation is limited to the previous news story and does not take into account all the news items the reader has seen since the beginning of her visit. Therefore, we adopt a $p$-gram model of news stories. We introduce here a new parameter, namely the past $p$. The past indicates how much historical information we consider for generating recommendations. For instance, when $p = 3$ we have a trigram model. More formally, let $N$ be the set of news stories and $n_i \in N$ a news item. Let $S$ be the set of states, and $s_i \in S$ a state. We define a state $s$ as a sequence of $p$ news items and write $s = < n_1, n_2 \cdots n_p >$ for a given sequence $n_1 \rightarrow n_2 \rightarrow \cdots \rightarrow n_p$. We represent a user's visit as a sequence of states. For instance in the case of $p = 3$, an anonymous user who reads news stories in the sequence $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4$ is represented as a path $< n_1, n_2, n_3 > \rightarrow < n_2, n_3, n_4 >$. So far, we model the visits as an ordered sequence of clicks on some news items, and we call it sequence-of-news recommender system. Alternatively, we could also consider a bag of news instead of an ordered sequence. For instance with a 2-gram model, two readers with the following histories $n_1 \rightarrow n_2$ and $n_2 \rightarrow n_1$ would correspond to the same state $< n_1, n_2 >$, and no longer to $< n_1, n_2 >$ and $< n_2, n_1 >$. By doing so, we make the model simpler and more flexible because we will have more data to generate the recommendations for a given state. In other words, the history of a reader does not need to match exactly the history of someone else, but it has to contain the same news items. We call this overall method as the bag-of-news model.

We experimented by using various $p$-grams. However, we found that using $p=2$ gave us best results. We named this the Bigram Bag of News.

3.3.2 MDP based Recommendation

Here we give a detailed explanation of our MDP based approach. But first, we explain the semantic similarity that we use in the construction of our reward function.
3.3.2.1 Semantic Similarity

We use a method similar to that of Semantic Recommendation mentioned in [18] for finding out similarity between different news articles. In traditional forms of text comparison all words in the text are considered. In our results we see that using traditional forms of text comparison like cosine similarity (in KNN) does not help much. One of the reasons for this is that it becomes difficult for such methods to identify the relatedness between two different words. For example, there is no way to explicitly identify the relatedness between Roger Federer and Rafael Nadal using traditional cosine. Users who are interested in the former, will be interested in reading news about the latter as well. To overcome this we create an ontology which is based on the concepts (topics) which best describe the given article.

For each news article in our data, we have at least 1 and at most 5 topics that best describe that article. We call these topics as concepts and using these concepts we find the semantic similarity between two news articles. Suppose, we have two news articles $n_i$ and $n_j$. We define a concept set of a news article $n_i$ as all the topics that describe that article (given in the data). We refer to this concept set by $C(n_i)$. Therefore, the concept set of $n_i$ is:

$$C(n_i) = [c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5}]$$

Now, we define concept equivalence set of a concept $c$ as the set containing all the concepts which occurred at least once with $c$ across different news articles. Let us denote the concept equivalence set of $c$ by $CE(c)$. Therefore,

$$CE(c) = \bigcup_{n_i \in N_c} C(n_i)$$

where, $N_c$ is the set of all news articles which have $c$ as a concept. Now, we use Jaccard’s similarity based on the concepts contained in the two news articles. Therefore,

$$ssim(n_i, n_j) = \frac{(\bigcup_l CE(c_{il})) \cap (\bigcup_l CE(c_{jl}))}{(\bigcup_l CE(c_{il})) \cup (\bigcup_l CE(c_{jl}))}$$

(3.4)

where, $c_{il}$ is the $l$th concept of article $i$.

One of the advantages of using such a measure is that, even for the users whose interaction with the website is very less, we can still come up with a plethora of related topics which might be of interest to them.

3.3.2.2 MDP Description for News

In [39], it has been argued that it is better to view the problem of recommendation as that of sequential optimization problem, and hence MDP is better suited for it. As we saw earlier, an MDP is by definition a four tuple: $< S, A, Rwd, TP >$, where $S$ is the set of states, $A$ is the set of actions, $Rwd$ is the reward function, and $TP$ is the transition probability from one state to the other. The decision makers goal in MDP is to maximize its reward stream.
There are two problems that we come across while using an MDP for news recommendation. Firstly, the state space is too large because of the vast number of news articles. Typically, an MDP solver requires a matrix of size $AxNxN$, where $N$ is the number of news articles. Hence, formulating the set of actions becomes crucial both for scalability as well as accuracy. In [39], the size of $A$ becomes equal to $N$. We change this by introducing our own set of actions. This is discussed later in this section.

We treat each read news article as a state. The transition probability is denoted by $TP(n_i, n_j)$, where $n_i, n_j \in N$. We use $U_i$ of each user along with an exponential discounting function to calculate $TP(n_i, n_j)$ as follows:

$$TP(n_i, n_j) = \frac{\text{count}(n_i, n_j)}{\sum_{n \in N} \text{count}(n_i, n)}$$  \hspace{1cm} (3.5)

$$\text{count}(n_i, n_j) = \sum_{k=0}^{K} \text{trans}_k(n_i, n_j) \times e^{-(k-1)}$$  \hspace{1cm} (3.6)

where, $\text{trans}_k(n_i, n_j)$ denotes the number of times exactly $k - 1$ articles were read from $n_i$ to $n_j$.

We use exponential discounting with the assumption that the current article that is being read by the user will have partial effects on the type of articles that the user will read in the near future. Also, this helps us to tackle the problem of varying interests of user.

Now, given the reading sequence $U_i$ of each user, we further define action $A_i \in A$. $A_i$ denotes the action in which $i$ number of clicks were required for transitioning between two states. For example: suppose a user read articles in the following sequence $n_1, n_2, n_3, n_4$. Then, an action $A_2$ over the state $n_2$ would lead us to the state $n_4$. Our action set consists of five actions ranging from $A_1$ to $A_5$. We then model our reward function.

As mentioned earlier, the lifespan of a news article is less. Hence, a user would have more incentive in reading an article which is published later in time than the one which is currently being read. Keeping this in mind, we define our first reward function as:

$$Rwd_1(A_i, n_j, n_k) = \begin{cases} \frac{dt(n_k) - dt(n_j)}{N} & \text{trans}_i(n_j, n_k) > 0 \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.7)

where $dt(n_i)$ denotes the discovery time of news article $n_i$. Secondly, to capture the similarity between two news article, we use the semantic similarity measure as follows:

$$Rwd_2(A_i, n_j, n_k) = \begin{cases} \text{ssim}(n_i, n_j) & \text{trans}_i(n_j, n_k) > 0 \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.8)

where $\text{ssim}(n_i, n_j)$ denotes the semantic similarity measure mentioned in 3.2. We then define a third reward function which is a weighted combination of the above two:

$$Rwd_3(A_i, n_j, n_k) = \alpha \times Rwd_1(A_i, n_j, n_k) + (1 - \alpha) \times Rwd_2(A_i, n_j, n_k)$$  \hspace{1cm} (3.9)

where $\alpha$ is a hyperparameter. It is tuned using a cross validation set. We use the Policy Iteration method using the MDP-toolbox to calculate the optimal policy[4].
3.3.3 Recommending Articles

The MDP gives us the information about the best action to be undertaken at a given state. To recommend articles we look at the reading history of each user. We denote the amount of history to be considered for recommendation as $RH$. For example if the reading sequence of a user was $[n_1, n_2, n_3, n_4, n_5]$, and our chosen $RH$ is 2, then we only use the decisions given by the MDP for states $n_4$ and $n_5$. Here each decision is an action. An action leads to another set of states. We then choose the top five states which have maximum probabilities of being transitioned to from the current $RH$ state. For example, in the above sequence, suppose the decision on $n_4$ is $A_1$ and that of $n_3$ is $A_2$. We would then consider all those states which could be reached from $n_4$ by performing the action $A_1$. Similar would be the case for $n_3$ when executing an action $A_2$ over it. It could be possible that both $n_3$ and $n_4$ lead to the same state, say $n_s$. In such cases we associate the probability of $n_s$ with the state for which $TP(n_i, n_s)$ is maximum. Finally, we get a list of states with decreasing order of probability values. We select the top 5 states from this list as our set of recommended articles. This list is basically a ranked set of recommended articles.

3.4 Data and Experiments

For the purpose of our study we received data from a social network aggregation website called Veooz.com\(^1\). The data contains news articles read by users in a sequential manner. This was collected across a period of three months. The news articles were in English. Each news article came tagged with at least one and at most five topics. The way in which these topics are found out is proprietary to the website. We removed all the users who had read less than 6 articles. Finally, the data contained 660 users, 1826 unique articles. We randomly select data of 60 users as our cross validation set for learning the hyper parameters involved in MDP.

In the other setting, we exclude the last article read by the user and use the rest for training. We then recommend a ranked set of $x$ articles. The performance of the ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG). As such, HR@$x$ intuitively measures whether the test item is present in the top-$x$ list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We fine tune our hyperparameters using the cross validation set and thus, set $RH = 6$, $\alpha = 0.7$ and $K = 5$.

In order to make our results more convincing, we compare it with several state-of-the-art-methods. These include KNN as mentioned in [36] which uses similarity between the articles already read by the user and a new article to recommend news articles. RegSVD[27] is a matrix factorization based approach for recommending news articles. We then incorporate a bigram based model as mentioned in [9]. We modify the bigram based model to use our discounted probabilities as described in equations

\(^1\)https://www.veooz.com
Table 3.1 Comparative Performance of MDP vs others

<table>
<thead>
<tr>
<th>Method</th>
<th>HR@10</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Popular</td>
<td>1.16</td>
<td>0.33</td>
</tr>
<tr>
<td>RegSVD</td>
<td>1.83</td>
<td>0.833</td>
</tr>
<tr>
<td>KNN</td>
<td>1.83</td>
<td>1.16</td>
</tr>
<tr>
<td>Bigram Bag of News</td>
<td>3.16</td>
<td>1.33</td>
</tr>
<tr>
<td>Discounted Bigram</td>
<td>5.33</td>
<td>1.83</td>
</tr>
<tr>
<td>MDP($R_{wd1}$, $RH = 6$)</td>
<td>6.83</td>
<td>3.5</td>
</tr>
<tr>
<td>MDP($R_{wd2}$, $RH = 6$)</td>
<td>7.16</td>
<td>3.66</td>
</tr>
<tr>
<td>MDP($R_{wd3}$, $RH = 6$)</td>
<td>7.66</td>
<td>4.16</td>
</tr>
</tbody>
</table>

(2) and (3). We then finally evaluate our model with the three different reward functions mentioned. For some of the baselines we use the LibRec implementation[11]. Others are implemented in python.

3.5 Results and Discussion

Table 3.1 summarizes the results. By looking at the performance of the most-popular baseline, we can say that popularity does not necessarily correlate with the users interests. Secondly, we see that both KNN based and MF-based methods perform poorly in case of their HR@10 accuracy. KNN uses the cosine similarity metric and hence is unable to generalize well because users have very less reading history. It is not able to capture the actual interests of users. One of the reasons for the MF-based methods to perform poorly is their inability to identify the important latent factors. With such a kind of data, where there is no severe user activity, MF-based methods seem to fail to identify latent factors which are crucial to the process of recommendation.

Next, we see that the simple bigram and discounted bigram models perform better than the above discussed. One of the most important reasons for this is that, it explicitly takes into account the reading sequence of a user. These methods are similar to a language model and make use of the sequence to recommend a news article. Recommendation is posed as a sequence prediction problem in these cases.

We see that our MDP-based model outperforms the baselines. When modelled using $R_{wd3}$, the MDP performs best. A combination of the information present in the reading sequence, along with the freshness and semantic similarity between the different read articles is fairly able to capture the users interests.

3.6 Summary

News aggregator websites, which are new to the market cannot generate a lot of data from user interaction. Most of the state-of-the-art methods heavily depend on huge amounts of data. Such websites would at least want their existing userbase to remain intact and in order to do so would deploy a recommender system. However, by the experiments conducted we can see that the baselines do not perform
very well in such a scenario (with less data). Here we show that by using the sequential nature of news reading combined with other aspects like semantic similarity and freshness, we are fairly able to generalize and provide better recommendations. Better results show that the semantic similarity measure is able to capture the diverse interests of users. Also, posing the problem of recommendation as that of sequential optimisation provides us with better results. Another advantage of such a system is that if we solely use $Rwd_1$ to model our MDP, then the entire recommendation system becomes language agnostic. Hence, for news aggregators which combine news belonging to a variety of languages, this could be helpful.
Chapter 4

User Profiling based Neural Network for Temporal News Recommendation

4.1 Overview

In Chapter 3, we presented an approach to tackle the problem of news recommendation when presented with a moderately low amount of user data. Although it had decent results in a low data scenario, it was not effective enough when the data scale was increased. This prompted us to start exploring other methods.

From our previous approach, we understood that considering the sequential reading history of a user plays an important role. Apart from this, knowing the weight/importance of each article in a sequence would help in determining the various interests of the user. For example: if the last five articles read by a user has been related to sports then certainly we would like to recommend some more articles which are related to it. There is a temporal sense in the reading pattern of the users. Capturing this would help us refine our recommendation as this would help the model to distinguish between the various interests of the user over time.

The other class of problems that we try to address with this approach is that of cold start as well as that of relying on implicit-only feedback. Other recommendation systems, like the ones which recommend movies, have ratings of each user and generally minimize the squared error for learning the model parameters. However, such a method is not suitable when presented with implicit feedback data.

Overall, our goal here was to come up with a recommendation system which can handle the temporal changes in users’ interest, can solve the cold start problem and can be effectively learnt using implicit feedback data.

In this work, we consider the problem of recommendation in a ranking based approach. We use the user-item interaction and the content of the news in order to capture the similarity between users and items (news). We do not use any explicit information about the user in order to model her interests. We only focus on the implicit information provided by the user, i.e., whether a user read a given article or not.
The sequence in which the articles are read by the user encapsulates information about the interests of the user. In order to capture the preferences from the sequence and create a user profile from it, we need a mechanism which should be capable of handling long term dependencies. We use the following steps to achieve this.

1. First, we learn the doc2vec [21] embeddings for each news article.

2. We then choose a specific amount of reading history for all the users.

3. Finally, we combine the doc2vec embeddings of each of the articles present in the user history using certain heuristics which preserves the temporal information encoded in the sequence of articles read by the user.

Then, in order to capture the similarity between users and items, we need to be able to project them to the same latent space. We adapt Deep Structured Semantic Model (DSSM) [17] for this task. DSSM was originally proposed for the task of web document ranking. Later, it was adapted for the task of recommendation in [8]. However, in [8] the features for the users are their search queries and features for items come from multiple domains (e.g. Apps, Movies/TV etc.) which makes it difficult for a news website to directly adapt it as a lot of information outside the news domain is required. Then, for learning the parameters of the model, we use a ranking based objective function. Finally, for recommending news articles to the users, we use the computed inner product between user and item latent vectors.

To summarize, the contributions of our work are as follows.

1. We use doc2vec embeddings of each news article in order to come up with user profiles for each user which encapsulates information about the varying interests of the user over time.

2. We use a deep neural architecture for news recommendation in which we utilize the user-item interaction as well as the content of the news (items) to model the latent features of users and items.

3. We perform experiments to demonstrate the effectiveness of our model for the problem of news recommendation. We then do the same for when the user has had very little interaction with items.

4. We also address the effectiveness of our model in solving the item cold-start problem.

4.2 Model Architecture

In this section we discuss briefly the components of our model. We first discuss user profiling, followed by the DSSM architecture. We then provide the training criteria for our model. Figure 4.1 illustrates the architecture of the proposed model.
model architecture for the modified DSSM. \( \oplus \) represents the operation using which the user profile is created. The left side of the model creates the user profile, while the right side of the model provides positive and negative instances for training the model parameters. \( r_1 \ldots r_R \) represent articles present in the user history. \( \text{item}^+ \) and \( \text{item}^- \) represents the positive and negative instances respectively.

4.2.1 User Profiling

We first define a set of notations useful in understanding the creation of user profile. We define the number of articles in the user reading history to be \( R \). The doc2vec embeddings of each article in the history is represented by \( r_h \) where \( 1 \leq h \leq R \). Each vector is of size 300. The user profile for a user is denoted by \( U \). We now discuss three kinds of operations using which we create the user profiles.

1. Centroid

In this method, we find the centroid of the embeddings of the articles present in the reading history of the user. The centroid then represents the user profile.

\[
U = \frac{1}{R} \sum_{h=1}^{R} r_h
\]  

(4.1)

2. Discounting

In this we first discount each of the vectors present in the user reading history by a power of 2 such that an article read at time \( t - 1 \) carries half the weight compared to an article read at time \( t \).
We then take an average of all the vectors.

$$ U = \frac{1}{R} \sum_{h=1}^{R} \frac{r_h}{2^{R-h}} $$  \hspace{1cm} (4.2)

### 3. Exponential Discounting

In this we discount each of the vectors present in the user reading history by a power of $e$ such that an article read at time $t - 1$ carries $1/e$ the weight compared to an article read at time $t$. We then take an average of all the vectors.

$$ U = \frac{1}{R} \sum_{h=1}^{R} \frac{r_h}{e^{R-h}} $$  \hspace{1cm} (4.3)

We experiment with different kinds of methods for creating a user profile in order to understand the temporal patterns present in the user reading history. The idea behind discounting and exponential discounting is to give more preference to the recently read articles and lesser preference to those read far away in the past.

### 4.2.2 Prerequisite

The Deep Semantic Structured Model (DSSM) [17] was proposed for the purpose of ranking. Essentially, DSSM can be viewed as a multi-view learning model that often composes of two or more neural networks for each individual view. In the original two-view DSSM model, the network on the left side was meant for query representation, whereas the networks on the right side were meant for representing the documents. The input to these networks could be of any arbitrary type like letter-tri-gram in the original paper or bag of unigrams used in [8]. Each input vector after that goes through a non-linear transformation in the feed-forward neural network to output an embedding vector, which is smaller in size than the input vector. The learning objective of the DSSM is to maximize the cosine similarity between the two output vectors. For the purpose of training, a set of positive examples and randomly sampled negative examples are generated in order to minimize the cosine similarity based loss.

### 4.2.3 Modified DSSM

In this work, we modify the DSSM in the following ways.

1. Instead of using letter-tri-gram, we use the doc2vec embeddings of each of the news articles as input.

2. The input to the left side of the model is the user reading history $R$. The doc2vec embeddings of each of the article present in the user reading history is passed as inputs. A user profile is then computed using these embeddings.
3. The input to the right side of the model consists of 1 positive instance (an article read by the user apart from the articles already present in the user history) and \(n\) negative articles. The \(n\) negative articles are randomly sampled.

### 4.2.4 Learning

Typically in matrix factorization, to learn the model parameters, existing pointwise methods [23] perform regression with a squared loss. This is based on the assumption that observations are generated from a Gaussian distribution. However, in [13] it has been shown that such a method does not tally well when we have implicit data available to us. Also, in [13] it has been shown that a ranking based objective function is more suitable for the task of recommendation. Keeping these two aspects in mind, we adapt the loss function used in DSSM. We first compute the posterior probability of a clicked news item given a user from the relevance score using a softmax function as follows.

\[
P(item^+ | u) = \frac{\exp(R(u, item^+))}{\sum_{item} \exp(R(u, item))}
\]  

(4.4)

where \(item^+\) denotes the item that was clicked by the user and \(R()\) represents the inner product function. We then maximize the likelihood of the clicked news items given the user with the following loss function.

\[
L(\Lambda) = - \log \prod_{u, item^+} P(item^+ | u)
\]  

(4.5)

where, \(\Lambda\) represents the parameters of our model.

### 4.3 Experiments

In this section we delineate our conducted experiments in order to answer the following questions.

1. How do the different profiling methods help in improving the overall recommendation?

2. How does our model perform against state-of-the-art methods?

3. How well does our model perform when the user has not had many interactions with the items, i.e., for users who have read very less number of articles?

4. How well does our model perform in recommending items which have not had any interaction by any user (item cold start problem)?
4.3.1 Dataset

For this work we use the dataset published by CLEF NewsREEL 2017. CLEF NewsREEL provides an interaction platform to compare performance of different news recommender systems in an online as well as in an offline setting [16]. As a part of their evaluation for offline setting, CLEF shared a dataset which captures interactions between users and news stories. It includes interactions of eight different publishing sites in the month of February 2016. The recorded stream of events include 2 million notifications, 58 thousand item updates, and 168 million recommendation requests. The dataset also provides other information like the title and text of each news article, time of publication etc. Each user can be identified by a unique id. For our task, we needed to find out the sequence in which the articles were read by the users. Along with this we also find out the content of each of these read articles. Since we rely only on implicit feedback we only need to know whether the article was read by a user or not.

4.3.2 Experimental Settings

As mentioned earlier, we use the dataset provided by CLEF NewsREEL 2017. We extract the sequence in which the articles were read by the users. For each article we concatenate the body and the text and use gensim [29] to learn doc2vec [21] embeddings for those. The size of the embeddings is set to 300. In the given dataset, almost 77% of the users have read less than 3 articles. We choose users who have read in between 10–15 (both inclusive) articles for training and testing our model for item recommendation. The frequency of users who have read more than 15 articles varies extensively and hence we restrict ourselves to the upper bound of 15. We then choose users who have read 2–4 articles for testing our model for the case when the user has had very little interaction with the items (user cold start problem). For the item cold start problem, we test it on users who have read in between 10–15 articles.

4.3.2.1 Evaluation Protocol

To evaluate the performance of the recommended item we use the leave-one-out evaluation strategy which has been widely adopted in literature [2][14][30]. For each user we held-out her latest interaction as the test set and utilized the remaining data for training. Since it is time consuming to rank all items for every user during evaluation, we followed the common strategy [20] that randomly samples 100 items that the user has not interacted with, and then ranking the test item among the 100 items. The performance of a ranked list is judged by two metrics: Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG) [12]. We truncated the rank list at 10 for both metrics. As such, the HR@$k$ intuitively measures whether the test item is present in the top-$k$ list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score.
4.3.2.2 Baselines

- **BPR** [30]. This method optimizes the matrix factorization method with a pairwise ranking loss, which is tailored to learn from implicit feedback. We report the best performance obtained by varying the learning rate.

- **eALS** [14]. This is a state-of-the-art matrix factorization method for item recommendation. It optimizes the squared loss (between actual item ratings and predicted ratings) and treats all unobserved interactions as negative instances and weighting them non-uniformly by item popularity.

- **NeuMF** [13]. This is a state-of-the-art neural matrix factorization model. It treats the problem of generating recommendations using implicit feedback as a binary classification problem. Consequently, it uses the binary cross-entropy loss to optimize its model parameters.

Our proposed method is based on modeling user-item relationship, hence we mainly compare it with other user-item models only. We leave out the comparison with other models like SLIM [26] and CDAE [42] because these are item-item models and hence performance difference may be caused by the user models for personalization.

4.3.2.3 Parameter Settings

We implemented our proposed method using Keras[6]. As mentioned earlier, for each user who had read in between 10-15 (both inclusive) articles we held out the last read article for our test set. We then construct our labeled set as follows.

1. We first define the reading history that we want to use. We denote the reading history by $R$.

2. For each user, we use $R$ number of read articles as inputs to the left side of the model. Leaving the last read article out, the remaining articles are used as positive samples for the right view of the model.

3. For each positive instance of a user, we randomly sample $n$ negative instances (news items that the user has not interacted with) which are used as inputs for the item view of the model. We experimentally set the number of negative instances $n$ to be 4.

We then randomly divide the labeled set into training and validation set in a 4:1 ratio. This helps us to ensure that the two sets do not overlap. We tuned the hyper-parameters of our model using the validation set. The model and all its variants are learned by optimizing the log loss of Equation 5. We initialize the fully connected network weights with the uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [10]. We used a batch size of 256 and used adadelta [43] as a gradient based optimizer for learning the parameters of the model.
<table>
<thead>
<tr>
<th>K</th>
<th>Avg</th>
<th>Discounting</th>
<th>Exponential Discounting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@K</td>
<td>NDCG@K</td>
<td>HR@K</td>
</tr>
<tr>
<td>1</td>
<td>0.447</td>
<td>0.447</td>
<td>0.474</td>
</tr>
<tr>
<td>2</td>
<td>0.635</td>
<td>0.566</td>
<td>0.644</td>
</tr>
<tr>
<td>3</td>
<td>0.710</td>
<td>0.603</td>
<td>0.711</td>
</tr>
<tr>
<td>4</td>
<td>0.744</td>
<td>0.618</td>
<td>0.744</td>
</tr>
<tr>
<td>5</td>
<td>0.769</td>
<td>0.627</td>
<td>0.767</td>
</tr>
<tr>
<td>6</td>
<td>0.784</td>
<td>0.633</td>
<td>0.785</td>
</tr>
<tr>
<td>7</td>
<td>0.798</td>
<td>0.637</td>
<td>0.798</td>
</tr>
<tr>
<td>8</td>
<td>0.810</td>
<td>0.641</td>
<td>0.810</td>
</tr>
<tr>
<td>9</td>
<td>0.819</td>
<td>0.644</td>
<td>0.821</td>
</tr>
<tr>
<td>10</td>
<td>0.830</td>
<td>0.647</td>
<td>0.830</td>
</tr>
</tbody>
</table>

Table 4.1 Performance of our model using different user profiling operations

![HR@K performance of our model vs state-of-the-art models](image)

Figure 4.2 HR@K performance of our model vs state-of-the-art models
4.4 Results and Discussion

From Table 4.1 we can see the performance of our model when using three different kinds of methods for user profiling. We observe that the Discounting based method for user profiling had better results in terms of HR as well as NDCG in most of the cases. One of the main reasons for this could be that, since discounting gives more weight to the recently read articles, it adapts to the user’s temporal changes in interests in a better fashion.

Figure 4.2 and Figure 4.3 show the performance of the Top-K recommended lists where the ranking position $K$ ranges from 1 to 10. We leave out the variants of our own model here for comparison and only use the best performing model, i.e., the discounting based model. From the figures, it can be clearly seen that our model shows consistent improvements over the other methods across all positions. The reason for this can be attributed to the fact that apart from accounting for the user’s general preferences we also account for the users changing interests and the extent of those interests which the baselines do not incorporate directly. Although, our model shows significant improvements in terms of HR when $K$ varies from 1 to 5, slowly the performance of the baselines become similar to that of our proposed method when $K$ is 10.

We observe major improvements in the NDCG scores of our model. There is an approximate 20% improvement over NeuMF. The reason for this is the loss function of Equation 5 used by our model. The loss function which is optimized for ranking, helps the model to recommend a better ranked list of items. There is however no significant difference between the performance of the baselines.

We then evaluated our model for the cold start cases. The results are shown in Figure 4.4 and Figure 4.5. For this task we segregated users who had read a new article in the end, i.e., they read articles
Figure 4.4 HR@K performance of our model on cold start cases

Figure 4.5 NDCG@K performance of our model on cold start cases
which had never been seen before they read it. We found out that the number of such users were 74. Out of these 74 users, at an HR@10 we observe that around 33% of the time we were able to recommend that article. This promises us that our model is well suitable for handling the item cold-start problem. For user cold-start, we test our learned model over users who had read articles in between 2 to 4 (both inclusive). The HR@10 score was around 47%. We see a gradual increase in the hit rates as we increase the value of k. The results promise the efficiency of our model to handle the problem of user cold start as well.

4.5 Summary

In this work, we tackle the problem of changing users’ interests by first coming up with a user profile and then using it to learn the parameters of DSSM. The parameters of the model are learnt by considering only implicit feedback of users. The content-embeddings help us in generalizing the user preferences. We then also show the effectiveness of our model in recommending articles to users who have a very small reading history. The success of this prompts us towards the observation that content is very relevant in modeling user preferences even when the user has a very low reading history. Next, we also show that our model is very effective in solving the item cold-start problem as well.
Chapter 5

Deep Neural Architecture for News Recommendation

5.1 Overview

In Chapter 4, we utilized heuristics based profiling methods to understand the temporal changes in users’ interest. In this chapter, we build on top of what was done in the previous one with the motive of developing a better mechanism to identify the various sorts of interests of a user. The major difference in this approach is in the way in which we model the sequential information of the user.

We use a similar approach as the previous one in which we utilize the user-item interactions and the content of the news to capture the similarity between users and items (news). We only focus on implicit feedback (which indirectly reflects the preference of the users) provided by the users, i.e. whether they have read a given article or not and in what sequence were those articles read by them.

The sequence in which the articles are read by the user encapsulates information about the interests of the user. Capturing the interests of the user from the sequence of read articles requires a component which should be capable of learning long-term dependencies. LSTMs in general have shown to be suitable for this particular task [15][41]. To capture both static and dynamic interests which the user has developed over time, we use bidirectional LSTMs [37]. We chose a specific amount of reading history of each user as input to the LSTMs. Once these interests are captured, we then need to know the extent of each of the user’s interests. We incorporate a neural attention mechanism [1] for this purpose. Then, in order to capture the similarity between users and items, we need to be able to project them to the same latent space. We adapt Deep Structured Semantic Model (DSSM) [17] for this. DSSM was originally used for the task of web document ranking. Later, it was adapted for the task of recommendation in [8]. However, in [8] the features for the users are their search queries and features for items come from multiple domains (e.g. Apps, Movies/TV etc.) which makes it difficult for a news website to directly adapt it as a lot of information outside the news domain is required. Then, for learning the parameters of the model we use a ranking based objective function. Finally, for recommending news articles to the users we use the computed inner product between user and item latent vectors.

To summarize, the contributions in this work are as follows.
1. We present a deep neural architecture for news recommendation in which we utilize the user-item interaction as well as the content of the news (items) to model the latent features of users and items.

2. In order to address the changing interests of the users and the granularity/extent of these interests over time, we incorporate attentional bidirectional LSTMs which in turn helps to model the latent features of the user.

3. We perform experiments to demonstrate the effectiveness of our model for the problem of news recommendation. We then perform experiments to show the effectiveness of our model to solve the problems of user and item cold-start respectively.

5.2 Model Architecture

We first briefly review DSSM and then we provide the description of our model. In 4.2.2 we already discussed DSSM, but here we introduce some more details to better understand our model. We then try to show the relationship between matrix factorization and our approach.

5.2.1 Prerequisite

The Deep Semantic Structured Model (DSSM) [17] was proposed for the purpose of ranking. Essentially, DSSM can be viewed as a multi-view learning model that often composes of two or more neural networks for each individual view. In the original two-view DSSM model, the network on the left side was meant for query representation, whereas the networks on the right side were meant for representing the documents. The input to these networks could be of any arbitrary type like letter-tri-gram in the original paper or bag of unigrams used in [8]. After that, each input vector goes through a non-linear transformation in the feedforward neural network to output an embedding vector, which is smaller in size than the input vector. The learning objective of the DSSM is to maximize the cosine similarity between the two output vectors. For the purpose of training, a set of positive examples and randomly sampled negative examples are generated in order to minimize the cosine loss on positive examples. In [8], authors used DSSM for recommendation where the first neural network contained the query history of users and the second neural network contained the implicit feedback of items (e.g. News Clicks, App Downloads). The resulting model is named as multi-view DNN (MV-DNN) since it can incorporate item information from more than one domain and jointly optimize them using the same loss function in DSSM.

5.2.2 Recurrent Attention DSSM (RA-DSSM)

In the MV-DNN, the input to the user view was merely the query history of users. In this work, we modify the way in which inputs are sent to the user view in order to adapt it specifically for the case
Figure 5.1 Model Architecture of Recurrent Attention DSSM
of news recommendation. One of the major issues in news recommendation is that of changing user interests. Interests of users can be classified into short term as well as long term interests. Hence, it becomes crucial for a news recommender to identify these interests and recommend accordingly.

LSTMs have shown to be capable of learning long-term dependencies [15][41]. Bidirectional LSTMs on the other hand can capture past and future information effectively. Users interests keep changing over time and at the time of recommendation we need to know the current interest and the long term user interest. Using Bidirectional LSTMs as an encoder helps us to identify interests which the user has taken up recently (short term) as well the long term interests of the user. For each user, we have the sequence in which news articles were read by her. We then choose the first \( R \) read articles for each user and use it as inputs to our bidirectional LSTMs. The forward state updates of the LSTM satisfy the following equations

\[
\begin{align*}
    f_t & = \sigma [W_f [h_{t-1}, r_t] + b_f] \\
    i_t & = \sigma [W_i [h_{t-1}, r_t] + b_i] \\
    o_t & = \sigma [W_o [h_{t-1}, r_t] + b_o] \\
    l_t & = \tanh [V [h_{t-1}, r_t] + d] \\
    c_t & = f_t \cdot c_{t-1} + i_t \cdot l_t \\
    h_t & = o_t \cdot \tanh(c_t)
\end{align*}
\]

Here, \( \sigma \) is the logistic sigmoid function, \( f_t, i_t, o_t \) represent the forget, input and output gates respectively. \( r_t \) denotes the input at time \( t \) and \( h_t \) denotes the latent state, \( b_f, b_i, b_o \) and \( d \) represent the bias terms. The forget, input and output gates control the flow of information throughout the sequence. The backward states are also computed in a similar manner as above. The forward and backward states are then concatenated to obtain the annotations \( (h_1, h_2, h_3, \cdots h_R) \).

We then need to identify the extent/granularity of each interests. Recently in [1], the effectiveness of attention mechanisms has been shown for the task of neural machine translation. The goal of the attention mechanism in such tasks is to derive a context vector that captures relevant source side information to help predict the current target word. In our case, we want to use the sequence of annotations generated by the encoder to come up with a similar context vector that captures the extent of the user’s interests. Though, in a typical RNN encoder-decoder framework [1], a context vector is generated at each time step to predict the target word, in our case, we only need to calculate the context vector for a single time step.

\[
    c_{attention} = \sum_{j=1}^{R} \alpha_j h_j
\]

where, \( h_1, \ldots, h_R \) represents the sequence of annotations to which the encoder maps the sequence of read news articles and each \( \alpha_j \) represents the respective weight corresponding to each annotation \( h_j \).
The user view (left view) of the model can be seen in Figure 1. The input to this is a selected amount of reading history of each user. Each \( r_i \) in the figure is a news embedding of dimension 300.

However, the right view of the DSSM remains the same as can be seen in Figure 5.1. For inputs to the right view of the DSSM, we select one positive sample i.e an article that has been read by the user (apart from those that were used as input to the user view) and \( n \) randomly selected negative samples (articles that have not been read by the user). Each \( item^+, item^- \) used as inputs to the item view is also an embedding of size 300.

### 5.2.3 Learning

Typically in matrix factorization, to learn the model parameters, existing pointwise methods [14][33] perform regression with a squared loss. This is based on the assumption that observations are generated from a Gaussian distribution. However, in [13] it has been shown that such a method does not tally well when we have implicit data available to us. Also, in [22] it has been shown that a ranking based objective function is more suitable for the task of recommendation. Keeping these two aspects in mind, we adapt the loss function used in DSSM [17]. We first compute the posterior probability of a clicked news item given a user from the relevance score using a softmax function

\[
P(item^+ | u) = \frac{\exp(R(u, item^+))}{\sum_{item} \exp(R(u, item))}\]

(5.8)

where \( u \) denotes the user, \( item^+ \) denotes the item that was clicked by the user and \( R \) represents the inner product function. We then maximize the likelihood of the clicked news items given the user with the following loss function

\[
L(\Lambda) = - \log \prod_{u, item^+} P(item^+ | u)
\]

(5.9)

where, \( \Lambda \) represents the parameters of our model.

### 5.2.4 Relation with Matrix Factorization

We now show how we could interpret our model as a special case of matrix factorization, which is one of the most popular model for recommendation and has been investigated extensively in literature.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality \( f \), such that user-item interactions are modeled as inner products in that space. Accordingly, each item \( i \) is associated with a vector \( q_i \in \mathbb{R}^f \) and each user is associated with a vector \( p_u \in \mathbb{R}^f \). For a given item \( i \), the elements of \( q_i \) measure the extent to which the item possesses those factors, positive or negative. For a given user \( u \), the elements of \( p_u \) measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product of the two vectors captures the interaction between the user \( u \) and item \( i \). This approximates the user \( u \)'s rating for the item \( i \), denoted by \( r_{ui} \), leading to the estimate

\[
r_{ui}^* = q_i^T p_u
\]

(5.10)
The major challenge in this is to compute \( q_i, p_u \in \mathbb{R}^f \). We solve this problem by using deep neural networks. The deep neural architecture allows us to learn a non-linear mapping for the users and the items to the same latent space. For computing the mapping for the users, we first use a recurrent network followed by an attention layer. Fully connected layers are then used for bringing in the user and the items to the same latent space. In the final layer of the DSSM, we compute the similarity between the user and the item using the dot product of the non-linear mappings of the input vectors. The user can then be represented as \( \Phi(u) \) and the item can be represented as \( \Phi(i) \) (here \( \Phi \) represents the learnt non-linear mapping). Finally we estimate the rating as,

\[
\hat{r}_{ui} = \Phi(i)^T \Phi(u)
\]  

Although in [13], to compute this similarity the authors resorted to learn any arbitrary function, we learn a non-linear transformation and then utilise the dot product for computing the similarity.

### 5.3 Experiments

In this section we conduct experiments to answer the following questions:

1. Does our proposed model outperform the state-of-the-art implicit collaborative methods? Also, how do the different variations of our model perform for the given task.

2. How does our proposed model work for solving the item cold start problem?

3. How does our proposed model work for solving the user cold start problem?

4. How does our model perform when we use one-hot item encodings instead of using learnt embeddings for individual news articles? How adaptable is our model?

#### 5.3.1 Dataset

For this work we use the dataset published by CLEF NewsREEL 2017. CLEF NewsREEL provides an interaction platform to compare different news recommender systems performance in an online as well as offline setting [16]. As a part of their evaluation for offline setting, CLEF shared a dataset which captures interactions between users and news stories. It includes interactions of eight different publishing sites in the month of February, 2016. The recorded stream of events include 2 million notifications, 58 thousand item updates, and 168 million recommendation requests. The dataset also provides other information like the title and text of each news article, time of publication etc. Each user can be identified by a unique id. For our task, we find out the sequence in which the articles were read by the users. Along with this we also find out the content of each of these read articles. Since, we rely only on implicit feedback we only need to know whether the article was read by a user or not.
For an another experiment, we use the Yahoo WebScope R6B dataset. It contains user click information of 637 news stories displayed on the “Today Module” on Yahoo!’s front page during a consecutive 15-day period from October 2 to October 16, 2011. One of the limitations of this dataset is that it does not contain any additional information like the content of the news stories. We use this dataset to evaluate our model using one-hot item vectors instead of using the embeddings of those items. Less number of read articles makes this dataset suitable for conducting such a kind of experiment.

5.3.2 Experimental Settings

As mentioned earlier, we use the dataset provided by CLEF NewsREEL 2017. We extract the sequence in which the articles were read by the users. For each article we concatenate the body and the text and use gensim [29] to learn doc2vec [21] embeddings for those. The size of the embeddings is set to 300. In the given dataset, almost 77% of the users have read less than 3 articles. We choose users who have read in between 10-15 (inclusive) articles for training and testing our model for item recommendation. The frequency of users who have read more than 15 articles varies extensively and hence we restrict ourselves to the upper bound of 15. We then choose users who have read 2-4 articles for testing our model for the user cold start problem. For the item cold start problem, we again test it on users who have read in between 10-15 articles.

5.3.2.1 Evaluation Protocol

To evaluate the performance of the recommended item we use the leave-one-out evaluation strategy which has been widely adopted in literature [2][14][30]. For each user we held-out her latest interaction as the test set and utilized the remaining data for training. Since it is time consuming to rank all items for every user during evaluation, we followed the common strategy [8][20] that randomly samples 100 items that are not interacted by the user, ranking the test item among the 100 items. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG) [12]. Without special mention, we truncated the rank list at 10 for both metrics. As such, the HR@$k$ intuitively measures whether the test item is present in the top-$k$ list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score.

5.3.2.2 Baselines

We compare our proposed approach with the following methods:

- **ItemPop.** News articles are ranked by their popularity judged by their number of interactions. This is a non-personalized method to benchmark the recommendation performance [30].
• **BPR** [30]. This method optimizes the matrix factorization method with a pairwise ranking loss, which is tailored to learn to learn from implicit feedback. We report the best performance obtained by fixing and varying the learning rate.

• **eALS** [14]. This is a state-of-the-art matrix factorization method for item recommendation. It optimizes the squared loss (between actual item ratings and predicted ratings) and treats all unobserved interactions as negative instances and weighting them non-uniformly by item popularity.

• **NeuMF** [13]. This is a state-of-the-art neural matrix factorization model. It treats the problem of generating recommendation using implicit feedback as a binary classification problem. Consequently it uses the binary cross-entropy loss to optimize its model parameters.

For all the above methods we choose that number of predictive factors which maximize the performance over our given dataset.

Our proposed method is based on modelling user-item relationship, hence we mainly compare it with other user-item models. We leave out the comparison with other models like SLIM [26] and CDAE [42] because these are item-item models and hence performance difference may be caused by the user models for personalization.

### 5.3.2.3 Parameter Settings

We implemented our proposed method using Keras [6]. As mentioned earlier, for each user who had read in between 10-15 (inclusive) articles we held out the last read article for our test set. We then construct our training set as follows:

1. We first define the reading history that we want to use. We denote the reading history by $R_h$.

2. For each user, we use $R_h$ number of read articles as inputs to the user view. Leaving the last read article out, the remaining articles are used as positive samples for the item view (right view) of the model.

3. For each positive instance of a user, we randomly sample $n$ negative instances (news items that the user has not interacted with) which are used as inputs for the item view of the model. We experimentally set the number of negative instances $n$ to be 4.

We then randomly divide the training set into training and validation set in a 4:1 ratio. This helps us to ensure that the two sets do not overlap. We tuned the hyper-parameters of our model using the validation set. All the model and its variants are learnt by optimizing the log loss of Equation 8. We initialise the fully connected network weights with the uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [10]. We used a batch size of 256 and used adadelta [43] as a gradient based optimizer for learning the parameters of the model. Also, it is worth
noticing that, just in the case of NeuMF [13], where the size of the last layer of the deep network determines the number of predictive factors, we can also treat the size of the last layer of our network (just before computing the similarity) as the number of used predictive factors.

5.4 Results and Discussion

Figure 5.2 and Figure 5.3 shows the performance of our model by varying the amount of reading history used as inputs for the user side of RA-DSSM. Overall we see that as we increase the amount of reading history used, the performance also increases. This shows that a user has multiple interests which slowly get captured as the number of articles used for the user view of RA-DSSM is increased.

Interests of the user develop and vary with time and hence we also experimented by concatenating the time at which the articles were read by each user along with the article embeddings and used these as inputs to the model. It was observed that there was no significant change in the performance. One of the prime reasons for this could be that the model is able to encode the aspect of time into itself given its sequential nature.

Figure 5.4 and Figure 5.5 shows the performance of the Top-K recommended lists where the ranking position K ranges from 1 to 10. We leave out the variants of our own model here for comparison and only use the best performing model i.e using RA-DSSM. As from the figure, it can be clearly seen that our model shows consistent improvements over the other methods across all positions. The reason for this can be attributed to the fact that apart from accounting for the user’s general preferences we also account for the users changing interests and the extent of those interests which the baselines do not incorporate directly. We observe major improvements in the NDCG scores of our model. There is an approximate 22% improvement over NeuMF. The reason for this is the loss function of Equation 8 used by our model. The loss function which is optimized for ranking, helps the model to recommend a better ranked list of items. For baseline methods we see that eALS outperforms BPR with a margin of 2%. We also note that ItemPop performs worst which indicates the need for modelling user’s personalized preferences.

We then evaluated our model for the cold start cases as can be seen in Figure 5.6 and Figure 5.7. For this task we segregated users who had read a new article in the end i.e they read articles which had never been seen before they read it. We found out that the number of such users were 74. Out of these 74 users, at an HR@10 we observe that around 35% of the time we were able to recommend that article. This promises us that our model is well suitable for handling the item cold-start problem. For user cold-start, we test our learned model over users who had read articles in between 2 to 4 (inclusive). The HR@10 score was around 50%. We see a gradual increase in the hit rates as we increase the value of k. The results promise the efficiency of our model to handle the problem of user cold start as well.

We then note the effects on our model by varying the kind of recurrent network used. We tested our model by using LSTMs, GRUs (Gated Recurrent Units) [7] and Vanilla RNN. From Figure 5.8 and Figure 5.9, the trend in the performance can be observed as follows: LSTM > GRU > RNN. One of the
Figure 5.2 HR@10 of RA-DSSM in w.r.t User’s Reading History

Figure 5.3 NDCG@10 of RA-DSSM in w.r.t User’s Reading History
Figure 5.4 HR@K of RA-DSSM vs state-of-the-art

Figure 5.5 NDCG@K of RA-DSSM vs state-of-the-art
Figure 5.6 HR@K of RA-DSSM on Cold-Start cases

Figure 5.7 NDCG@K of RA-DSSM on Cold-Start cases
Figure 5.8 HR@K of RA-DSSM for different recurrent units

Figure 5.9 NDCG@K of RA-DSSM for different recurrent units
Figure 5.10 HR@10 performance of RA-DSSM w.r.t the number of factors

Figure 5.11 NDCG@10 performance of RA-DSSM w.r.t the number of factors
Figure 5.12 HR@10 performance of RA-DSSM w.r.t the number of negative samples

Figure 5.13 HR performance of RA-DSSM vs state-of-the-art models on Yahoo WebScope R6B
Figure 5.14 NDCG performance of RA-DSSM vs state-of-the-art models on Yahoo WebScope R6B

<table>
<thead>
<tr>
<th>Method</th>
<th>HR@10</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention BiLSTM with DSSM</td>
<td>0.877</td>
<td>0.664</td>
</tr>
<tr>
<td>BiLSTM with DSSM</td>
<td>0.868</td>
<td>0.653</td>
</tr>
<tr>
<td>LSTM with DSSM</td>
<td>0.866</td>
<td>0.650</td>
</tr>
</tbody>
</table>

Table 5.1 Performance of some variations of our model

reasons for this could be the fact that an LSTM or a GRU is better able to encode the interests of the user. In Table 5.1, we note the performance by adding bidirectional units and attention layer to the LSTM. We note that Attention BiLSTM > BiLSTM > LSTM. We also note that the attention does indeed enable us to capture the extent of interests as it performs slightly better than the bidirectional LSTM.

We also note the changes by varying the number of factors for our model. It should be noted that the last layer of the RA-DSSM network determines the number of factors we are using. This is similar to the way in which NeuMF [13] describes its factors. We choose a factor size of [8, 16, 32, 64] and report the performance of our model with respect to NeuMF, BPR and eALS. From Figure 5.10 and Figure 5.11, we see that even for smaller number of predictive factors RA-DSSM performs better than the others. The expressiveness of the model increases as the number of predictive factors is increased to 64.

Figure 5.12 notes the performance of RA-DSSM when the number of negative instances $n$ is varied. We choose the number of negatives as [1, 4, 8, 12, 16, 20]. We observe that there is no significant change in the HR up to 8. However, after that as we increase the number of negative instances, the performance starts deteriorating. Similar is the case for NDCG. We find that the most suitable choice for $n$ is 4.

Finally, we choose Yahoo WebScope R6B for evaluating our data when the input to our model is one-hot encoded item vectors. Since the dataset contained history of only 637 news article, it was suitable
for this task. From Figure 5.13 and Figure 5.14, we see that our model significantly outperforms the other models in this case as well. This suggests the adaptability of our model for other recommendation scenarios where the item context may not be available at disposal.

From Table 5.2 we can see how user history helps us to provide useful recommendations. In the first example, it can be clearly seen that the user only reads about articles pertaining to football and hence the recommended article is also related to football. In the second and the third example, the user only reads about crime and politics and hence the recommended article is also related to the same.

5.5 Summary

In this work we used deep neural networks for news recommendation. We combined user-item collaborative filtering with the content of the read news articles to come up with our model. We tackled the problem of changing and diverse reading interests of users using a recurrent network combined with neural attention. We also show the effectiveness of our model in solving the problem of user cold-start and item cold-start as well. We then also show the effectiveness of our model when using one-hot item encodings. This shows the adaptability of our model for other recommendation scenarios which purely rely on implicit feedback. In future, we would like to note the effects of learning an arbitrary function instead of using the inner product to calculate the similarity between the user and the item. We would also like to evaluate our model over different recommendation scenarios.
<table>
<thead>
<tr>
<th>User-View Input (User History)</th>
<th>Item-View Input</th>
<th>Recommended Article</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer details for Alonso unveiled, Van Gaal sends Schweinsteiger on vacation, Mourinho fills the desire for Man United, Rangnick comments on Weinzierl, Boss turns away - Effenberg fights, Benzema: ”Treated as a terrorist”. This is how Darmstadt 98 trickles, Shitstorm for Hfl-Riesch. Press: ”Khedira has lost its strength”</td>
<td>That is why Guardiola is in a bad mood</td>
<td>Why Rode left Bayern?</td>
</tr>
<tr>
<td>One needs help - and all help: ”Suspects” arrested, ”The Chancellor has incapacitated the citizens”. Two women and baby attacked on open road, Suspect is obviously mentally ill, With cheap trick thousands of euros captured, Thilo Sarrazin has written a new book, ”I came to Berlin with a healthy heart and went as a wreck””, 14-year-old intensive care worker strikes at station, Man pulls women by the hair and strikes them</td>
<td>Syrian family attacked woman in refugee home</td>
<td>The big response of Frauke Petry</td>
</tr>
<tr>
<td>Dead is apparently ”Squeezer” singer Jim Reeves, Man harassed woman and injured police officer, Dancers were looking for the wrong victim, Even for Kreuzberg too crass, Men leave behind severely injured women, One needs help - and all help: ”Suspects” arrested, Two women and baby attacked on open road, With cheap trick thousands of euros captured, What’s behind Angela Merkel’s visit to George Clooney?, Thilo Sarrazin has written a new book, Angela Merkel has to change her lonely attitude</td>
<td>Merkel alone in Europe</td>
<td>Berlin police declares fate of ”Diensthund Sam”</td>
</tr>
</tbody>
</table>

Table 5.2 Some examples for which our model recommended the correct article on the first hit (HR@1).
Chapter 6

RARE: A Recurrent Attentive Recommendation Engine for News

6.1 Overview

In Chapter 5, we introduced a novel approach for recommendation which had the capability to adapt to the changing user preferences and identify the extent of the those preferences. However, we need to be able to identify both the specific and the generic preferences of the users in order to make more accurate and diverse predictions. Keeping the basic functionality of our previous method intact, in this chapter, we come with up a method which accounts for both the specific and generic interests of the user, and in turn is able to make diverse recommendations. In order to explain this in more detail, we present an example below.

As can be seen from Figure 6.1(A), if a user reads four different articles belonging to tennis and football, then we would like our model to infer that the generic interests of the users lie in reading articles about sports. Hence, this would allow articles belonging to different topics in the sports category to be recommended to the user. However, since the user reads more articles on tennis rather than football, we would like to give more weight to the articles related to tennis as can be seen in Figure 6.1(B). Hence, in our overall list of recommended articles to the user, we would like to present news article related to sports among which articles related to tennis would be given more importance. It may also happen that the user suddenly starts reading articles related to business rather than sports. In such a case, we may also want to start recommending articles related to business as well. This can be seen in Figure 6.1(C). However, it is important to note that in all these cases the sequential reading history of the user becomes important.

To tackle the above problem, we propose a novel neural network framework namely Recurrent Attentive Recommendation Engine (RARE). RARE consists of two components. The first component is based on a recurrent neural network and uses the sequential reading history of the user as its input. We call this the generic encoder. This helps us to identify the generic/overall interests of the users i.e provides a summary of the user’s interests. The second component utilizes a recurrent neural network with an attention mechanism to identify the specific interests of the user. We call this the specific encoder. The part dealing with attention allows the model to attend to articles in a differential manner,
Figure 6.1 In (A), the user's sequence is used to model her general interests. While in (B), the user's specific interests are captured. In (C), the changing interests of the user are modeled. In all these cases, the sequential reading history of a user plays an important role. Different colors represent the different topics of the article.

- Discriminating the more from the less important ones.

We then concatenate the representations obtained from both these components and call it the unified representation of the user's interests. Limiting the size of the user reading history used as inputs to both these components allows us to adapt to the changing user preferences. We then feed this unified representation along with the representation of the candidate article to a Siamese Network and compute an element-wise product between the outputs obtained at the final layer of the sister networks. Finally, we use a logistic unit to compute the score for recommendation. Using such a network enhances the model with further non-linearities and enables it to capture the user-article interaction in a better sense. It also allows the model to learn an arbitrary similarity function instead of traditional metrics. The distributed representation of each news article is used as inputs to our model. This gives the capability to recommend articles as and when they are produced, without depending on any prior user interaction with that article.

To summarize, the main contributions of this work are as follows:

- We present a neural network-based architecture (RARE) with the following capabilities:
  - It utilizes the content of the news articles giving it the ability to recommend articles as soon as they are published.
  - It takes into account the user's generic as well as specific interests.
It adapts to the changing interests of the user

- We carry out extensive experiments over three real world dataset to show the effectiveness of our model. The results reveal that our method outperforms the state-of-the-art.

- We also show the effectiveness of our model for solving the cold-start cases as well.

6.2 Model Architecture

In this section we first introduce our task and then provide an elaborate description of the various components of our model.

6.2.1 Task Description

Given a series of news articles read by the user, our task is to recommend articles of interest to the user. The implicit feedback provided by the user is available to us i.e we have information about the articles clicked by the user. Apart from this, we also have the content of the news articles available at our disposal.

We first select a reading history of size $R$ for each user. The value of the reading history determines the number of past interactions we use for making predictions. The articles previously read by a user can be represented as $[r_1, r_2, r_3, \ldots, r_t]$ where $1 \leq t \leq R$. Using this list as inputs to our model we need to recommend a ranked list of articles which are aligned with the users interests.
6.2.2 Overview

We propose a novel Recurrent Attentive Recommendation Engine (RARE) to address the problem of news recommendation for news aggregators. An overview of our method can be seen in Figure 6.2. The basic idea of RARE is to build a unified representation of a user’s interests which encapsulates both her specific and generic interests. Apart from this, using a specific amount of reading history of a user provides RARE with the flexibility to adapt to the changing interests of the user. The pipeline of RARE can be enumerated as follows:

- We first learn a distributed representation for each news article by combining its title and text.
- We then fix a reading history value $R$, and then use the representations of the previous $R$ articles read by the user as inputs to the model.
- We come up with a unified representation of the users interests using recurrent neural networks with an attention mechanism.
- Treating the unified representation of the user as a query and the representation of the candidate article as a document, we use a siamese network to make them undergo similar transformations and supercharge them with non-linearities to discover user-item interactions.
- We finally perform an element-wise product between the outputs of the siamese network and use it further to make predictions.

6.2.3 Distributed Representation for News Articles

We learn a 300-dimension distributed representation [21] for each news article by combining the title and text of the news articles. Learning such a representation allows us to

- Capture the overall semantics of the news article
- Enables the model to come up with a representation for new news articles as well as of articles with varying lengths

News articles generally follow an inverted pyramid structure where the title and the first paragraph give away the desired information. Hence, we only choose the title and the first paragraph because it usually contains all the relevant information without delving into detailed explanations. We also experimented by choosing the entire news article but found better results with just the first paragraph.

6.2.4 Generic Encoder

The inputs for the generic encoder are the representation of the articles previously read by the user. Figure 6.3(a) shows the graphical model of the network used to identify generic interests in RARE.
We use Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells. LSTMs have been shown to be capable of learning long-term dependencies [15][41]. The whole point in using this component is to understand the generic (broader/overall) interests of the user. The last hidden state of the RNN i.e $h_t$ encapsulates this information for us which we represent as $c^g$. We can think of the final hidden state as the overall summary of the users interests.

### 6.2.5 Specific Encoder

The architecture of this is similar to that of the previous one. The graphical representation for this can be seen in Figure 6.3(b). We use LSTM cells here as well. To capture the specific interests of the user i.e to understand the deeper interests of the user within her broader interests, we use an article level attention mechanism. This provides us with a context vector which encapsulates the specific interests of the user. This can be represented as,

$$c^s = \sum_{j=1}^{R} \alpha_j h_j$$  \hspace{1cm} (6.1)

where $\alpha_j$ determines the part of the input sequence which should be emphasized or ignored and $h_j$ stands for the output of the hidden units.

Using such a sort of mechanism gives RARE the capability to adaptively focus more on the important items.

### 6.2.6 RARE

The complete architecture of our model can be seen from Figure 6.4. The outputs obtained from the specific and the generic encoder are concatenated and then used as inputs to a siamese network alongwith the candidate article.
For the given task, the generic encoder captures the overall interests of the user i.e it captures the summary of the entire news articles read by the user. At the same time, the specific encoder adaptively selects the important articles to capture the specific interests of the user. Hence to take advantage of both kinds of information we concatenate the outputs of both the encoders.

As shown in Figure 6.4, we can see that $h^g_t$ is incorporated into $c_u$ to provide the summarized user interests. An important thing to notice is that, different encoding mechanisms will be invoked in both the encoders when trained jointly. The last hidden state of the generic encoder $h^g_t$ plays a different role from that of $h^s_t$. The former has the responsibility to encode the information present in the sequence in which the articles were read by the user. While the latter is used for computing attention weights. Information obtained from both the encoders is utilized to come up with a unified representation of users interests,

$$c^u = [c^g; c^s] = [h^g_t; \sum_{j=1}^{R} \alpha_j h^s_j]$$

(6.2)

where $c^u$ represents the unified representation of users interests.

We then use $c^u$ as inputs to one of the sister networks in the siamese network as shown in Figure 6.4. The input to the other sister network is the learnt representation of the candidate article. The siamese network supercharges RARE with further non-linearities and makes the user representation and the article representation go through similar transformations. In [17], an architecture similar to that of a siamese network has been used for ranking documents with respect to a query with great effectiveness. If we try to draw a parallel between the query-document problem with our task, one can see that a query in our case is $c_u$ and the document is the representation of the candidate news article. Hence, it seems apt to use such a network if we were to project both of these into the same geometric space to uncover the underlying user-article interaction pattern. A similar sort of technique has also been used by authors in [13] for modelling user-item interactions. We then perform an element-wise product between the outputs obtained from the sister networks of the siamese network. The element-wise product is used as inputs to a logistic unit for making predictions.
Once could also argue upon the usage of such a method for recommendation i.e instead of this, why not a typical encoder-decoder framework? If we look at the problem of machine translation, a typical encoder-decoder framework is unable to produce out-of-vocabulary (oov) words. In our case, each new published article, that has not been interacted by any user would act as an ”oov word”. However, it is very crucial for a news recommender to recommend articles as soon they are published which is why we resort to such a method as it allows us to handle such cases well.

6.2.7 Learning

Typically, to learn the model parameters, existing point-wise methods [33] perform regression with a squared loss. This is based on the assumption that observations are generated from a Gaussian distribution. However, in [13] it has been shown that such a method does not tally well when we have implicit data available to us.

Given a user $u$ and an article $x$, let $\hat{y}_{ux}$ represent the predicted score at the output layer. Training is performed by minimizing the point-wise loss between $\hat{y}_{ux}$ and its target value $y_{ux}$. Considering the one-class nature of implicit feedback, we can view the value of $y_{ux}$ as a label 1 meaning the item $x$ is relevant to a user $u$, and 0 otherwise. The prediction score $\hat{y}_{ux}$ then represents how likely an item $x$ is relevant to $u$. Hence in order to constrain the values between 0 and 1, we use the logistic function. We then define the likelihood function as follows.

$$p(\gamma^+, \gamma^- | I, \Theta_m) = \prod_{(u,i) \in \gamma^+} y_{ui} \prod_{(u,j) \in \gamma^-} (1 - y_{uj})$$

(6.3)

where $\gamma^+$ and $\gamma^-$ represent the positive (observed interactions) and negative (unobserved interactions) articles respectively. $I$ represents the input and $\Theta_m$ represents the parameters of the model. The negative log likelihood can then be written as follows (after rearranging terms).

$$L = -\sum_{u,i \in \gamma^+ \cup \gamma^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui})(1 - \log \hat{y}_{ui})$$

(6.4)

The loss is similar to binary cross-entropy and can be minimized using gradient descent methods.

It is also worth noticing that the likelihood function is such that it simultaneously adjusts the model’s parameters by maximizing the score of the relevant articles and at the same time adjusts to minimize the score of the non-relevant articles. This is similar to what is done while ranking documents corresponding to a query in [17]. Using such a likelihood also gives us the advantages of a raking function.

6.3 Experiments

In this section, we describe the datasets, the state-of-the-art methods, evaluation protocol along with the settings used for learning the parameters of the model.
6.3.1 Dataset

We use three real world datasets for evaluation. First, we use the dataset published by CLEF News-REEL 2017 [16]. CLEF shared a dataset which captures interactions between users and news stories. It includes interactions of eight different publishing sites in the month of February 2016. The recorded stream of events include 2 million notifications, 58 thousand item updates, and 168 million recommendation requests. It also includes information like the title and text of each news article. For this dataset we considered all the users who had read greater than 10 articles after which we get a total of 22229 users. The other two datasets are provided by Veooz.com. The second dataset contains a list of articles read by 10297 users in an Indian language, Malayalam. The third dataset contains a list of articles read by 22828 users in an Indonesian language, Bahasa.

6.3.2 Baselines

We compare our proposed approach with the following methods:

- **ItemPop.** News articles are ranked by their popularity judged by their number of interactions. This is a non-personalized method to benchmark the recommendation performance [30].

- **BPR** [30]. This method optimizes the matrix factorization method with a pairwise ranking loss, which is tailored to learn from implicit feedback. We report the best performance obtained by fixing and varying the learning rate.

- **eALS** [14]. This is a state-of-the-art matrix factorization method for item recommendation. It optimizes the squared loss (between actual item ratings and predicted ratings) and treats all unobserved interactions as negative instances and weighting them non-uniformly by item popularity.

- **NeuMF** [13]. This is a state-of-the-art neural matrix factorization model. It treats the problem of generating recommendation using implicit feedback as a binary classification problem. Consequently it uses the binary cross-entropy loss to optimize its model parameters.

Our method is based on user-item interactions, hence we mainly compare it with other user-item models. We leave out the comparison with other models like SLIM [26] and CDAE [42] because these are item-item models and hence performance difference may be caused by the user models for personalization.

6.3.3 Evaluation Protocol

To evaluate the performance of the recommended item we use the leave-one-out evaluation strategy which has been widely adopted in literature [2][14][30]. For each user we held-out her latest interaction as the test set and utilized the remaining data for training. Since it is time consuming to rank all items for every user during evaluation, we followed the common strategy [8][20] that randomly samples 100
items that are not interacted by the user, ranking the test item among the 100 items. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG) [12]. We truncated the rank list at 10 for both metrics. As such, the HR@k intuitively measures whether the test item is present in the top-k list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score.

### 6.3.4 Parameter Learning

We use an Intel i7-6700 CPU @ 3.40GHz which has a RAM of 32GB and a Tesla K40c GPU. We implemented our proposed method using Keras [6]. We randomly divide the labeled set into training and validation set in a 4:1 ratio. We tuned the hyper-parameters of our model using the validation set. The proposed model and all its variants are learned by optimizing the log loss of Eq. 6.10. We initialize the fully connected network weights with the uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ [10]. We used a batch size of 256 and used adadelta [43] as the optimizer.

**Figure 6.5** HR of RARE vs state-of-the-art on CLEF News- REEL

<table>
<thead>
<tr>
<th>Layers</th>
<th>HR@10</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0.913</td>
<td>0.659</td>
</tr>
<tr>
<td>128-&gt;64</td>
<td>0.934</td>
<td>0.671</td>
</tr>
<tr>
<td>128-&gt;64-&gt;32</td>
<td>0.912</td>
<td>0.666</td>
</tr>
</tbody>
</table>

**Table 6.1** Performance of RARE by changing number of dense layers
Figure 6.6 NDCG of RARE vs state-of-the-art on CLEF News-REEL

Figure 6.7 HR of RARE vs state-of-the-art on Malayalam Dataset

Table 6.2 Performance using different Encoding Mechanism on CLEF NewsREEL

<table>
<thead>
<tr>
<th>Method</th>
<th>HR@10</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific Encoder</td>
<td>0.916</td>
<td>0.657</td>
</tr>
<tr>
<td>Generic Encoder</td>
<td>0.920</td>
<td>0.664</td>
</tr>
<tr>
<td>Specific + Generic (RARE)</td>
<td><strong>0.934</strong></td>
<td><strong>0.671</strong></td>
</tr>
</tbody>
</table>
Figure 6.8 HR of RARE vs state-of-the-art on Malayalam Dataset

Figure 6.9 HR of RARE vs state-of-the-art on Bahasa Dataset
Figure 6.10 HR of RARE vs state-of-the-art on Bahasa Dataset

Figure 6.11 HR of RARE w.r.t. the Users Reading History on NewsREEL. A similar trend was observed for the other datasets.
Figure 6.12 NDCG of RARE w.r.t. the Users Reading History on NewsREEL. A similar trend was observed for the other datasets.

Figure 6.13 HR of our RARE w.r.t recurrent unit used in RARE on NewsREEL. A similar trend was observed for the other datasets.
Figure 6.14 NDCG of our RARE w.r.t recurrent unit used in RARE on NewsREEL. A similar trend was observed for the other datasets.

Figure 6.15 HR of RARE on cold-start cases
Figure 6.16 NDCG of RARE on cold-start cases

Figure 6.17 HR of RARE on negative sampling
6.4 Results and Discussion

In this section we present the results obtained by carrying different experiments with our method.

**Performance Comparison with Baselines** : For MF based methods like BPR and eALS, the number of predictive factors chosen is equal to the number of latent factors. We report the best performance in this case. For NeuMF, we vary the size of the CF layers (also latent factors) to choose the best fit for our model. In Figure 6-11, the performance of ItemPop measure was very weak and hence it is not clear in the graphs.

In Figures 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, we compare our method with the baselines. The Top-K recommended lists are used where K varies from 1 to 10. It is very clear from Figures 6-9 that RARE outperforms the other by a significant margin across all positions on the NewsREEL and the Malayalam datasets. Although, RARE outperforms the others in case of Bahasa dataset as well (Figures 10-11) but the margin is not that large. Amongst the different baselines, the trend in the performance can be seen as follows : NeuMF >eALS >BPR (in terms of both HR and NDCG). Although, in [30] it has been shown that BPR can be a strong performer for ranking performance owing to its pairwise ranking aware learner, but for our datasets it does not seem to be so. On the other hand RARE outperforms all the other baselines in terms of NDCG as well.

Overall, our method consistently outperforms the state of the art, which suggests that modelling user sequence behaviour to understand their preferences is very crucial to the task of news recommenda-
tion. It may not be futile to conclude that by taking both the users generic preference and the specific preferences, RARE is able to make better recommendations.

**Effect of Size of Reading History** : We also vary the size of the reading history $R$ used as inputs to our model. From Figures 6.11, 6.12, one can see that the Hit Ratio slowly increases with the size of the reading history until a certain point after which it decreases. However, the NDCG keeps on increasing. We can attribute this behaviour to the fact that, users have diversified reading interests which only get effectively captured after a substantial amount of interactions have been observed. However, after a while, increasing the user history often leads to over-specialization where the generic interests tend to overpower the specific ones. This is also an indicator of the fact that the preference of a user keeps varying and hence a window size should be chosen such that it helps the model to dynamically adapt to the users changing behaviour.

For all our methods, we chose a reading history of 12 for the users. We needed to make a choice between 12 and 14 and we choose 12 because we gave more importance to the HR rather than the NDCG. We argued that as our system recommends top-10 articles at once, the actual item that a user might pick should be amongst those 10 articles. Hence we gave more importance to the HR metric rather than the NDCG and chose a value of 12.

**Effect of different Encoders** : We first note the effects on RARE by varying the kind of recurrent network used. We tested our model by using LSTMs, GRUs (Gated Recurrent Units) [7] and Vanilla RNN. From Figure 6.13, 6.14, the trend in the performance can be observed as follows: LSTM $>$ GRU $>$ RNN although the differences are not very large. One of the reasons for this could be the fact that an LSTM or a GRU is better able to encode the interests of the user as they handle long term dependencies better.

We also note the effects when using the different variants of our own model, i.e when we replace the unified representation in RARE with solely the specific or the generic encoder. The results for this can be seen from Table 6.2. We note the trend in performance as follows RARE $>$ Generic Encoder $>$ Specific Encoder. This indicates that merely identifying the users generic interests (a summary of overall interests) is not sufficient for learning a good recommendation model. However, when we use a combination of both in RARE, we find that the recommendation performance improves which clearly indicates that identifying both the specific and generic interests are essential for better recommendations.

**Performance on Cold Start Cases** : We then evaluated our model for the cold start cases as can be seen in Figures 6.15, 6.16. For this task we segregated users who had read a new news article in the end i.e they read articles which had never been seen before they read it. We found out that the number of such users were 74 in CLEF dataset. The other two datasets that we had did not have sufficient amount of such users to test on. Out of these 74 users, we see that the HR@10 is around 0.35. This promises us that our model is well suitable for handling the item cold-start problem.

For user cold-start, we test our learned model over users who had read articles in between 2 to 4 (inclusive) over the same dataset. Since we set the history size to 12, we had to set the remaining inputs to 0s. The HR@10 score was around 0.5. We see a gradual increase in the hit rates as we increase
the value of k. The results promise the effectiveness of our model to handle the problem of user cold start as well. Although, this is not exactly the user cold start problem because it still considers some amount of user interactions. But it is still worth noticing the performance because the baselines need a considerable amount of user history before making predictions. While in our case, we can simply use the trained model for recommending articles to users who have had very few interactions. Over time, with sufficient data about these users, the recommendations would get better.

**Effect of Varying Layers**: We observe the performance of our model when we vary the number of layers used in the Siamese Network in our model. We experiment by varying the number of layers along with the number of hidden units. We experiment by using one layer with size 128, two layers with sizes 128 and 64 and three layers with sizes 128, 64 and 32. From Table 6.1, we can see that the best performance is observed in the second case. If we think of the number of hidden units as the number of discriminative factors which tell us how similar the unified representation of a user and the candidate articles are before making predictions, then we could say that a larger number of hidden units are not suitable for discriminating and at the same time a fewer number of hidden units are not suitable as well.

**Effect of Negative Sampling**: The results can be seen from Figures 6.17, 6.18. Since we only have information about the articles read by the user, in order to pick negative samples we randomly choose articles that the user had not interacted with (read previously). We pick \( N \) negative instances per each positive instance. We varied \( N \) from 1 to 4 and noted the results. We found that 2 was best for our case. We observed that choosing larger number of negative instances per positive ones causes lesser generalization and performance deteriorates. While randomly generating negative samples it might be possible than an article of interest to a particular user is chosen. In the future, we would like to explore more on sampling strategies which avoid this issue.

### 6.5 Summary

We have proposed the Recurrent Attentive News Recommendation Engine to address the problem of news recommendation. We try to understand both the generic and the specific interests of the users. For the former we use a recurrent neural network while for the latter we use a recurrent network with an attention mechanism. We use the unified representations obtained from both these along with a siamese network to make predictions. We conducted extensive experiments on three real-world datasets and demonstrated that our method can outperform the state-of-the-art methods in terms of different evaluations metrics.
Chapter 7

Word Semantics based 3-D Convolutional Neural Networks for News Recommendation

7.1 Overview

In the previous chapters, we have relied on Recurrent Networks for identifying the varying preferences of the user. However, we also wished to observe the effects of a non-recurrent deep learning model when applied to the similar problem, with a similar motive i.e to understand the preferences of the user over time and make recommendations.

In this chapter, we experiment with a recommendation model which uses semantic similarity between words as input to a 3-D Convolutional Neural Network in order to extract the reading preferences of the users. 3D CNNs [19] have successfully been used for the task of action recognition where capturing spatial and temporal information is very important. In our case, applying 3D convolution helps us to identify both the spatial information (features of a particular article) as well as the temporal information (features present in the sequence of articles read by the user) which are pertinent to a user’s interest without delving into the task of manual feature engineering.

7.2 Model Architecture

In this section we briefly provide the description of our model. We divide the model into three parts, 3-D Tensor, 3-D Convolutions and Score Aggregation. We then explain the training criteria for the model.

7.2.1 3-D Tensor

Similarity Tensor is a three-dimensional structure (as seen in Figure 7.1) where each element $M_{ijk}$, denotes the similarity between the $j$-th word of the $i$-th read article (of the user reading history) denoted by $w_{ij}$ and $k$-th word of the article which is to be considered for recommendation (i.e test article)
denoted by $v_k$:

$$M_{ijk} = w_{ij} \otimes v_k \quad (7.1)$$

where $\otimes$ stands for a general operation to obtain the similarity. We then define the general operation between $w_{ij}$ and $v_k$ to be the cosine similarity of their respective word embeddings.

We use Word2Vec[24] in order to learn word embeddings. We concatenate the title and text of the news article. We then compute the similarity tensor by finding out the similarity between each word of the article in the user history with that of the item that is to be considered for recommendation. One can think of the similarity tensor as a stacked representation of 2-D matrices, where each stack (2-D matrix) is a similarity matrix between an article in the user history and the test item. For example: suppose we choose the reading history for each user to be 4, then $M_{234}$ would represent the similarity between the word embeddings of the 3rd word in the 2nd article of the user history and the 4th word in the article that is to be considered for recommendation.

### 7.2.2 3-D Convolution

Based on the Similarity Tensor, we then conduct 3-D convolution to extract features that would depict the changing/evolving interests of the users. The model consists of 3-D convolution layers and pooling layers. Kernel size in each convolutional layer are the major hyper parameters. In text processing, the size of the kernel determines the number of words we want to compose together as well as the extent of temporality we would like to consider. Besides, pooling sizes in each pooling layer are also important which decide how large area we want to take as a unit.

Formally in a 3D CNN, the value at position $(x, y, z)$ on the $j^{th}$ feature map of the $i^{th}$ layer is given by:

$$v_{ij}^{xyz} = \tanh(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijmn}^{(x+p)(y+q)(z+r)}) \quad (7.2)$$

where, where $R_i$ is the size of the 3D kernel along the temporal dimension, $w_{ijmn}^{pqr}$ is the $(p, q, r)^{th}$ value of the kernel connected to the $m^{th}$ feature map in the previous layer.
7.2.3 Score Aggregation

After a series of convolutional layers followed by pooling layers, three additional fully connected layers are used to aggregate the information into a single matching score/target value. In this paper we use 128 hidden units for the first hidden layer followed by 64 units for the second hidden layer. We use ReLU as the activation function for these layers. For the final output unit we use the Logistic Function.

7.3 Experiments

As mentioned earlier we use the data provided by CLEF NewsReel 2017. We choose users who have read in between 10-15 (inclusive) articles for training and testing our model for item recommendation. The frequency of users who have read more than 15 articles varies extensively and hence we restrict ourselves to the upper bound of 15. We set the lower bound to 8 since we need some history in order to capture the changing user interests. However, for future work we would like to investigate how changing the lower bound affects the performance of our model.

Evaluation Protocol: For each user we held-out her latest interaction as the test set and utilized the remaining data for training. We then recommend a ranked list of articles to each user. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative gain (NDCG). Without special mention we truncate the ranked list at 10 for both metrics.
Figure 7.3 NDCG of 3D CNN vs some baselines

**Baselines:** We compare our method with several others. First we look at item popularity based method (ItemPop). In this we recommend the most popular items to the user. We then evaluate User-to-User (U2U-KNN) and Item-to-Item (I2I-KNN) by setting the neighbourhood size to 80. We then compare it with Singular Value Decomposition (SVD). We also implement Word Embeddings based Recommendations as in [25] and Keyword based Vector Space Model (Key-VSM) as mentioned in [32].

**Parameter Settings:** We implemented our proposed model using Keras [6]. We then construct our training set as follows:

1. We first define the reading history. We denote the reading history by $h$.
2. Leaving the latest article read by each user, the remaining articles are used as positive samples.
3. Corresponding to each positive sample, we randomly sample 4 negative instances (articles which the user did not read).

We then randomly divide the training set into training and validation set in a 4:1 ratio. This helps us to ensure that the two sets do not overlap. We tuned the hyper-parameters of our model using the validation set. We use a batch size of 256. In the first layer of the model we apply a 3-D Convolution of size 3x3x3, followed by a max pooling layer with a pooling size of 2x2x2. We then repeat 3-D convolution with the same kernel size followed by a pooling layer with pooling size of 1x2x2.
We experimented with different variations in kernel and pooling sizes. The above mentioned seemed to be performing the best.

### 7.4 Performance Comparison

From Figures 7.2, 7.3 it can be clearly seen that 3-D CNN outperforms the respective baselines in terms of the Hit Ratio. We also see that the NDCG scores of 3-D CNN is high as well. Further it can be clearly noticed that U2U, I2I and SVD do not perform well. One reason for this could be the sparsity of the data. In presence of sparse data these methods fail to capture relevant information. The low performance of Word Embedding based Recommendations suggests that a representation of words alone is not effective in profiling the user. The model also outperforms Key-VSM [32]. Key-VSM is fairly able to capture the users interests but are still not at par with 3-D CNN. This shows that in tasks such as news recommendation, where the user interests keep varying, 3-D CNN is better in accounting for the temporal changes.

### 7.5 Summary

In this work we explore the idea of using semantic similarity in combination with 3-D CNN for capturing users preferences in order to recommend news articles.
8.1 Conclusion

In this thesis, we present a series of approaches for providing personalized news recommendation. We progress in a step by step manner to come up with a robust news recommendation system with the desired capabilities.

In Chapter 3, we come up with a method for providing decent recommendations when presented with a moderate amount of user reading data. Most state-of-the-art methods rely on a considerable amount of user reading history before making good predictions. Another class of deep learning models fail to learn the model parameters when the data is insufficient. News aggregators which are new to the market cannot generate a lot of data for user interaction. However, such websites would want their existing userbase to remain intact and, in order to do so, would deploy a recommender system. In that case, it would be desirable to have a recommender system which performs decently even on a moderate amount of data. With this in mind, we came up with an MDP based approach for news recommendation. Here we show that by using the sequential nature of news reading and aspects like semantic similarity and freshness, we are able to fairly generalize and provide better recommendations. Also, posing the problem of recommendation as one of sequential optimization provides us with better results. However, such a method had its own disadvantages. Compared to the state-of-the-art methods which perform well when presented with a huge amount of user interaction data, the MDP based approach did not match up. We identified a few reasons for this. Firstly, the increasing amount of read news articles creates the need to have an increasing amount of states in
an MDP. Secondly, the semantic similarity measure tends to get biased over time. Third, such a method does not have the capability to handle the cold start problem, which is crucial for news recommendation. With this, we started to explore other methods which can generalize well when presented with a large amount of data and also can work well for users who have had little interaction with news articles.

In Chapter 4, we try our first approach on coming up with a system which would perform well even on a large amount of data. With our previous experiments, we understood that utilizing the sequential reading history of a user was very important. Apart from this it was also important to figure out the role of individual articles in understanding user preferences. Primarily, we wanted to understand the effect of a user’s latest read articles on her interests. This could also be thought of as identifying the changes in user’s interests. In order to understand this transient interest, we first came up with a user profile and experimented with various heuristics. We then used it as inputs to a DSSM. The user profile was generated by utilizing the content embeddings of the news articles. The parameters for the entire model were learnt by utilizing implicit feedback of users. Experiments demonstrated that our method was able to make good predictions. We could also use a similar model for recommending articles which had a very small reading history. Utilizing the content of the article gave the model an advantage in working in cold start scenarios as well. However, with such a method, we were still relying on heuristic based user profiling methods. It was not possible for the model to discriminate between the various interests of the user. Since our method relied on the sequential reading history of the user, in our further approaches we started experimenting with recurrent neural networks for user profiling.

In Chapter 5, we explore user recurrent neural networks for recommending articles. The building blocks of the model remain the same i.e. utilizing the content of the news articles, utilizing the sequential reading history of the user and learning the model parameters based on the implicit feedback of the user. There were two main differences in this approach. Instead of heuristics based profiling, we used recurrent neural networks. This helped us get an overall summary of the users interests. However, apart from identifying the various interests of the user, our aim was also to understand the extent of the those interests. For this purpose, we employed an attention-mechanism in combination with the recurrent neural network. We
observed that using such a method improves the overall performance of the recommendation system. Apart from this, since this model was based on a similar principle as the previous one, we were also able to address the cold start problems effectively. However, even though we did identify the various interests of the user, we observed that using such a method biased the recommender towards the specific interests of the user. The diversity in the recommendations was lost in this case. In order to make our system more robust, we then proposed our final solution.

In Chapter 6, we have proposed the Recurrent Attentive News Recommendation Engine (RARE) to address the problem of news recommendation. RARE consisted of two components and utilized the distributed representations of news articles. The first component was used to model the user’s sequential behaviour of news reading in order to understand her general interests i.e to get a summary of her interests. The second component utilized an article level attention mechanism to understand her specific interests. We fed the information obtained from both the components to a Siamese Network in order to make predictions which pertained to the user’s generic as well as specific interests. Various experiments over three real-world datasets demonstrated that our method outperformed the state-of-the-art. Apart from this, we were also able to address the cold start problem.

In our few previous approaches, we experimented with recurrent neural network. In Chapter 7, we experimented with non-recurrent networks with an objective similar to our previous one i.e to be able to understand the different interests of the user which might keep changing with time. We experimented with a recommendation model which used semantic similarity between words as input to a 3-D Convolutional Neural Network in order to extract the reading preferences of the users. Applying 3D convolution helped us to identify both the spatial information (features of a particular article) as well as the temporal information (features present in the sequence of articles read by the user) which are pertinent to a user’s interest without delving into the task of manual feature engineering. Experiments showed that our method was better than the content based methods, but was not at par with the collaborative filtering approaches. However, this was a positive sign as we understood that using 3D CNNs could be effective if utilized in some other setting. We try to give a brief description of this setting in our future work.
8.2 Future Work

To carry forward and augment the work presented in this thesis, one of the primary options, given the success of using implicit feedback, would be to incorporate more user dependent features. These include demographics, age, and sex, to assimilate more information about readers and their related reading patterns.

We have observed that using item/article attention mechanisms to capture generic and specific user interests have yielded improved results. There is scope to integrate word-level attention mechanisms in order to help the model further discriminate between the user’s interests.

During the model’s training phase, we used a negative sampling strategy for training the model parameters. However, other sampling techniques might help improve the performance of the model further.

There are various signals encoded in implicit feedback. An instance of the same can be taken as the user clicking on a link and going to a page, but returning back quickly. This indicates the article’s irrelevance to the user and can be leveraged to the recommender system’s advantage.

Considering the utilities of the 3D CNN approach as detailed in the thesis, work on the same can be expanded, setting up a 3D CNN based on a two component approach. The first component of the setting would, in turn, be based on a 3D CNN which takes as input the word embeddings of the articles from the user’s reading history. The second component would be based on a 2D CNN and take as input the word embeddings of the test article. Such a setting is very similar to the architecture that has been used in RARE.
Related Publications


Bibliography


