STREAMLINED NEWS RECOMMENDATION SYSTEM USING A VARIABLE MARKOV MODEL

by

Dejan Spanovic

B.Sc. Ryerson University, Toronto (ON), Canada, 2016

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2019

Author's Declaration

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

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

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

I understand that my thesis may be made electronically available to the public for the purpose of scholarly research only.

Abstract

STREAMLINED NEWS RECOMMENDATION SYSTEM USING A VARIABLE MARKOV MODEL

Dejan Spanovic
Master of Science, Computer Science
Ryerson University, 2019

Providing news to users in a news article recommendation system is a balancing act between delivering news that is recent and news that is relevant to their interests. Users should be able to receive a stream of similar articles that interest them and control their traversal through the topics of news articles in a stream-wise fashion as well. A Variable Markov Model (VMM), built on trends in recently published news articles, is proposed as a single solution to categorically cater news to all users with minimal overhead and maintenance. This single model provided to all users throughout experimentation has shown that, though it is not built based on user interests, it is applicable as a basis for applying user interest and trend factors upon to achieve catered and novel news recommendation experiences.

Acknowledgments

I would like to express my gratitude to Dr. Cherie Ding, for the opportunity to continue my education, providing opportunities to learn more in the field, and guiding me through the process of both developing the thesis project and the writing of this thesis. She had greatly eased the process of understanding of development of a thesis and steered me to the completion of both the preceding project and this thesis. This includes the project itself started by Marj systems and further funding brought into it.

I would like to thank my parents for being a motivational factor, going above and beyond for the website-hosting of the application locally until I decided to stop bothering them with it, letting me know what steps I can take in writing my thesis, and providing desserts while I worked.

My brother and friends for providing their feedback whether it be testing before testing, ideas on how to approach my analysis, and further motivation. Finally, I would like to thank Obi for being the closest thing to a duck I could use for rubber duck debugging.

Table of Contents

List of Tables

List of Algorithms

List of Figures

List of Appendices

Chapter 1

Introduction

1.1 Background

News is a type of media for providing information of events so that readers are aware of the world around them. In paper format, these articles are usually organized in sections, so that users can easily navigate to subjects that may contain information that interests them. Since the creation of the online news media, it has become easier for people to gain access to information they would like to access from various global news sources, rather than just the local paper, and navigate to their subjects of interest through links. The accessibility of interests is expanded through more specific organization by using key-worded topics. Worldwide news organizations such as New York Times, CNN, and Reuters are examples of organizations around the world that provide online news websites that contain organized news content from different perspectives to their readers. These sites provide an opportunity for aggregation of news sources that can expose people to more points of views on common stories as well as stories they otherwise may not come across from a single source.

A prominent example of a news aggregator site is Google news. It is not the traditional kind of news website in that it does not provide its own content, but rather it provides content from a collection of external sources. Google compiles news from thousands of news organizations around the world and presents them to the user under the categories of those sources. The various levels of categorical identification of articles are also presented from these sources. Categories range from high-levels like “world news”, “science”, and “business”, to more specific topics such as “Donald Trump”, “Facebook”, and include the same stories told from multiple sources. This organized aggregation provides a casual user the capability to quickly identify what categories may be of interest to them and narrow down more specific topics of interest through the organized aggregation of the articles. Furthermore, if a user were to have a Google account, they can also be shown news that is catered to their interests as a list of recent articles based on what they have previously read. In short, news aggregators, like Google News, provide a single personalized source of targeted news to prevent information overload created from the many possible sites that users could get their news from .

The convenience of these aggregated news recommender systems is that they help mitigate any issues with the user conveniently finding articles that are of interest to them. This convenience extends to mobile platforms, so that users may be provided news recommendations on the go. A mobile platform expands the number of contexts that a user may access from a recommendation engine. These contexts include factors like location and a larger window of access time associated with mobile platforms as opposed to a static desktop site. Thus, mobile systems are key ways of exposing users to news and acquiring information that can be used to further cater to the user experience.

Through ingesting information based on what the user has read and shown interest in, along with the context which the user has read the news under, a news recommender can take the aggregated articles at its disposal and provide the user with news that would appeal to them. When placed on a mobile platform that is more frequently accessed, the user can quickly establish themselves a profile for their viewing tendencies as they access the application through the application’s information gathering techniques. There are numerous ways to provide new information to a user that are not limited to what they have shown interest in. The news recommender can also start recommending articles they may have not thought to be interested in through the inclusion of other similar user's viewing behaviors, adjusted recommendation algorithms, and other factors that can bring in a level of serendipity to the results. The news a user reads not only becomes catered to their interests, but they are also now being provided an extended perspective of those interests and a set of potential interests. This is all possible through the aggregation of multiple viewpoints and adjustment of the processing of user data to provide a fresh experience and perspective to the readers.

In these catered aggregators there are two generalizations of information available for interpretation: content-based and user-based information. Within content-based information, news articles can be linked together based on the attributes that define them and allow for aggregators to categorically sort items so that they are more easily identifiable by a reader. User-based information is used to link users to the articles that may interest them, as well as produce links to other users and their reading history and habits. Based on the history of each individual user, aggregation systems can have the ability to link the categories of the user's past reading tendencies to an assortment of articles that may be of interest to them. Furthermore, these users can be linked to one another to categorically sort similar reading tendencies and potentially introduce new interests to a user as well. How this information is inferred or gathered depends on the system. There are many methods at various hierarchies of specification to consider for identifying the features of articles and users in order to create the needed links from: articles to articles, users to other users, and users to articles. In this thesis, we will explore creating a system that utilizes these links to deliver a catered user experience. Within the environment that the system is completed in, the methods of information gathering, processing, and delivering news articles to the user will be explored. This thesis will then deliver a method of creating an aggregator within the constraints provided.

1.2 Problem Statement

There is a broad definition for a successful recommendation engine. Like search engines, there are items being retrieved and provided to a user. Measurements like precision and recall can be used as a direct reflection of what is presented to a user from a set of data. However, recommendation engines do not focus on optimizing these measurements as they do not directly reflect what is optimal for user satisfaction in this case. Factors like spreading out the hit rate of documents in the database, serendipitous results, diversity, and novelty should be considered as well. It is through reviewing metrics beyond accuracy that the user experience remains interesting and fresh, rather than predictable and repeated. Real-time experiments and evaluation of results will further confirm or deny whether the user satisfaction with the proposed experience reflects numerical optimization on these kinds of factors. Deciding the appropriate optimization of these metrics will determine the effectiveness of the selected methodology along with user results.

News acquisition and classification is real-time process. To provide fresh results to a user, up-to-date articles must be aggregated. Quick and efficient techniques of classifying these articles and relating them to one another must be established to maintain consistency with both real-time news and what are considered to be linkages between articles. From here, strategies for grouping news in real time must be considered to allow for relevant results to be returned to the user without compromising real-time speed. With respect to many recommendation methods, this would not be a factor as many tend to use a ranking system like a search engine to find and return a list of articles to the user’s interface. However, in this thesis, the constraints of the proposed system limit articles to be under the same topic during recommendation, rather than a list of likely unrelated articles catered to the user on a mix of factors such as their read history and popular topics. Because of this, pre-emptive clustering methods for articles must also be considered. This adds an extra layer in what should otherwise be a real-time system of recommendation.

A real-time system also means that user interests must be updated immediately from interactions to meet the need for immediate returns in feedback from these system interactions. There are many methods for incorporating user information in a recommender system. Methods typically involve updating the user profiles on current interactions upon past user information to predict their current interests. More in-depth collaborative methods that look to cluster user information to determine their potential interests in articles and topics must also be considered. Ultimately, the system should combine real-time content classification and user classification to return uniquely catered results. Evidently with both content and user data, creating usable profiles to accurately return the desired output will have to meet some sort of compromise in real-time processing. As both user and article information will be available, a hybrid system could be made that can accurately reflect using both user and article information. There are many modern methods being developed to model articles and users. These methods will be explored in the Literature Review Chapter. The goal for the selected method should be for the result of the selected strategy to meet the desired metrics of evaluation with minimal compromise to real-time performance.

The restrictions the system is under are also important in understanding the decisions made in the selection of methods. The system must fall within a mobile-recommender system context. The interface must be easy to understand and effective and providing recommendations. In this system the user receives their recommendations sequentially, one at a time. Each article must fall under the same category as the previous one, unless the user decides they want to change the category.

This is unlike a traditional aggregator where a user will receive multiple suggestions on the same page where everything is categorically sorted. The next article must be influenced by real-time interactions. Whether a new article was added to the aggregator database or the user shows interest in the current article through an interaction, the next articles must reflect the changes in the system. Furthermore, if the user changes the topic of the article, it must go to a completely different topic. This means there must be a way to cluster similar articles under certain criteria before recommendation, so that these real-time recommendations are not held up. Finally, the user has no explicit way of stating their preferences. An example of explicitly stating preferences is how the Flipboard News Aggregator application requires a user to state whether they like topics like “science” or “sports” before going into the application. This means that all information regarding user preferences must be done implicitly to narrow down their interests and updated over time.

1.3 Objectives

In circumventing the problems as well as the restrictions in the presented system, there is a number of objectives to be completed to establish a working recommendation pipeline. The primary objective is to provide a seamlessly streamed aggregation of articles to the user. In addition to this, up-to-date profile information must be made available to the system to allow for relevancy in the next step in the stream with respect to previously retrieved information in the recommender stream. This is depending on whether the user decides to change the topic or go to the next item under the current topic, as well as their interactions with those items. Relevancy to the previous items will be lightly explored, as the focus is not on the details, but rather creating a functional system with satisfactory results. For this, this thesis will look into multiple simple, yet effective, strategies to combine in the ranking scheme with respect to both content and user information.

Encompassing the ranking scheme, and another objective in this thesis, is to create a method for clustering similar items for both faster processing and allowing for consistent relevant information appearing in the stream. One efficient clustering method, that will remain effective as new information appears and does not strain resources when users request new information, will be defined. Article topics must also be effectively and evidently changed as a user traverses the available topics. For this, methods to avoid categorical repetition and straining of user interest in the application will also need to be defined in tandem with the user profiling methods that are typically used to bring in similar information.

The sum of these parts is as follows. First is the ranking system based on the content and user profile information. Second is the clustering method required to organize information in real time. Once these two main components are completed, a system within the constraints can be defined. Finally, methods of evaluation should be defined to evaluate both the system's efficiencies in the forms of accuracy and other metrics, as well as user feedback gathering and evaluations of that feedback.

1.4 Proposed Methodology

There are many modern aggregation systems and many different approaches exist for creating a new aggregator that provides recommendations. The constraints within the proposed system are a factor that limits the complete duplication of algorithms that are otherwise used by other platforms and makes it unique. Because of the constraints, an adapted approach of other methodologies to the desired system will be applied and evaluated. The goal of this thesis will be to apply the knowledge of possible methodologies within the system's constraints and create a working system.

The news articles in this system are extracted from a variety of sources. As the scraped websites have varying ways of organizing their contents that cannot be uniformly identified, the proposed system must have its own method of content classification. Thus, the contents of the articles will be processed through established text evaluation methods. Once content is organizable from the results of this evaluation, the clustering method will be applied to tag articles under certain topics and link them to one another in recommendation.

During the recommendation phase, the system will build on top of the selected clustering method. Here both content-based and collaborative-filtering methods will be applied to rank potential articles of interest to a user as well as possible categories that would be of interest to them. With respect to user information for influencing recommendations, past reading history will be considered in the form of interests shown in specific topics and similar articles through user interaction history and user profiles. From this, both long-term and short-term profiling methods will be applied in influencing the next recommendation.

Furthermore, other methods of identifying possible interests along the lines of context will be considered. This may reflect context in the form of time, place, or other external factors like trending information. The extraction of these pieces of information will be observed and considered. Ultimately, the application must be quick and effective at providing results to a user. Due to this constraint, the thesis will mostly cover simpler and quicker methods of applying such information.

1.5 Outline

The following chapters will provide an overview of concepts and approaches used in the research. The Literature Review Chapter that will cover samples of literature both indirectly and directly related to the methodologies is used to provide understanding of the reasoning behind the strategies implemented. The Methodology Chapter will follow this and discuss the methods used in the system and how they were implemented. The Experiment Chapter will go over the parameters of the system, how the system was tested, and discuss the results of the experiment. The Conclusion Chapter goes over what can be understood from the experiment, the evaluations, and what can be derived from it in future work.

Chapter 2

Literature Review

2.1 Introduction

This Chapter will cover the basis for understanding the methods used for building recommender systems with a focus on news recommenders. It is through establishing this portion of the thesis that the constraints can then be applied to existing algorithms to create the proposed system. Optimization of how information is manipulated and interacted, the concepts that guide those methodologies and real examples of recommendation systems, the retention of user interaction and understanding, and established ways of evaluating news systems will be observed. This review will begin with coverage of what researchers have done in the field of recommendation systems and move on to focus on specifically news recommendation systems, so that a general understanding can be established of how these systems work and why certain approaches were taken for them. What should be kept in mind is that the ultimate task of all news recommender systems is to estimate the ranks of news articles that the user has not seen before .

2.2 Recommender Systems Overview

A brief overview of recommender systems must be made before observing the advancement of research on them. An understanding of the three approaches to creating recommendations should be understood. Content-based, Collaborative, and Hybrid recommendation systems are the three generalized types of systems used to provide recommendation services. With an understanding of these systems, examples of them can be understood and the shift of general methods to new-specific methods can be made.

2.2.1 Content-Based Recommender Systems

Content-based services are defined as recommender systems providing results to a user based on items that are similar to ones that the user has interacted with . Items that a user had interacted with will be evaluated based on their traits, like their classifications or their contextual information, and be compared to other items within a set. The compared items will then be chosen based on their attributes and how the chosen recommendation method handles them. In written articles, these traits can include the most important keywords that are derived, through text processing techniques, that will be used to represent the article . Furthermore, user profiles can be derived from methods that utilize their reading histories. For example, a user can be defined by the averages of the numerical values representing the traits of their accessed content, in the form of content-vectors. These can then be used to compare the user's access preferences to items. This, however; has its shortfalls due to overspecialization. This can result in recommendations that are strictly based on what the user has interacted with and result in a lack of diversity in results. The new user problem is another issue that arises from such a system . A goal of the proposed system will be to minimize the effects of such issues.

2.2.2 Collaborative-Based Recommender Systems

Collaborative methods revolve around finding other users with similar preferences or access patterns and basing the results from what items they have interacted with . Unlike in content-based services, this does not look at the traits of articles, but users. Based on items a user has interacted with and the representation of these actions in their profile, other users will be found who have similar tastes in the same items. New recommendations will then be based on these new items that the user has not yet interacted with. The first recommender system, the Grundy System, is a good example of this. The Grundy System was based on building stereotypes of users as a cluster and recommending books to similar people within that cluster. This was assorting people based on their personality traits into a structure that stems from the traits that define the average person. This categorization allows for a frame of reference to a default value and provides value in the form of now being able to identify what specific kind of category the subject of interest falls under. These collaborative methods can be further moved into being categorized as knowledge-based or model-based. This is creating recommendations based on collections of interacted items by users or using the collection of interactions to create a model to processes future information. Like the content-based systems, collaborative methods suffer from issues like the new user problem, along with issues regarding new items being recommended, and sparsity issues when dealing with predicting a rating for items that are rarely interacted with. These issues can be mitigated through the implementation of both user attributes and contextual user information and other methodologies, which will also be explored.

2.2.3 Hybrid Recommender Systems

Hybrid systems are the final broad category of recommender systems. These are methods to combine both collaborative and content information during the recommendation of results. This can be something as simple as taking the two methods and aggregating the results to creating a model that combines the information of both systems and produces a result. Hybrid systems can be further distributed as being weighted systems, switching systems, feature combination systems, cascading, feature augmentation systems, or meta-level systems that take in models as inputs. These encompass systems that are based on scores of multiple recommenders, switching between techniques, presenting the results of many recommenders, combining features from many sources, and finally using one recommender to adjust the recommendations of those preceding it . One popular area covered extensively within research, particularly within hybrid systems, are multidimensionality functions. Multidimensionality functions focus on combining overlapping features of users and items in order to find the users' ideal recommendations. This is further improved upon when combined with contextual information, such as time and place, of both user and an article to be recommended. Contextual information is a large factor in creating accurate recommendation systems and will be covered in the following sections.

2.3 Approaching the Recommender System

News Recommender systems provide articles to users that are both topical and relevant to their interests. This section will approach the challenges in recommender systems, particularly news-recommender systems, which will provide an idea of what the techniques used in such systems are trying to accomplish with regards to mitigating these challenges. Analysis of metrics used in these approaches are covered. The thesis will then move into Markov models which will heavily influence the approach used in this paper. The development of a recommender system will be defined and any considerations regarding how data is handled. Finally, coverage of how these systems are evaluated will be given.

2.3.1 Challenges to Overcome in News Recommender Systems

Before observing examples of recommendation systems, the three challenges that encompass building a news recommender system should be kept in mind. The first of these challenges, particularly a news recommender system, is the dynamic domain. As time passes, events occur and the news accumulates. As articles are added, relevance to previous news articles may wane as topics and vocabulary changes , and therefore clustering and comparison of articles and users change. Next there is the lifespan of the article. Not all stories retain relevancy as well as others. Some articles from a year ago may be more relevant than ones that were written a week ago. While the more recent articles are what news recommender systems should provide to users, the relevancy of these articles becomes more apparent as articles become older and users are either willing to or unwilling to interact with the age of the subject matter . This also applies to the lifespan of a user’s interests. Challenges in receiving the latest article also apply through the new-item problem , particularly where recommended items are based on previous reading histories of other users. Finally, there is the concept of serendipity. This is finding interests of a user without them implying, explicitly or implicitly, that a recommended article would be of interest to them. Serendipity reflects a system’s ability to identify completely different items to what the user is used to and its ability to recommend items that are unrelated to a user’s interaction history or other factors. This can be through ranking functions that encourages dissimilarity, or through users who have explored the same interests, but have items that are outliers within their view sets that have a high priority in being recommended to another user due to its outlying nature. This factor can expand user’s knowledge on topics that they would have otherwise not explored and retain their attention to the application . By acknowledging these three factors, a recommender system should be able to mitigate issues, like user disinterest, while providing engaging results.

2.3.2 Example Systems and Their Applicable Techniques

There are two general categories of systems. The first are knowledge-based systems. Knowledge-based systems derive their results from explicit comparisons on items contained in their databases. The other systems are model-based systems. Model-based systems return results through inputting the values required for a recommendation, like a user profile, into a model and receiving the recommendation based on the output of the model. Knowledge-based systems are typically used for news recommendation systems in particular, as there is a constant increase in content and users at every moment. This would have an immediate impact on a system and would be difficult to adapt a static model from. In the following section an example of a knowledge-based system with its own uniquely adapted method of information processing will be presented as well as a prominent method used in knowledge-based systems.

EntreeC is an example of a knowledge-based system that utilizes a technique called cascading. It uses levels of importance for item attributes that a user selects in their query to bring recommendations. A collaborative system is added on top to allow for future users with similar queries to gain different and better results based on how the current user critiques their search results . This added collaborative filtering creates an increasing level of performance over time, which is not found in standard knowledge-based systems, though knowledge-based systems show adequate performance with respect to the cold start problem .

Neighbourhood-based systems are a large topic in recommendation systems, particularly as they bring attention to distance metrics in associating items and users alike. Unlike the EntreeC system, neighbourhood methods are good for finding associations between data that are otherwise require keyword-matching searches like in EntreeC. This is because items can be linked through more general labels, or topics in a news system, as vectors. Lesser known items to be recommended can then be associated with more popular items through distance metrics that perceive association.

Allowing the recommendation of lesser known results can bring to the user new interests, beyond what is expected. This unexpectedness is called serendipity. Serendipity is an important factor in recommendations as it allows for users to uncover new interests that they would otherwise have not found . For example, if a user prefers articles on basketball, such a system may recommend them articles within the area of sportswear and technology. The topic may not be a reflection of the athletes themselves, but it reflects a step that they may take from that subject. This may also be a jump into a completely new topic area that can provide more novelty in the results. There is also the concern for obvious recommendations. For some systems, it would be important to keep items as familiar as possible. Other systems may benefit from having non-repeating items for the user between recommendations . What should be considered in a news recommender system is whether the user wants their views and interests challenged or if they want consistency in the items provided to them.

Neighbourhood–based systems are simple. There are no complicated calculations, the results are simple to justify to the user if needed, and the system is stable as it does not need retraining upon new data being added . Such a system is also applicable for both regression and classification algorithms. Regression is better for a ratings scale system whereas the latter is better for a binary “good or bad” system. Regression favours accurate results when ratings are involved, though classification provides opportunity for serendipitous results in this system . In a news system, minimizing interaction between the user and the recommender application is preferred, especially in a constrained system. In either case, the calculations for such metrics remain simple.

One notable method to encourage serendipity is to exclude groups of results from appearing for a period. This is the idea behind Sidelines. This algorithm is applied to the topics that define each item by adding a set amount of time those topics may not occur again in a recommendation. Through doing this, the sidelines algorithm naturally reduces the recommendation of popular topics and eventually evens the exposure of topics to users depending on the duration of the sideline . It is a solution for diversity that simply and effectively promotes differentiations in recommendation results in systems where items are classified categorically.

Nearest-neighbour methods can be used to evaluate item similarity or user similarity, thus an appropriate scheme must be selected . Depending on the system demographics, a strategy is chosen for a ratings-based recommender system. The criteria are: accuracy of ratings and factors like average number of ratings and neighbours, the time complexity with respect to the number of subjects being evaluated, stability of changing data with regards to how often the data is updated, justifiability, and finally serendipity. The latter factor usually prefers user-based techniques as they provide insight into what the user has interacted with and how the system can provide the user more serendipitous results .

2.3.3 Similarity Metrics

There are many methods of calculating similarity such as cosine-vector similarity and the variance-adjusting Pearson correlation. Weighing and normalization methods are usually considered in order to better fit into the user’s interests and adjust for various biases. Factoring in a user’s rating trends is an important consideration within a ratings-based recommendation system. This can enable the system to normalize a user’s predicted rating, based on their scale of rating to their rating trends, and adjust it based on the difference from the average rating in the recommended item. There are many normalization schemes available, each with their own benefits. Inverse user frequency, which is like Inverse Document Frequency in search engines, is typically used in this case to adjust the user's prediction with respect to the weighted similarity of users who have read and rated the article to be recommended . This weight is calculated using a frequency-weighted-Pearson correlation function that summarizes the similarity between two users . However, the idea of inverse user frequency is to reduce the influence of more popular items as it assumes that similarity is better represented by less common items and provide a weighted scale of influence per item . Without a collaborative ratings scale present between users, using total article interactions on each individual article to infer interests in articles is another method to infer the next appropriate recommendation. The calculations can be done similarly to a ratings recommender system with using shortest-path methods instead. For that sort of system, you would need to find the average amount of jumps between users or items before an item is reached. The item recommended would be the one with the shortest average number of jumps .

Within large systems, this comparison of users and paths becomes a computational issue. Therefore, for nearest-neighbours algorithms, filtering techniques are usually utilized so that not every user or item within a large system is used in a prediction. This keeps results relevant to a set of neighbours who meet certain criteria such as level of similarity. Examples of pre-filtering solutions are: Top-N filtering (limit the number of closest neighbours), threshold filtering (keep only those that satisfy a weight), and negative filtering (observing negative ratings correlations between users to find possible results) . There are a couple of problems with a nearest neighbour system: limited coverage, due to only being able to reach close users, and bad results when data is sparse in an area.

2.3.4 Context

Identifying key information in recommendation process is important when formulating recommendations. With respect to news recommender systems there are texts, labels, and maybe ratings which can be used to create relevancy and rankings of certain articles for a user. Potentially there is information beyond the article in the form of contextual information. Context is information that relies on the conditions and circumstances that make a user access a certain article and are usually vital in providing adequate recommendations. These are factors like time, location, life events, and other external factors, not part of a user or item's defining qualities, that may directly impact the user's choice of reading material . In news recommendations systems, accessible and mobile context aware systems are of utmost importance in providing users articles that are relevant to their setting. Recommendations to users may also be further influenced by the reading materials of users around them . Database management is also potentially valuable in a news recommender system with regards to context. Based on queries a user procures for articles, the context that backs the queries can be aligned with the user's future interests and to provide unique result , this is like the EntreeC system mentioned previously.

Context should be seen as an added dimension to previously mentioned systems. For example, where other hybrid systems would provide recommendations based on the dimensionality that is "User x Item → Recommendation", contextual systems add another dimension to creation of recommendations as "User x Item x Context → Recommendation" . Within context, each type of context available is to be considered. Every context may even have a hierarchy of sub-contexts within it that can be specified further to remove generalizations that include many of the sub-context categories (e.g. the "Weekend" context can be split into "Saturday" and "Sunday" sub-contexts). In the case of a news recommendation system it may be the location such as a Country, which can be specified to a province, further to a city, and even further to a riding in order to get more location-specific news, rather than news from all over the country . This directly reflects the people that read certain subject matter, whether they are widespread or not. This can then be further used to influence the choice of a ratings function when evaluating recommendations due to contextual factors that may make a user select a particular kind of item out of many other types . Such contextual information can also be implicitly or explicitly supplied by the user. This information is usually best suited to be defaulted by the system to a select range of attributes and updated over time to reflect the user's interests and provided context .

When implementing contextual information, there are three paradigms to consider: contextual pre-filtering, contextual post-filtering, and contextual modeling. Pre- and post-filtering techniques are similar as they both use context to select a relevant result, but before or after a model creates a list of recommendations respectively . In pre-filtering, data can be filtered during a search phase for articles to be potentially recommended, whereas in post-filtering it is done when the results have been selected by a model. Contextual pre-filtering has the chance of over-specification, due to the lack of hierarchy, but this issue can be solved through hierarchical generalizations of context to reduce three or more hierarchies into smaller and potentially 2D contexts . This outperforms normal non-contextual approaches . Heuristic approaches can extend traditional 2D approaches in post-filtering by using context as an added dimension (e.g. User x Item x Time), to create distance vectors between users and items and apply Euclidean or Manhattan distance calculation to find the most relevant recommendations .

When applying context and results derived from it there are a few things to consider. Context granularity is the first consideration. Granularity is how specific or generalized item and user context should be perceived . This is solved using hierarchical solutions to contextual information, which will be observed in the next section. Next is the persistence of long and short-term trends. These reflect how the interests of users evolve over time and how to adjust a system for it. The final factor is serendipity, which determines how flexible a recommendation system is in providing unexpected results. This is usually solved through various weighing schemes , which will be explored.

Hierarchical methods also encompass clustering items themselves via numerical linkages to particular topics. This can be used to easily assign a user to a cluster upon recommendation creation. These topics can be detected through labels, but in news topic analysis specific topics should be derived from text. This is done using probabilistic language model like PLSi, LSA, or LDA (Latent Dirichlet Allocation), which extract words from articles and assign weights to them . This can be similarly done through a NER (Named Entity Recognizer) method, which identifies key words within a document that represent it, such as people, places, or events . These topic weights can then be applied to items or users, through their access patterns of classified items, and allow them to be assigned to clusters . These keyword extractions, combined with word hierarchy, have the potential to mitigate concerns of vocabulary changes and further help link users through their similar access patterns and interests. A hierarchical function of topics determines how similar two items are in a hierarchy in the form of a sub-modularity function. This is a function can help differentiate between topics, such as whether an article is discussing the actors of a movie or one discussing the editing process of the same movie. Both topics are in the same general category of movies, but both general topics are essentially different and would attract different users . This functionality aims to fix the granularity of contextual implementation. It can then be further expanded on by being combined in a function that further determines weighted relevancy through popularity adjustments on an article and recency of an article .

The numerous model-based approaches find a way to append context information to user and item information and base recommendations on it. This may also include multiple models using different levels of hierarchical generalizations working together to provide optimal results, such as using context to determine which model to use to provide recommendations . Model-based approaches are very commonly used in text-based recommender systems. The first of such to be explored in this thesis is factorization.

Factorization methods are some of the most successful latent factor models in the realm of recommendation systems . They are a widely used method of finding features of items and users and compressing them into a smaller space. The main idea behind this is to have both the user and item matrix spaces represented in the same space and have their characteristics “factorized” into a smaller representation matrix of that space. This new matrix will represent users and their preferences based on their own attributes and those of their interacted items. This allows for new users and their limited information to be added more easily and have an immediate impact on recommendations. It makes the learning process faster and it overcomes the shortfalls of a neighbourhood-based algorithm while retaining similar ways of comparing users to potential items .

Factorization is completed by creating a new matrix that represents the user that is an approximate factor of two other matrices. Two other matrices would be the user matrix created from interacted topics and item matrices made up of the topics pertaining to the item. The new matrix can be used to approximate future user behavior with respect to their preferences as it is a generalization of their previous interactions. This combination of user and item information in a single matrix decreases the amount of necessary processing and memory required in supplying recommendations to users. Biases can be applied to the factorization to adjust the amount of influence item and user information has on the final factorization result. This can be based on how related items are, adjusting based on popularity biases, adjusting based on user interactions on types of items, classified by their static categories, and adjusting based on user preferences . Furthermore, contextual information may be used in providing more relevant recommendations and mitigating the effects of lacking information when adding a new user to a system . Many systems also implement factors such as time within processing or post-processing in order to dynamically adjust for relevancy to the user's current interests. This factor can be influenced by events, such as a new movie coming out for an actor that would bring more attention to their more recent movie than their previous ones, to which someone may give a lower rating or less interest to due to some bias towards nostalgia. This is an example of naturally drifting of ratings . This can be applied to news articles with respect to favouring more updated versions of events to dated facts.

Bayesian Probabilistic Tensor Factorization (BPTF) is one such method where model systems are influenced by time. BPTF is a forecasting model that applies time as the third dimension for the tensor instead of tags or queries. It is used to rescale relevance between users and items based on a user's history of interacting with items and predicting the next season's interactions, even though they may be different, based on the previous season's patterns . This even fully accounts for changes in completely different items being interacted with the following season, which is potentially satisfying the need to understand vocabulary changes and allowing for serendipitous results without sacrificing reliability.

More common methods of using time as a context involve comparisons to current time. One example is adjusting the popularity of an item with respect to current time and published time difference. Time can also be used with respect to time slots. These can be used to deduce a user's usage patterns, somewhat similar to the BPTF example, and providing recommendations based on the time slot a user is currently residing in. This essentially splits a real user into many virtual ones . Time may also be a factor considered with respect to the current session relative to previous ones. Whenever a user completes a new session, their session results are aggregated to the previous sessions. However, there is a parameter which determines the weight of the current session with respect to the ones before it. This will allow for long term interests to fade unless their topics are visited more frequently, with respect to the percentage of relevancy an old session is given. The lower the percentage, the faster the context of previous sessions is forgotten .

The forgetting mechanisms are as important as appropriately weighing the user's more recent experiences in time-decay functions . While the previously mentioned method was done through fixed weights to determine the impact of previous sessions to the most recent one, there exist methods that perform this forgetting function more precisely. Relevance-based forgetting is one approach which assigns the user's interests in certain topics on a scale from -1 to 1. Based on how often the user interacts with a topic, the user will have their interests based on this scale which will adjust on all topic categories based on the most recent item interacted with and the items that were still relevant during the previous relevancy calculation (e.g. without a relevancy of -1). Each new page to be recommended to the user will be based on the profile derived from these calculations as being relevant to the user (i.e. closer to 1), somewhat relevant (approximately 0.5), or would definitely not interest the user (i.e. -1).

Similar and simpler methods of applying decay of item relevancy over time involves using a logarithmic time-based decay function. These methods extend the idea of putting more importance on more recent items and compensate for changes in user interest over time during the desired time-period to be used for relevance. As time passes, items that approach the age of decay logarithmically approach zero in relevance. The items within this time-period are used for developing the user's profile in interest and their weight of influence on the user's profile is determined by the logarithmic function . This is a simple and effective way of applying time as an influence both for user profiling and for applying weighted scores in item recommendations.

2.3.5 Mitigating Unknown Factors

With respect to methods of factorization, or matrix decomposition, sometimes matrices can be more than two dimensions. There are various methods of decomposing 3rd order or higher matrices into smaller versions such as Tucker's decomposition and Candecomp/Parafac decomposition . With every dimension added to a matrix, there are more relationships being observed. With a 3rd order tensor, the relationship is being observed as one between three entities. Decomposition is used to link users to resources through smaller tensors consisting of non-zero values influenced by the user's actions, such as queries or clicks on certain articles that fit into particular topics. Tophits is an example of using co-occurrences of items appearing within user queries basing relevancy for recommendations on them. This query and keyword matching is typically done through crawlers that link together items in a database rather than an explicit model .

There is also the concern of unknown values in the matrix, which should not just be set to zero as this would eliminate the possibility of serendipitous results. This is addressed in Ranking Tensor Factorization methods, which are an extension of CubeSVD methods that fill in missing tensor entries by interpreting negative feedback from the user in the form of incorrect tags the user has assigned to an item . Methods of assigning values to unknown values may also include simply assigning confidence values to observed and unobserved events with respect to the representative user matrix or vector. High confidence values can be assigned to articles a user has read whereas lower confidence values can be assigned to values that have not been observed by the viewer, but the viewer may prefer. These events can then be applied to the user item set during factorization .

TAPER (Tensor Based Approach for Personalized Expert Recommendation) is a system that aims to apply context to factorization. Through the actions of "expert users", TAPER applies context in the form of geo-spatial, topical, and social information of the user of the application with relevancy to the experts and returns results based on the choice of experts. User and Expert factorization matrices are compared and similarity is observed through distance metrics . Experts are assigned to users, so various factors are considered. Factors like Tobler's law are to be used with respect to geographical context, which means that users that are close to one another usually have more in common and are able to be used in providing relevant recommendations . This consideration also mitigates the new-user problem, as recommendations can be applied based on other users' proximities. Applying all three contextual considerations within factorization or vector-based methods provides a significant boost in the accuracy of recommendations . With respect to finding similar users, similar interactions must be considered if a system were to assign a user to a community. This way, recommendations can ultimately be provided to users based on others with similar interests . Furthermore, to address the concern for lack of serendipity, cross-domain recommendations can be used for topics a user has had limited interactions with. This is essentially looking at the reading patterns of one domain and using them to predict the reading patterns of another domain with similar interests . Similar access patterns are used to determine potentially similar interests between user groups.

An effective example of defining a domain with items is StreamRec . This system defines users as the set of items they interacted with. Each user is a table of item similarities between each item a user has read within a time interval. When comparing two users to recommend new items, the user to be recommended, User1, checks the similarity table of the other user, User2, for items they have not interacted with that have similarity metrics based on the item they want to get a recommendation of. The item to be recommended is then given a predicted score. This is done across multiple users and potentially across multiple items that User1 wants to base their next recommendation from . As similarity metrics are calculated as a user reads a new item, there are no extensive lookups of items or many more calculations needed, if any, when a user wants to find similar items to the ones they have read. This also opens the potential for developing domain profiles upon similarity calculation.

2.3.6 Variable Markov Model Recommendation

Another approach to accounting for serendipitous results is through Markov Models. Markov Models are a sequence of random variables, with each sequence being like an order of events that have a probability of occurring at each step of the sequence. Thus, each step taken in this sequence can be determined by a probability assignment P(A|S), where A is a state and S is the series of states that occurred before it . The idea behind these models in a news recommender system is to increase exposure to articles beyond those that would be the most popular, or otherwise the main headlines on the front page. As a Markov Model's creation is probabilistic based on the current state of the environment, it reaps the benefits of not only capitalizing on the information that it was built on, but also in representing future trends as more data is added to the set. They are also used for maintaining grasps on the most recent articles, which would be those that are the least read due to their recency. Markov Models approach these solutions through context trees.

Context trees are an approach to provide recommendations that are fresh and can be updated incrementally to provide fresh recommendations. They are a subset of Variable Markov Model (VMM) approaches used to represent data as a series of symbols or contexts used to create a predictive model. The general idea behind context trees is to provide more specific contexts as a user goes down branches. Each branch, or node, in the tree is a finer grained subset of the previous one . The branch that the user is on and the ones that they had traversed in the current sequence can be used as a collaboration of experts to provide the list of recommendations for the next item set. Old contexts are removed from the tree as they expire over time and new contexts are added as new news becomes prevalent and the user traverses the recommendations. A way of building context trees is through a sequence of articles and topics. If a topic within node expires, the node in the tree is removed from the pool . When a user reads an article, they traverse to a new node with the added context of the article they interacted with and are recommended another group of articles that are related to the added context and all of the other contexts that preceded it in the sequence. The user is presented an article from each of the potential future branches. Every preceding node to the node that a user is sitting on is a weighted expert used to predict the order of the articles presented to the user. The weights of an expert node's prediction are calculated through methods like Bayesian probability based on how correct it is in predicting the user's interactions with items. The higher the expert weight, the greater influence it will have on the order of recommended items.

The general idea is calculating the probability of stopping at one of the possible contexts given the expert's current standing. Examples of possible models are: Bayesian probability, Pop-rank models, Freshness models that only recommend least read items that were recently published, or mixed models . For example, given that **a**, **b**, **c** are the possible contexts to go to on top of the contexts the user is being recommended from, their probability of adding one of those contexts is based on the context sets that they have previously interacted with. As a user selects an item, an expert's influence will increase depending on whether they had correctly predicted the next preferred item. These methods can have various approaches: VMM Recsys that was explained in the previous paragraph, Content-based VMM Recsys that uses LDA to apply topics to items and then the tree is built through the likelihood of jumping from one topic to another, and Hybrid VMM Recsys that combines the two other mentioned forms using a k-d tree . A k-d tree is a tree where topics are used, rather than contexts and articles themselves, to narrow down groups of similar articles. They effectively organize and store information for fast accessibility, retrieval, and modification of storage through modifying the tree's branches . Creating a tree based on topic distribution is possibly even more beneficial, as topic similarity can be used to recommend articles that are potentially interesting, but otherwise do not gain much coverage in the database . Given the constraints of a stream-based one-by-one approach of the proposed system, the methods that will be followed will have to revolve around the latter k-d methods of next-article identification and clustering. These VMM approaches are reminiscent of the uses of previously mentioned hierarchical methods and can therefore assist in clustering sub-modularity of articles as an added benefit.

2.3.7 Build of a News Recommender System

Modularity of a recommender system is very important in the construction. The CROWN recommendation system will be used to show and create a general idea of the flow of the component modules in a recommender system. The four components that make up this system, and are a good consideration when building any system, are: preparation, the recommendation engine, user modeling, and the user interface . Preparation consists of news crawling, extracting items with features that reflect on the contexts behind users accessing items, and similarity computation between items and users . Classification and clustering of items occurs in this portion of the system. Within Recommendation, tensors are factorized from user and item information using a HOSVD function, which also infers missing entries within the tensors . With respect to a system that uses vectors, this includes any distance calculations between users, clusters, and individual articles. The user profiles incrementally update as the user accesses more articles. User modeling in a system combines keywords and weights in a vector space of read articles in combination with user implicit and explicit data, which is used by the system to provide a recommendation list view of articles .

2.3.8 User Data Collection

How user information is used is key to providing catered results in a recommender system. When considering how user data is used and collected, the concept of intrusiveness must be considered when looking into the user space. Depending on the way an application is made, the benefits and shortfalls as to how user information is collected and used must be considered in reflection of both privacy concerns as well as the user-friendliness of an application. This is finding the balance between adding functionality, getting optimal results from a set of data, and respecting user data sensitivity.

Allowing users to have control over a system is one way of tackling the issue of user trust and increase recommender result validity. This may be giving the user some sort of control of how their results are formed or presenting feedback to the user that can be used to interpret why recommendations are being returned . Allowing users to interact and have transparency to the consequences can be used to further enhance the user experience through increased understanding of how the system works as well as benefit those creating the system in understanding how to build it. This further leads to user trust in the application and less interaction in the long run, due to the site being able to form an optimal system for users as they interact with more items. User retention can then be observed and user communities are formed in many cases which benefit in site development and item labeling . It may be that recommendation systems should be more focused on the initial information gathering component to build trust with the user, rather than focusing on the effectiveness of the system when information is available. With respect to new users, it is evident that the initial impact of transparency and accessibility is one of the first issues to approach in a recommendation system.

Acquisition of user information should not compromise the user's positive experience with the application . Experience with the application's design should be reflected in three design factors while maintaining accessibility: freshness, popularity, and relevance . Through freshness, a user has access to more recent articles and thus also be catered to their most recent interests . Displaying article popularity provides insight to recommendation and also conveys to a user that information is topical at the time . Finally, relevance is the way that an application will be able to clearly show the user why they are being recommended an article . Once these three factors are effectively fused with consideration of the required information to make recommendations, an effective system proposition can be put in place.

A system called “Focal”, provides an example of a recommendation interface that shows information in graph form in the user interface. Freshness, popularity, and relevance are represented by node transparency, node size, and edge thickness respectfully . It shows, at the very least, that a news recommender system does not have to be limited to a list view, but can be expanded on creatively and reviewed in effectiveness from there. User control in the system is a way to expand upon these three factors by allowing the user to have a direct influence in the recommendation system itself.

“Contextual”, a system developed by the Norwegian University of Science and Technology aims at allowing users to have direct interaction with the recommendation system. The idea is to give the user the ability to influence the system through a user interface that enables them to adjust their contextual and topical preferences in order to make the system as useful as possible for them . This allows for a high level of transparency as users can ultimately decide the freshness, popularity, and relevance of their searches. This then drives the users to adjust to their own level of the three challenges of recommendation systems (serendipity, relevance, vocabulary) and the particularly useful setting of locational context that suits their preferences . Balancing interaction and usability has experimentally proved that users are generally satisfied with the quality of recommendations that results from such interactions. Additional interaction does prove to have beneficial outcomes in recommendations, but conflicts with the need for minimal work on behalf of the user.

As the Sidelines project had shown, users tend to show no significant difference in satisfaction when considering whether they are receiving information that is targeting them or information that aims to sideline as much popular information as possible by giving less-popular articles a chance being used in the recommendation system . This project also concentrated on people’s political views and found that sidelines challenged their perspectives. It was found that those that were challenged in their views valued the diversity as opposed to readers who had had results based on pure popularity recommendations . Thus, serendipity is noticeable and ultimately has a positive effect on reader experience, no matter what their perceived satisfaction is. Viewpoints can also be taken into respect to showing users topics they are otherwise unaware of and provide exposure to new perspectives and topics of potential interest.

2.3.9 Evaluation Beyond Accuracy

It is also important to consider what data is being recommended, no matter the accuracy. Some algorithms ignore factors like RMSE, precision, and recall and focus on one aspect of the service that is vital to the recommender’s functionality. Algorithms that lean towards items that are popular may have good recommendations and accurate simulated click-through rates, but would completely avoid providing the requirement of serendipity. What has been found in relation to unpopular data in many sets, is that the removal of less popular items does increase accuracy, until a threshold is reached at which point it rapidly declines. Thus, pure popularity in a news recommender system is not an optimal solution . For other recommendation systems not focused on strictly optimizing stats like RMSE, the average ratings of retrieved items are found to be like the global average rating across all items. This implies that factors like novelty become a possibility within the recommendation system and provide results of similar quality to the users .

Catalogue coverage is also very important within a news recommendation system. The more relevant items are covered by the system, the less wasted effort there is on the whole system and the more user engagement there is. The goal for this is to have as much of an equal, or reasonable, distribution as possible between the most popular and the less popular items. This measure is provided through the Gini coefficient. The Gini coefficient is a scale from 0 to 1, with 0 being total equality and 1 total inequality , it is found that factorization methods like ALS, Koren-MF show to have different results across multiple datasets, but the results show diversity . Sidelines algorithms are also particularly used to give less popular items an opportunity to be recommended and ultimately result in more diverse results.

When adjusting a model to have ideal performance, the optimal step is found to not be when the RMSE is lowest, but when one of the other factors concerning Gini, diversity, or the number of recommended items reaches a peak low for Gini or high for diversity. Optimizing these metrics are just as important in maintaining user interest as they can promote serendipitous and novel results without compromising relevance to interests . There are also other methods of avoiding popularity biases. One is to flip the probabilities of selecting the most popular items with those of the least popular before evaluation. This is how the Bayesian Personalized Ranking (BPR) algorithm works. When such a system was implemented upon a few recommendation methods, the bias decreases with the accuracy, though the accuracy decreases slower . Adjusting the model does not stop when RMSE is at its lowest. It completes when diversity reaches a peak. This will happen before any significant drop in accuracy .

Other algorithms include adjusting the predicted scores of items in post-processing. This would include increasing the relevance scores of unpopular items slightly and possibly even lowering the ratings of popular items . ALS factorization showed similar changes as BPR when such an algorithm is applied to it, while insignificantly changing RMSE and decreasing the gini index to ensure a more distributed hit rate of the database. Adjusting this in post-processing is easily done and can maintain a desired bias within the application . Diversity, however, tends to decrease as the number of recommended items increases, though this is to be expected as there is an increase in potential categories.

With respect to context trees, evaluation is done with observance of accuracy of the recommended items, personalized items, and novelty. Personalized items are with respect to guessing correctly the reduced set of items the user would want to interact with in proportion to the total item set. Novelty is the ratio of unseen or unpopular recommended items over recommended items . Like diversity being key in providing the user varying and different items, this is the metric that gives users a chance to be recommended less-common items. Given the restraints of the system to be proposed, understanding the popularity of an item to create novelty is not entirely possible. There are a few ways to approach this: Discovery-based novelty and Distance-based novelty . Discovery-based is reflective on the probability a user is familiar with an item. The goal of this is to identify the items that are most likely in the long-tail of items frequently seen. . Given the constraints of the system to be proposed in this thesis regarding the requirement for no repeated items, this metric is not practical to implement. The second method, distance-based novelty, is a calculation of how different an item is from the user’s past experience. What is looked at in this case is the similarity between context and the item given. This is calculated as the probability a user selects an item under a context. This is done using the complement of a similarity metric between the item previously viewed and to be viewed, normalized to Boolean, multiplied by the probability of selecting an item under the current context, over all relevant contexts . Novelty, and all other evaluation metrics, should be calculated with respect to a reference set of items. Depending on what further restraints are applied to the system, an appropriate selection for novelty calculation should be made.

2.4 Literature Discussion

Though it may seem that creating a news recommender system is based on compromises and considering the benefits of methodologies and finding a balance, the approach will likely greatly vary based on the architecture provided.

The first of these considerations is the choice of generalized system. Depending on the amount of user information available, a general content-based, collaborative, or hybrid system should be chosen. Some experiments have found ways of going around a lack of any user information, one simulating users through sessions of connection acting as individual users. It is clear however, that much of the possible contextual information relies on the acquisition of user information in one way or another and the acquisition of specific user information would be key in providing a catered recommendation system.

After identifying the available data, the approach to recommender systems should be selected. The proposed system should be able to adjust or avoid conflicts with too much or too little serendipity and avoid bias. Experiments have shown success in both pre- and post-processing techniques in mitigating issues in these factors. This may be with regards to, but not exclusive to, adjusting ratings to recommendations, changing the order of processing recommendations, or producing a new algorithm based on existing proven methods that finds ways to elude popular results and allow for less popular results to have their recommendations present. The prevailing research methods for these factors is through influencing the models by adjusting how they apply weight, bias, and ratings normalization. It would be interesting to see more comparisons done between pre-, post-, and model-based filtering, and observe the conditions where each method, particularly the latter one, works more effectively.

Factorization appears to be an effective method to produce recommendations of news sources. Context is a potentially vital factor in presenting effective recommendations, and factorization appears to be the most effective in applying context across all attributes. Furthermore, factorization simplifies the ability to create relevancy between context, users, and items alike into one simple repeatable process. This also includes the inference of otherwise missing information. Using these methods to fill in missing values in matrices based on assumptions or other correlations allows for serendipity as well as significant coverage of articles in the database. This range of ability in factorization has been proven to be of value within the sidelines as well and further amplified by the importance of diversity. Factorization methods have proved to be more effective at reaching consistent diversity across datasets, while retaining a high amount of accuracy. Regarding ratings, it does not have to be a factor in deciding whether an item is relevant to a user. Factorization enables relevancy analysis beyond ratings using generalizations to which an item can be recommended to a user through the mentioned assumptions on access patterns.

However, in a dynamic system in the realm of news recommendation, the constant refactorization of user interactions and context may not be practical in providing real-time-quick-updating profiles recommendations. VMMs provide similar benefits to factorization techniques with regards to the potential for high novelty and diversity without significant compromise in accuracy factors, as well as the potential to use effective weighted ranking algorithms within their nodes. Tree-based models dynamically adjust to new items coming into a news system and can more easily adapt to the existence of newly formed categories that could appear as new topics appear. VMM algorithms are an appropriate approach to news systems.

The challenge is to find an appropriate scheme based on the dataset given. Through the kind of information given, format, and context, a system can be derived to provide optimal results to users.

Within such systems, various methods of deriving significance of attributes, like word parsing and hierarchical analysis of attribute to determine relevance, can be considered. Once a system that satisfies the challenges of a recommender system is created within the constraints, a design that appropriately balances the required metrics and meets user interaction requirements can be tested.

Chapter 3

Methodology

This thesis will propose a stream-based news recommender system. The scraped news is categorized based on their contents and organized so that various recommendation methods can be applied on top of the proposed approach. The objective is to identify the appropriate method of recommendation in the constraints that grant limited room in providing the user results. We are going to build a VMM used by all users as opposed to a VMM created for each individual user. Amongst these algorithms, a modified version of an effective approach to news recommender systems will be used for clustering. This chapter will discuss the flow of information in the system architecture, the proposed classification and clustering method, and finally evaluate the proposed ranking models to be used to give insight on what can be expected from each approach.

3.1 System Architecture

In this section, the flow of data in the news recommender system will be covered. An objective to keep in mind is that this system needs to provide recommendations in real-time. As new news articles come into the system, the system should be able to capture it and recommend it to the user as soon as possible. The data flows from component to component and is finally recommended to the user in a stream of relevant articles. The flow of the articles through the system is as follows:

1. **Scraper**: Retrieves articles from a set of news sites. Places articles in database for further processing.
2. **Filterer**: Identifies articles that are not duplicates of existing articles and meet requirements, such as size and language, to be classified as articles valid enough to be

processed. Saves them in database separately from scraped data and flags filtered news as being filtered.

3. **Classifier:** Views filtered data and assigns NER and LDA classifications to the articles. Filtered articles are updated with LDA classifications and marked as classified.
4. **Node Manager:** From classified articles, produces a VMM-Tree based on cumulative and time-decayed topic scores of recently processed articles. Serializes and stores the tree in the database.
5. **Recommender:** When a user interacts with the system, it traverses the tree to a topic set it deems is relevant to the user's interests and recommends articles from that topic, based on a ranking model.

The following sections will cover the system in more detail, concepts will be described, and formatting will be covered. **Figure 3.1** illustrates the flow of the system.

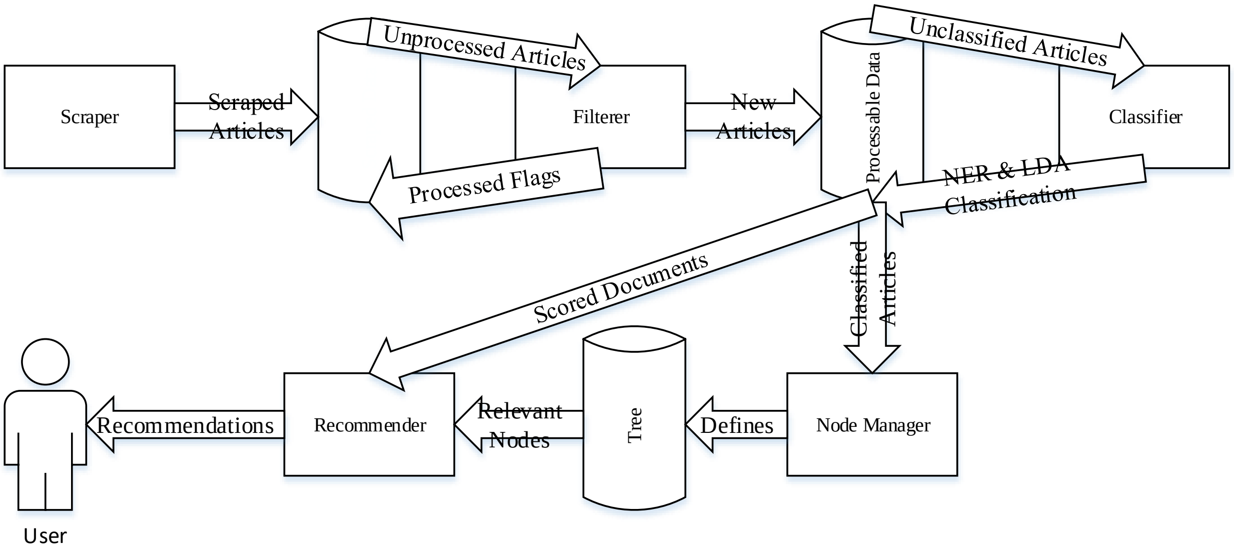


Figure 3. 1: System Design

3.2 Scraper

The news articles are acquired from a scraper that was built for this project. The pages for scraping are taken from Google News custom RSS feeds that are filtered based on the websites desired to be used for scraping. This limits results to the most prominent current articles on each site. News sites are selected for their coverage of events as well as their coverage of topics. Many news sites cover major world events, so they will provide multiple perspectives on prevailing topics. As there will be greater coverage of events, the models used to classify and cluster articles will be more refined, as common word combinations across articles will benefit LDA topic training, and contain better represented topic categories for the recommendation process. Other sites that have specializations on sports, entertainment, and science are also selected to add more potential topics into the mix. This also allows for topics that are less covered in world news sites to be better represented during recommendation modelling. Localized news sources, like Toronto Star and City News prominently provide for their areas, were selected. Considering that the experimental phase is done by subjects within the area covered by these sites, it is beneficial for testing to include articles that are more likely to be locally useful to the reader. Other sites, like Global News or CBC, may also include information on the Toronto area as they encompass Canadian news. Likewise, The Toronto Star may cover prominent world news or Canadian news. These news sources cover both local and world news and will ultimately lead to better modelling and user interaction due to their relevancies to one another and the other sites accessed during the scraping process.

The scraper is run at regular intervals through a list of Google News RSS feeds. Each feed provides the queried results from Google News on the selected sites from the last two days. On a daily basis, there are 500+ uniquely published articles from the websites designated for scraping. These articles are saved to a NoSQL ElasticSearch database in the format shown in **Table 3.1**. As this dataset is constantly being written to, the filter is accessing the information in it and moving it to an Elasticsearch index for classifying.

Table 3. 1: Article Format

Field	**Description**
date_publish	Article publishing date
image_url	Location of title page image for article
title	Title of the article
source_domain	Publisher name
Text	The contents of the article
URL	The web link to the article to be given to the user

3.3 The Filter

The filter is the module in the process that moves the information from one database to another based on its validity. The filterer deletes each article in the scrape database after a move in order to conserve hard drive space. This scraped news information is moved to another index using a deduplication and language identification program to remove duplicate articles and restrict processed documents to English-only. For deduplication, the program indexes Elasticsearch item ids based on URL, up to the first "?" in the URL. As this deduplication is not effective for all URLs, as some news websites will have multiple distinct URLs for the same article, a SHA-256 hashing scheme is applied to the text as well in the new index. In case the URLs are not the same, further comparisons will most of the time find duplicates of documents as the text will be the same and thus their stored hashes will be the same as well. This allows for faster retrieval and identification of duplicate documents. This also applies for not moving previously deleted and re-scraped documents. This is still not a completely foolproof method of removing duplicates, as some documents are in American English and others are in Standard English, but the presence of these duplicates is negligible. All documents moved in this way are flagged as processed in the former database, so that they are not accessed again for a move and confirmed duplicates are not moved. This entire process is necessary for the classifier to not learn a bias from repeats of the same document for the inferred topics and allow for a larger range of topics to be available.

3.4 Classifier

Moved documents are processed by the LDA Classifier program. LDA is a model of multiple topics that are each defined by a set of words with weight distribution scores relevant to the topic they are located in. This is done through multinomial distribution calculation of these words. Each topic is defined by the same number of words. **Table 3.2** shows 3 topics defined by 5 weighted words each.

Table 3. 2: Topics Defined By 5 Words Each and Their Percent Affinities to Those Words

	Topic A	**Topic B**	**Topic C**
1st Word:score	Politic:0.3	Nature:0.45	Sale:0.37
2nd Word:score	Parties:0.25	Earth:0.25	Business:0.31
3rd Word:score	Law:0.2	Animal:0.12	Stock:0.22
4th Word:score	World:0.19	Land:0.10	Equity:0.045
5th Word:score	Bill:0.6	Forest:0.8	Share:0.045

Documents can then be sent through the model and assigned a relevancy score to each topic based on the document's matching of each topic's words and their weights . LDA was selected as it is a proven method of capturing multi-topic phenomena within long documents , the parameters, words per topic, number of topics, and amount of training iterations can be easily adjusted to set the level of generality of these topics, and creates an easy to understand language model that is the result of approximate estimation of values . It has also shown to effectively solve issues with heuristic-folding of contexts into a two-dimensional space in comparison to other methods due to its approximations . LDA can also assign topics to documents that the model was not trained on , which is important in maintaining a consistent stream of new files into the system. Depending on initial testing of the program, the number of topics created per LDA training run can be decided. The format of this data is similar to the format in **Table 3.2**, with the additional nested "TopicScore" field. In this field the top normalized topics are stored in a list, each in the format found in **Table 3.3**. This allows for quick access to these fields in Elasticsearch , particularly with respect to building or traversing the VMM Tree to be mentioned in the next subsection. An LDA model is created based on the group of documents fed into the LDA Classifier's scope of news articles.

Table 3. 3: TopicScore

TopicScore field	Description
Topic	Query-able topic name (String)
topicScore	Query-able topic value (Double)

In addition, the Classifier also contains a Named-Entity (NE) extractor. This identifies the key nouns in the document and lists them out as a property of the document. Before LDA classification, these named entities are identified. A "Document-Dependent Named Entity Promoting" algorithm is then used to increase the impact of named-entities in the document while it is being classified. This is done by identifying the maximum term frequency of a non-stopword in a document and increasing the frequency of each named-entity of the document by that frequency . For example, if the most common non-stop-word appears 17 times then before LDA or classification of that particular document, every word identified as a Named Entity is appended to the document contents 17 times. LDA model building and document classification is then run on this modified text. If the noun "Toronto" is identified as a named entity by the extractor, it will be appended to the document the amount that the most frequent non-stop-word in the document exists. Doing this will improve the quality of LDA topics through promoting common and valuable words in appearing within documents for modelling. When used on documents being classified, this will also improve the quality of topics being assigned to documents that contain the named entities. This algorithm is illustrated in **Equation 3.1**. This equation replaces the frequency of a word in a document, Freq(w), with a new frequency, NewFreq(w), if it is a Named Entity, NE, with the sum of the total frequency of the named entity and the non-stop-word that appears the most in the document, maxFreq.

$$NewFreq(w) = \begin{cases} Freq(w) + maxFreq, \ if\ w\ is\ NE \\ Freq(w), \ else \end{cases} \qquad \textbf{(3.1)}$$

This is also done with the title and the website address. So, if the most frequent non-stop-word appears 23 times, then the title of the article, the article URL with spaces replacing non-letter characters, and named entities will be appended to the document 23 more times.

3.5 The VMM-Tree

Before the user can access groups of articles that share common attributes, items must be clustered. In this work, a VMM is used to create sequential context trees based on the topics related to the articles, with nodes that can be easily used to recommend a list of articles . The tree is constructed on different sequences of topics. **Figure 3.2** is an example of a small context-based VMM.

3.5.1 Construction

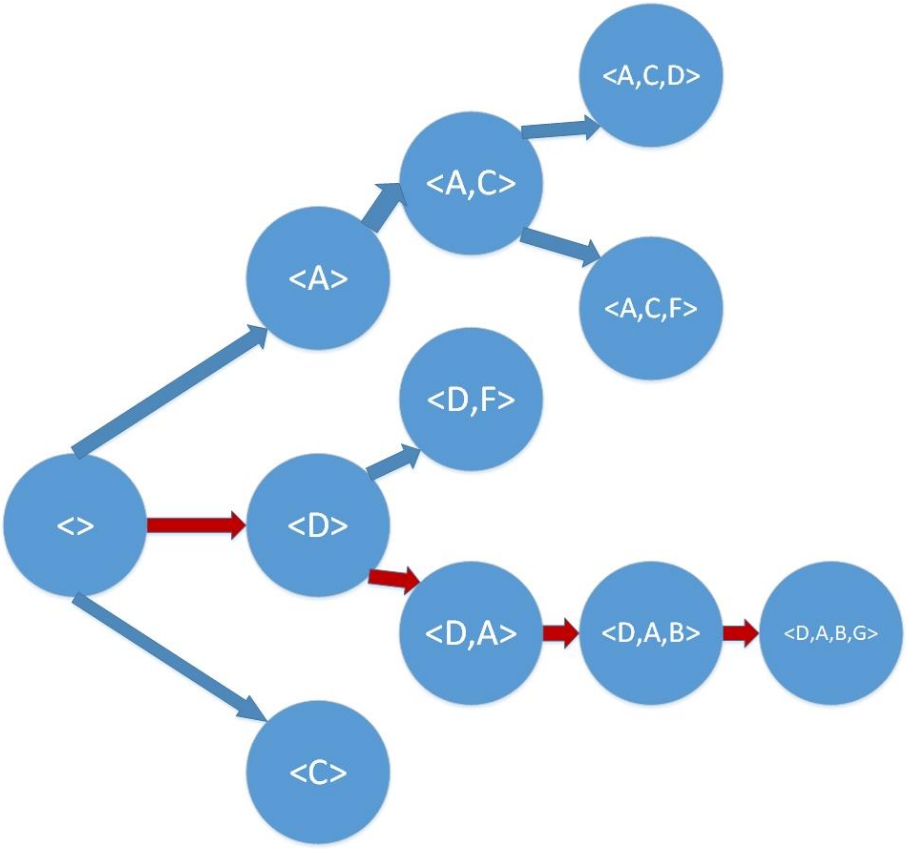
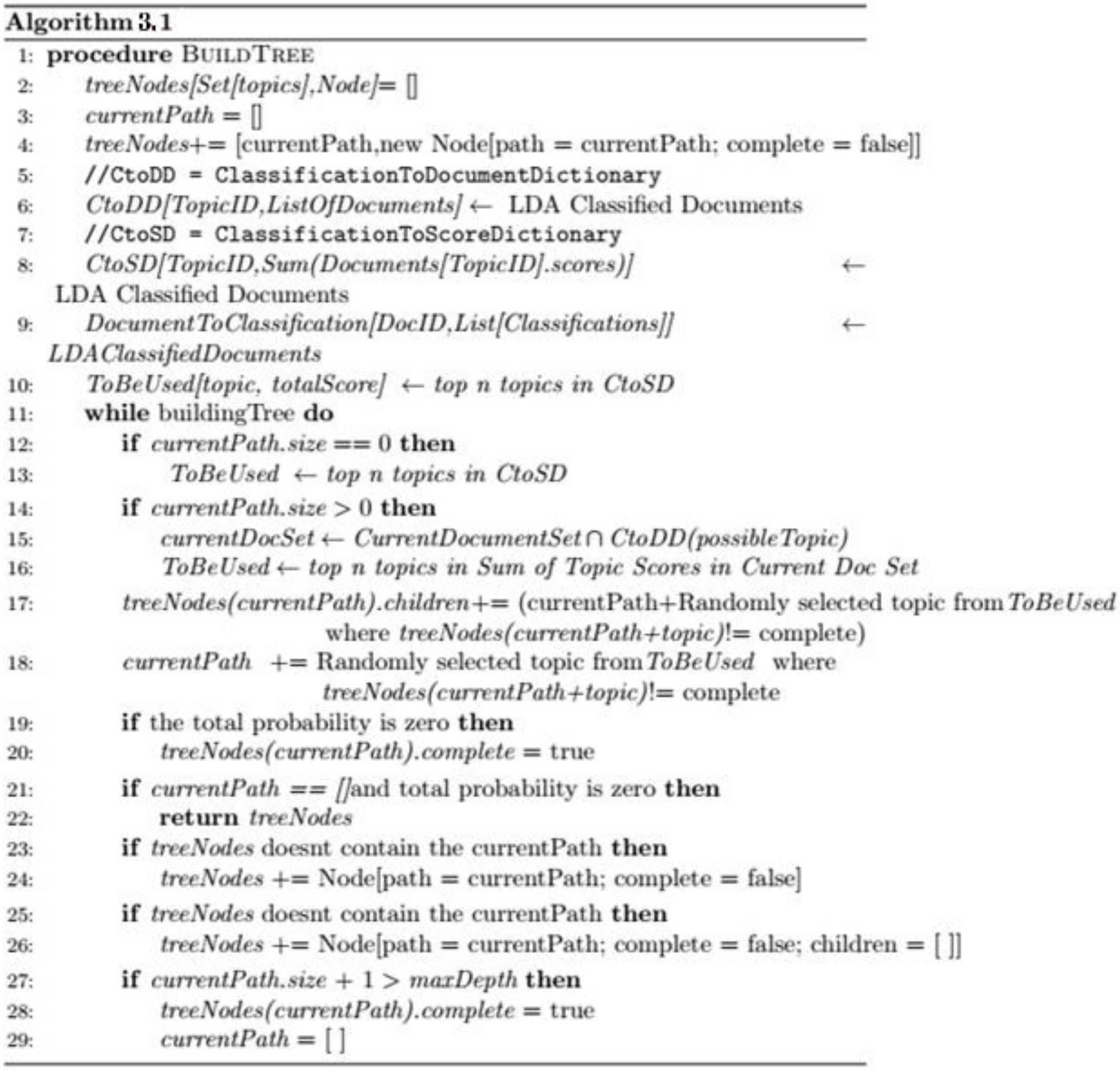


Figure 3. 2: VMM-Tree Example

Branches within these context trees are built one at a time. In the above diagram the red arrows in this tree indicate the path of a branch that is currently being constructed. Each node represents a set of topics that define a set of articles contained within the node. So node <A,C,F> contains articles defined by all of those topics. Every context tree in this manner is initialized from an empty root node "< >", or all articles, that has no context applied to it. Subsequent nodes contain a subset of the node that was built before it.

Table 3.2 is an example of what the topic letters in the tree may represent. Each topic is defined by a group of words with scores representing the percent affinity of the words to the topic. The more an article matches the words, the more highly a topic will be scored against an article.

Algorithm 3. 1: Tree Building



Algorithm 3.1

```
procedure BuildTree
    treeNodes[Set[topics],Node]= []
    currentPath = []
    treeNodes+= [currentPath,new Node[path = currentPath; complete = false]]
    //CtoDD = ClassificationToDocumentDictionary
    CtoDD[TopicID,ListOfDocuments] ← LDA Classified Documents
    //CtoSD = ClassificationToScoreDictionary
    CtoSD[TopicID,Sum(Documents[TopicID].scores)] ←
  LDA Classified Documents
    DocumentToClassification[DocID,List[Classifications]] ←
  LDAClassifiedDocuments
    ToBeUsed[topic, totalScore] ← top n topics in CtoSD
    while buildingTree do
        if currentPath.size == 0 then
            ToBeUsed ← top n topics in CtoSD
        if currentPath.size > 0 then
            currentDocSet ← CurrentDocumentSet ∩ CtoDD(possibleTopic)
            ToBeUsed ← top n topics in Sum of Topic Scores in Current Doc Set
        treeNodes(currentPath).children+= (currentPath+Randomly selected topic from ToBeUsed
                            where treeNodes(currentPath+topic)!= complete)
        currentPath  += Randomly selected topic from ToBeUsed  where
                            treeNodes(currentPath+topic)!= complete
        if the total probability is zero then
            treeNodes(currentPath).complete = true
        if currentPath == []and total probability is zero then
            return treeNodes
        if treeNodes doesnt contain the currentPath then
            treeNodes += Node[path = currentPath; complete = false]
        if treeNodes doesnt contain the currentPath then
            treeNodes += Node[path = currentPath; complete = false; children = [ ]]
        if currentPath.size + 1 > maxDepth then
            treeNodes(currentPath).complete = true
            currentPath = [ ]
```

Algorithm 3.1 is a low-level view of how the tree is constructed. At the end of construction there will be a dictionary of Nodes identified above as *treeNodes*. Each node is indexed in this variable by the set of topics that define it. Within each node is a list of topic sets that are its children, which can subsequently be looked up in the dictionary to find the next node’s information. The tree stops constructing when all Nodes constructed are identified as complete and no further branching can be done.

The tree is built probabilistically from the empty node “< >” onwards. The total LDA values of the articles contained within each node determines the probability of each of the LDA topics in being the next topic added to the topic subset in tree construction. In **Algorithm 3.1**, the totals of each of the potential topic branches is represented by variable *ToBeUsed* in line 10. It is defined in lines 12 to 16 as the total of all topics if the build is sitting on an empty node or as the LDA totals from the available articles in the current node if the node contains one or more defining topic, such as <A> or <D,C,W>. The new topic is then probabilistically selected to be added to the current path based on its probability to be added. Nodes that are at the end of a possible branch list themselves as “complete” on line 20 and 28.

Table 3. 4: Article-Topic Affinity Example Using 2 Topics

Article Name	TOPIC **A**	TOPIC **B**
Article 1	0.05	0.2
Article 2	0.25	0.0
Article 3	0.12	0.12
Article 4	0.7	0.15
Article 5	0.0	0.11
Article 6	0.33	0.35

For example, referring to **Table 3.4**, the possible topics to branch to could be only *A* or *B*. If the total of the topic scores for all articles belonging to *A* is 1.45 and the total of topic scores for all articles belonging to *B* is 0.93, then *A* will have a 60.92% (1.45 out of 2.38) chance of being the node added to the tree and *B* will have a 39.07% (0.93 out of 2.38) chance of being the next topic to add on the next node that is branched out on. Likewise, if there are more topics to choose from, then the percentage of the branch out will be proportional to the additional topics, not just A and B. This process continues from the branched-out Node <*A*>. The possible topics to select will be similarly probabilistically selected, except only using articles defined by topic *A*. In **Figure 3.2**, the next branch-off selected is *C* from all items that are relevant to topic *A*. The node that the building process now resides on is Node <A,C>. From here, the run considers all articles that contain both topics *A* and *C*. The next step could possibly branch out to include topic *D* or *F*, or some other possible topic that is within the articles defined by topics *A* and *C*. When either a maximum depth or a threshold to the minimum number of items in a node is reached, the tree will stop the current branching run. For example, if the current process is at Node <A,C,D> and the maximum depth is 3, then the current branching will end there. Likewise, if there are a total of 6 articles that are in that node, where the minimum number is 5, and all the possible child nodes like <A,C,D,X> and <A,C,D,Y>, as an example, have less than the minimum of 5 articles, the current <A,C,D> node will be listed as being a complete node. This means there are no more possible valid branches, and the branching will end there.

This process then repeats. If *A* was previously made and it was selected in the new run beginning at the root node, then the node to be branched from would now be *A*. Like in **Figure 3.2**, node <D> was originally created in a previous branching when <D,F> was created. In the same tree, the example is creating branches to <D,A,B,G>. <D> is not created twice. <D> is traversed to in the creation process rather than being recreated. The branching will attempt to branch out to any possible nodes that are not listed as being complete or any other nodes that can be created. If the branching goes to a node, and all of that node's children are listed as complete, then the node will be listed as complete and branching will start from the empty node again. Further creation of the tree will commence at the root node once more. This tree creation process stops when the root node becomes listed as complete.

To prevent highly popular topics blocking other less-common topics from probabilistically being added to the tree, the sidelines algorithm is utilized to block a topic from being added for a certain number of iterations after being traversed during construction. For example when "<A,C>" is created and the sideline is 1, for the next iteration of constructing the tree none of those topics will be considered. It will then be more possible for branches that contain less common topics sets like

"<D,F>" to be made. Another measure taken to make sure branches are different is that no two nodes may contain the same set of topics. So "<D,A,F>" cannot coexist in the same tree as "<A,F,D>". This increases the possibility of freshness between when the user will need to traverse the tree when changing topics. During that part of the recommendation process the user will go to the furthest possible node.

Logarithmic time decay is applied as a weight to the topic scores of articles used in the building of the tree. This will create a bias to more recent topics in tree building so that more recent articles appear during recommendations.

$$\boldsymbol{Decay_{(article)}} = (\mathbf{1})/\left(\mathbf{1} + \boldsymbol{log_{(RoD)}}\left(\boldsymbol{age_{(article)}} + \mathbf{1}\right)\right) \qquad \textbf{(3.2)}$$

The decay of the article's values in the system to be proposed is based on its age in minutes with respect to a rate of decay (RoD in **Equation 3.2**). If the rate of decay is two days, 2880 minutes, then the decay value of the article's topic values will be 1/2 of their initial value at the 2879-minute mark since its publishing. If the article is an hour old then the decay value will be 0.696.

3.5.2 Recommendation Experts

Experts are prediction models for determining the user's next set of topics. An expert exists on each node of the tree. As a user traverses this tree, an expert based on the current node will determine the next node the user will jump down in the contexts . For this example, consult **Figure 3.2**. If the user is on the starting node < >, the expert on that node will determine whether the user will move onto <C>, <D>, or <A>, based on a set of factors. These factors include, but are not limited to, the user's interaction with those topics previously, other similar user's interactions with those contexts, or trends in those contexts. Experts only provide their insights when they are active.

This can be when the user is in a branch that stems from the expert or when the user is sitting on the expert. Depending on the chosen number of experts influencing the user's recommendations, weights may also be applied to them . In the case of this project, the exclusive expert used is the one based on the node that the user is currently on during tree traversal. This can be a node like <A,C,F> or <D,F> in the tree above.

Expert models are ranking functions for determining the order that results, in this case news articles, will appear. In this project, the ranking function used is ItemRank. For this model, cosine similarity is used as a scoring function of the user to items, items to items, users to users, trends to items, users to nodes, trends to nodes, and other factors. Cosine Similarity is a method of measuring the similarity of two pieces of data by the angle between two vectors, rather than distance between them in the case of Euclidian-Distance Similarity, as a ratio from not related 0 (180 degrees) to related 1 (0 degrees). This reduces the impact of over-reliance on high-magnitude values that would be found in distance similarity . The equation of cosine similarity is as follows: where **X'** is the vector of the article being compared to, **X** could be a user vector, publishing trend vector, social media trend vector, or item vector, **i** is the index of an LDA topic in the vector, and **n** is the total number of LDA topics we consider in the system.

$$CosSim(X,X') = \frac{\sum_{i=1}^{n} X_i \times X'_i}{\sqrt{\sum_{i=1}^{n} X_i^2 \times \sum_{i=1}^{n} X'^2_i}}$$

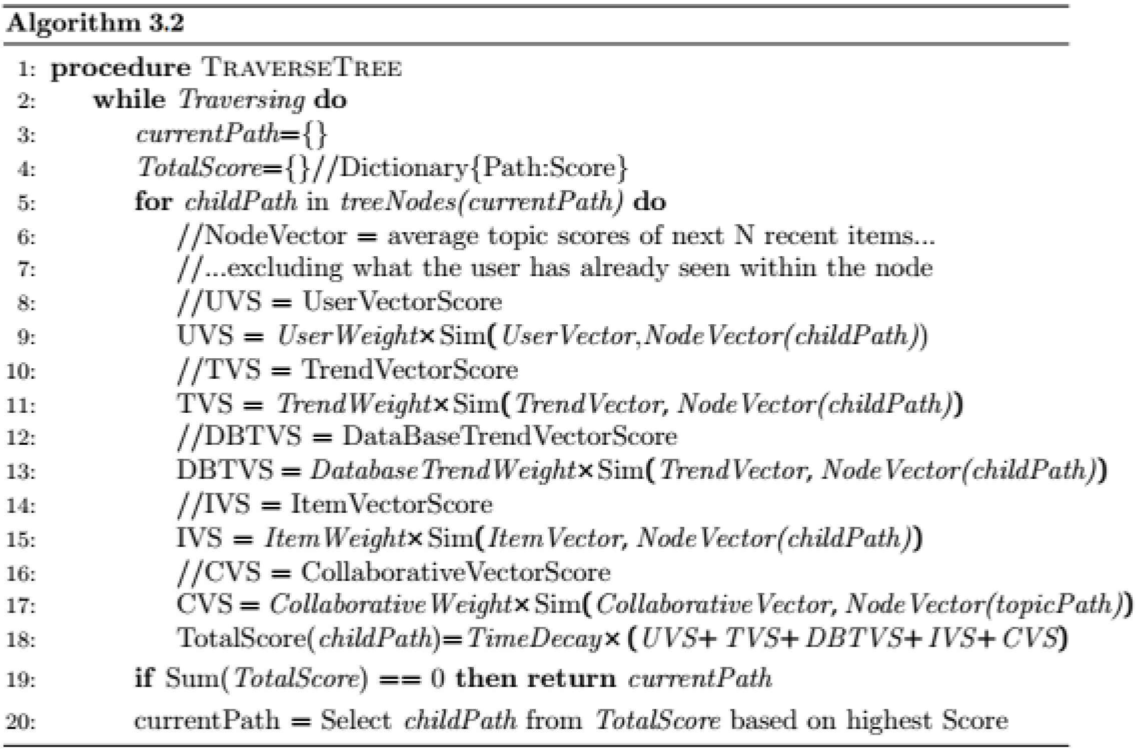
(3.3)

Each of these factors' influences are applied with a weight that determines their impact on deciding the score of the branch to go to. This is further explained in **Algorithm 3.2**.

The ItemRank model selected for this experiment involves selecting the result that meets the highest probability on the model it is based on. With respect to this implementation it will be referred to this way based on the sum of similarity measurements of the factors . If the total similarities of all incorporated factors for a topic are higher than the other calculated possible topic branches, then that is the topic set that the user will traverse to. For example, in comparing user vectors to nodes, the normalized LDA values of the possible nodes is used as a score. User vectors are a representation of the user through values on topics they have previously interacted with. Assume that the two nodes the user can traverse to are <D,A> or <D,F>. If topic set <D,F> has a similarity score of 0.8 to the user's representative vector and <D,A> has a similarity score of 0.65, respectively split 55% and 45% on user influence between the values, then <D,F> will be selected. This will be further covered in **3.6.1**.

3.5.3 Traversal

Algorithm 3. 2: Tree Traversal



Algorithm 3.2

```
procedure TraverseTree
    while Traversing do
        currentPath={}
        TotalScore={}//Dictionary{Path:Score}
        for childPath in treeNodes(currentPath) do
            //NodeVector = average topic scores of next N recent items...
            //...excluding what the user has already seen within the node
            //UVS = UserVectorScore
            UVS = UserWeight×Sim(UserVector,NodeVector(childPath))
            //TVS = TrendVectorScore
            TVS = TrendWeight×Sim(TrendVector, NodeVector(childPath))
            //DBTVS = DataBaseTrendVectorScore
            DBTVS = DatabaseTrendWeight×Sim(TrendVector, NodeVector(childPath))
            //IVS = ItemVectorScore
            IVS = ItemWeight×Sim(ItemVector, NodeVector(childPath))
            //CVS = CollaborativeVectorScore
            CVS = CollaborativeWeight×Sim(CollaborativeVector, NodeVector(topicPath))
            TotalScore(childPath)=TimeDecay× (UVS+ TVS+ DBTVS+ IVS+ CVS)
        if Sum(TotalScore) == 0 then return currentPath
        currentPath = Select childPath from TotalScore based on highest Score
```

The goal of traversal is to go to the furthest possible node down a path of the tree. The user begins at the empty node and then, based on the algorithm, traverses to the next node. Repeat this process for each subsequent node until no further node recommendations can be made through either the lack of a subsequent branch, or the lack of items in subsequent branches. The similarity calculation to go to a node is done through the similarity between user factors and the articles contained within the node.

When a change of topic is made, the set of topics on the node will be sidelined for a set number of topic changes. This means that nodes for consideration during the next traversal will exclude nodes that contain sidelined topics. Thus, their score is not considered in the possibilities. This benefits the system as it limits the influence of popular and repeating topics with respect to both database concentration of a certain topic and interactions by other users that may produce topic bias in the database . In the event that all topics are sidelined, topics are un-sidelined until a traversal is possible.

3.5.4 Comparison to Traditional VMM-Trees

The way VMMs are typically made is branch-by-branch as a user interacts with a node, with each expert providing a new recommendation that brings the user to a new node deeper in the tree. Each branching would subsequently add a topic to the topic set the user interacts with. This begins with no topic restriction at an empty node and every subsequent node would add a topic that a list of recommendations must meet. Incorporating LDA as a classifier for articles provides accurate jumps with regards to topic subsets in VMM techniques. Normally, within the context of news recommender system, this tree would be built and/or traversed as the user selects articles that interest them from a list of recommendations. The recommended article's highest LDA topic would determine the next topic added which set of topics the user will traverse to from the current node . Where the proposed algorithm diverges from the typical VMM is that the tree will need to be prebuilt, rather than built as the user interacts with the system.

Users must be given an entirely new group of articles when switching between item clusters. To ensure that items are as different as possible between topic changes, the number of topics that classify the similar articles in a cluster should be as many as reasonable. Articles can belong to many categories, so preferably if a user desires to change the subject matter, the change should be visible. Due to the need for articles to be more specific in topic, the user should not begin at a node in the tree defined by a single topic but begin from a node that contains a greater set of topics. Doing this will make subsequent recommended articles to be as similar to the previous one as possible and changes of topics as different as possible. The system delivers a one-by-one stream of results, and with the requirement to have subsequent results be like previous ones, the user will begin at a node as far down the tree as possible. For this to be possible and quick, the current system will build a global VMM model that is accessed by all users. This model is rebuilt on a frequent basis to account for changes of frequency of topics within the more recent set of articles. A traditional VMM would remove nodes that have expired over time . For the methodology in this thesis it is simpler to frequently rebuild the tree to keep it fresher with respect to more recent results.

3.6 The Recommender

The Recommender component is made up of a few modules: the user vectors, item-concentration vectors, trend vectors, collaborative vectors, the VMM-Tree, and ranking algorithm which is the expert applied to items as opposed to nodes. Each of these key modules will be thoroughly explained and the exchange of information between them will be clarified.

3.6.1 Creating the User Vector

For the user profile a time-based decay function is used in calculating each user's changing interests. The idea is that the user's most recent actions will give a better representation of the current interests on a per-user basis. As actions that the user made before a significant amount time can be considered negligible in current interests, only the most recent items within a time window are considered. As an item reaches the end of the window, its significance must approach zero. To apply time decay, a logarithmic decay function is used to calculate each article's weight so long as it falls within the window . The following equation is used to calculate the weight per article, where $age_{(action)}$ is the difference between the current time and the time the article was accessed, and *window* is the amount of time the actions are decayed for. In this experiment, age is measured in minutes from the current time for more precision in the decay value. The same equation is used for article decay (**Equation 3.2)**, but with decay being based on the recency of the action rather than the recency of the publish date of an article.

Each article is represented by a topic vector in the format of **Table 3.3**. The topic values per article are multiplied by the time-decay-weight of the article. As articles are iterated the total value per weighted topic is summed up into the user's new topic vector that represents their profile. The new user profile calculation is summarized in **Equation 3.4**. Variable *t* represents a time window, such as the past week, and *a* represents an article published within the time window *t*. If the article doesn't have a value for a topic, it is treated as a zero value.

$$userTopicScore_{topic} = \sum_{\forall a, a\ published\ within\ t} weight_a \times a_{topic} \quad \mathbf{(3.4)}$$

The topic scores for the user are then normalized as a percentage of the total of all the topic scores and stored as fractions of one. For example, assume the topic set for the user contains only topics *A, B, and C*. After the above calculation for the accessed articles topic *A* has a score of 0.23, topic *B* has a score of 0.3, and topic *C* has a score of 0.15. Their normalized scores out of the 0.68 total, rounded to the second decimal, in the user vector is: *A* = 0.338, *B* = 0.441, and C = 0.221.

The user profile is updated with these equations whenever a user decides to click on an item to read in the proposed system. User profiles are created and updated based on the user clicking to view the full article, as this displays that the user has shown interest in the article's contents. This reflects implicit feedback that can be acquired through basing user interest on whether the user finds a story interesting through their desire to interact with the article, while skipped stories could be a sign that the user is uninterested and could be used to influence the user profile further, their inclusion in profiling is proven to work in a traditional news aggregator system where a group of unrelated stories are stated to the user at once . This is a running-context user profile that is only considering interactions rather than other factors like reading time .

3.6.2 Creating the Item-Concentration Vector

This vector is developed from identifying the number of items within a potential node during traversal. The number of items that exist within a timespan, such as 2 days, in a node will determine the probability in going to a node. For example if the user is sitting at node <A> during traversal and the possible branches to go to is either <A,B> with 7 articles, <A,C> with 10 articles, or <A,D> with 4 articles, the total probability is out of 21. Topic B for the <A,B> branch will have a 7/21, or 0.333, score in the concentration vector, topic C for the <A,C> branch will have a 10/21, or 0.476 score, in the concentration vector, and topic D for the <A,D> branching vector will have a 4/21, or 0.19, score in the concentration vector. There is no need to also add values for the previously traversed topics to the vector, which in this case is only A, since the traversal has already gone over it. So the vector representing item-concentration in the format <B,C,D> will be represented by vector <0.333, 0.476, 0.19> respectively.

3.6.3 Trend Vector

The trend vector is an LDA vector that is created by running trending words extracted from twitter through the LDA model. The twitter API has a function to extract trending words. The trending words are placed into the database in the same format as any news article, with the trending words as the content. The Twitter-API trending word extractor is regularly run and after every N-th article processed by the classifier, the trending document, and thus the vector, are classified and updated.

3.6.4 Collaborative Vector

The Collaborative vector is based on commonly read items, using the StreamRec algorithm . It is made by finding users that have read the same recent items as the user being recommended, identifying the recent items read by these other users that the user to be recommended has not read, and creating a vector that is the sum of those items' attributes normalized . In the below equation, variable *i* is an index for an article the user has not read that was interacted with by another user within the time window. The sum of all of the vectors is calculated and then normalized for all of the topics within the vector to add up to 1.

$$\boldsymbol{CollaborativeVector} = \| \sum_{\boldsymbol{i}}^{\boldsymbol{window}} \left(\boldsymbol{articleVector}_{(\boldsymbol{i})} \right) \| \qquad \textbf{(3.5)}$$

This is conceptually meant to bring the user to the topics and articles that others have read by creating an overarching vector that represents that group of unseen items.

3.7 Recommendation Process

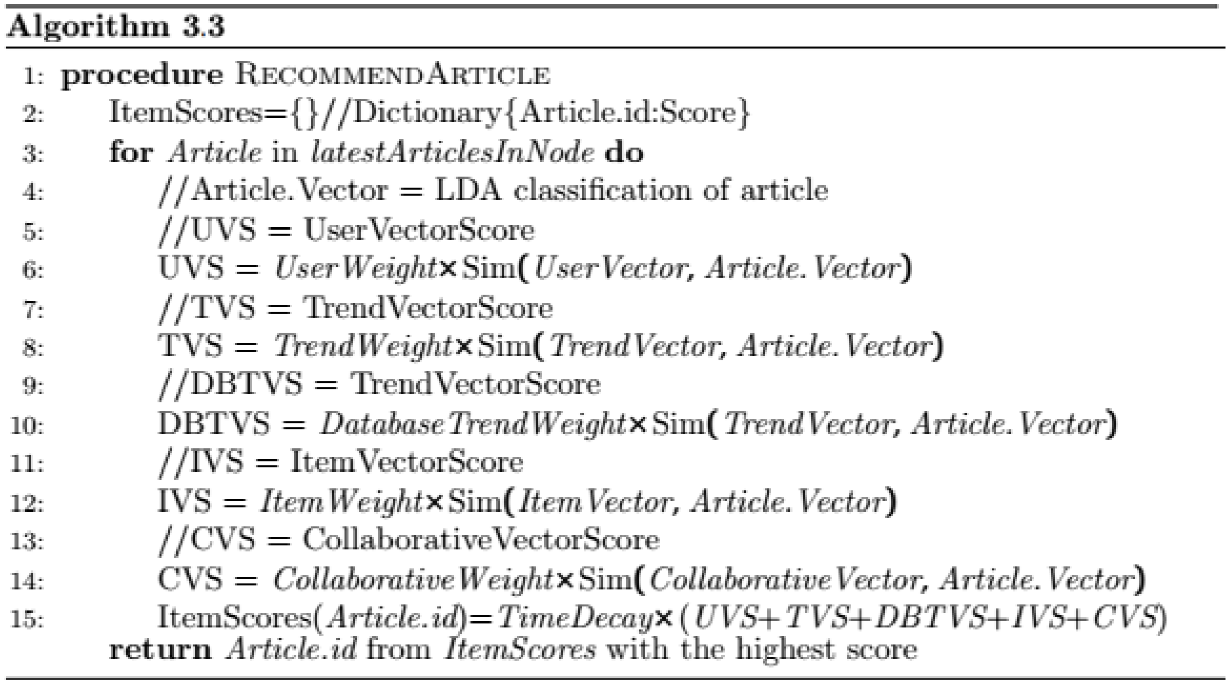
After traversing to the farthest possible node down the tree (not necessarily a leaf as sidelining may prevent this), an article is recommended from it. In this process ItemRank, the same expert used in traversal, will be used in providing to the user their next item. ItemRank will be used to rank the order of the next topic to append in the traversal of the tree and then it will also be used to rank the articles to give to the user.

With respect to the probability weights for ItemRank in both tree traversal and scoring the articles, the score is determined from six potential factors:

1. **User profile:** Vector representation of the user's topic interactions. This is calculated by applying time decay to the articles read by users and summing up topic values of the read articles. This is represented by **Equation 3.4**.
2. **Publishing Trends:** The more articles recently published in a topic set, the higher the score that set will have in the traversal process. Calculated with respect to the number of items available in each possible branch during traversal, and then the score is set as a total from each branch.
3. **Social Media Trends:** Trending key words are extracted from a social media API, such as Twitter, and those key words are sent through the same LDA model used to classify the articles in the article set. The Trends will then be represented by their own LDA scores in the database.
4. **Item similarity:** How similar the current item is to the next ones considered for recommendation. This is cosine similarity when looking for which item to bring up next as a recommendation. This is also comparing the current item to the top-N sum and normalized vectors in a desired path during tree traversal.
5. **Collaborative Users:** A general vector that is based on users who have read the same articles as the user, but the vector is constructed with articles the user has not interacted with. Created with **Equation 3.5**.

Each of the aforementioned vectors are added up with the desired weight of influence, not decay, applied to each one into a single vector. With respect to scoring articles: the user profile, trends, item-similarity, and collaborative user vectors are utilized. **Algorithm 3.3** shows an example with UserWeight (the user profile), TrendWeight (influence of trends), and ItemWeight (influence of similarity of the currently observed article to the next possible article to recommend). These weights act as fractions of 1 and add up to 1.

Algorithm 3. 3: Recommendation Scoring



Algorithm 3.3

```
procedure RecommendArticle
    ItemScores={}//Dictionary{Article.id:Score}
    for $Article$ in $latestArticlesInNode$ do
        //Article.Vector = LDA classification of article
        //UVS = UserVectorScore
        UVS = $UserWeight$ × Sim($UserVector, Article.Vector$)
        //TVS = TrendVectorScore
        TVS = $TrendWeight$ × Sim($TrendVector, Article.Vector$)
        //DBTVS = TrendVectorScore
        DBTVS = $DatabaseTrendWeight$ × Sim($TrendVector, Article.Vector$)
        //IVS = ItemVectorScore
        IVS = $ItemWeight$ × Sim($ItemVector, Article.Vector$)
        //CVS = CollaborativeVectorScore
        CVS = $CollaborativeWeight$ × Sim($CollaborativeVector, Article.Vector$)
        ItemScores($Article.id$)=$TimeDecay$ × ($UVS+TVS+DBTVS+IVS+CVS$)
    return $Article.id$ from $ItemScores$ with the highest score
```

At the end of traversal, the recommendations are scored and returned one-by-one to the user. **Algorithm 3.3** is how the program used user, trend, item, and collaborative factors in determining the article to recommend, which is done similarly to Node recommendation in **Algorithm 3.2**. The recommendation method used in this part for recommending articles is nearly the same as the method used for traversing the VMM tree. The exception is that rather than scoring possible topics, the scores are placed on each of the top-n latest articles that are defined by set of topics traversal had arrived at. The variables used for this calculation are: the user profile, trends, and item similarity. Comparisons are done using cosine similarity. The user profile, trends, and items all contain vectors that represent them. The profiles from users and trends are compared to the topic vectors representing each of the articles. When the user traverses beyond the first item in a set, the cosine similarity to that item will also be used in determining the next item. Each cosine similarity score is a value between zero and one, so the values of each of the six factors can be multiplied by its corresponding weight and then added up for a total similarity score that determines the ranking in one of the three algorithms. Similar to selecting the next topic in the set with the highest score, ItemRank selects the article with the highest score in this module.

In this project, there are three states that must be visible to the user at a time. The system must know these states to provide the user with some sort of feedback on how the system is processing the information. This also reflects the way the system runs. These states are:

1. **User's current state**: current set of topics and the file they are viewing.
2. **User's next item**: item to be presented and become the current one, should the user click to go to the next item in the current topic set.
3. **User's change item**: the item to be recommended if the user changes topics and the set of topics it belongs to. Taken from a fresh traversal where the topics in the current state are sidelined.

These three states can be better understood in the implementation in **Figure 4.1**. All three states must be visible at the same time, which means that any sort of change in the model will not take effect until the user changes the state of the current item. For the first current state, the user will traverse the VMM tree, and be recommended an item. This is the article they are recommended and will be used for user profile interactions. For the next item, assuming the user has not reached the last item in the topic set, the system will buffer the next item as per recommendation algorithm and provide it to the user. If the user decides to go to the next item, the buffered next will become the current item and the recommender will retrieve the next item to be provided through the ranking algorithm without traversal. The change item is selected through traversing the tree and recommending a new item as if the user has asked for an initial recommendation. This article and its topic set are also buffered in the database and provided to the user. If the user decides to change the item, the buffered change item is loaded, a new next item is determined by the Ranking algorithm, and a new traversal to a topic for the next change item is completed and buffered. It is important to understand that any interactions will not have an effect to the immediate next or changed item. As these items are buffered, it will not be until the following next or change recommendation that the ranking may change. The reason behind this design choice is for users to not have to constantly download new articles and to save time in the recommendation process. While this may seem to have detrimental impact on users immediately seeing the results of their interactions, buffered results can be previewed by users and changing recommendations and topics is a simple enough process that this concern should be mitigated.

Chapter 4

Experiment

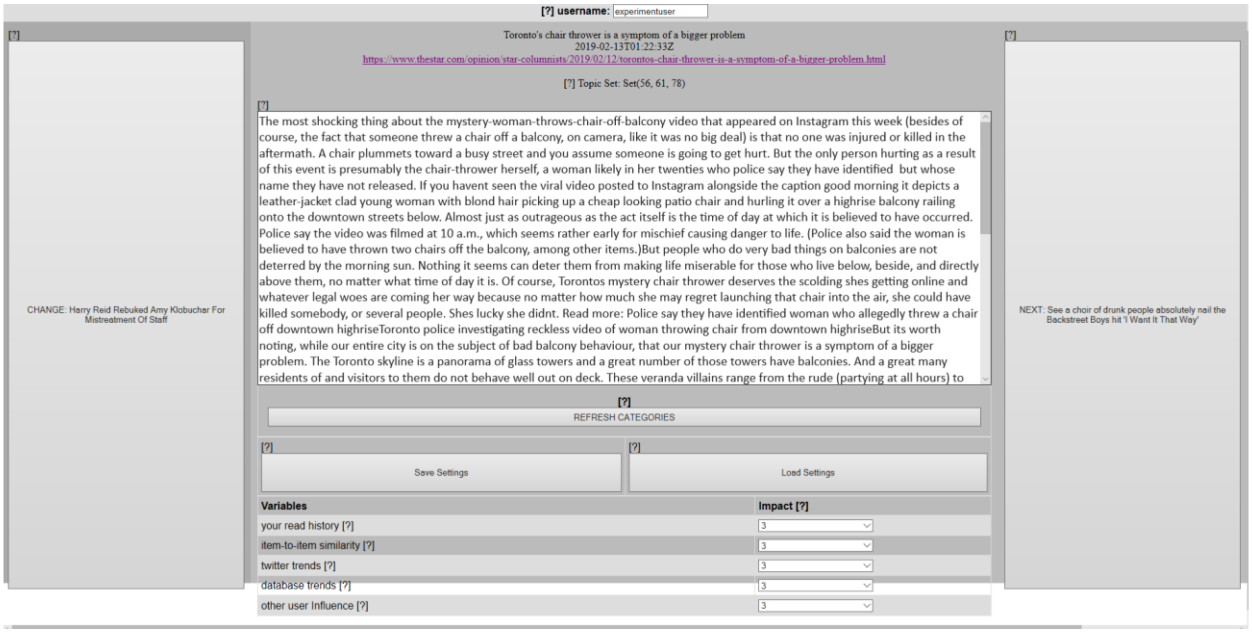
This chapter will focus on experiments that were performed in the proposed recommender system under a selection of variations. First, the focus will be on the design of the experiment. Here the variations of the experiment and the data used for evaluation will be discussed. Next, the focus will be on the collection of data in the experiment and how this data is aggregated for evaluation. The following section proposes the metrics to be used in evaluation and weighs their importance on the concluded design. Next the results will be weighed against the metrics and a conclusion will be made as to which variation of the system is most effective.

4.1 Experiment Design

The design of the experiment is to study the engagement of users under the various recommendation parameters within the proposed system. The idea is to make them feel more engaged in identifying what is effective to the user by identifying their preferred parameters, and understand what metric features, with respect to novelty, diversity, similarity, click rate, and so forth, reflect these opinions. With respect to feedback, this means both explicit and implicit feedback will be taken in the conclusions.

In explicitly collecting user feedback a survey is used. With respect to user understanding of the system, an accepted practice is to use John Brooke’s system usability scale . This has been applied previously to news recommender applications to test for the user’s understanding of both the application and the system. It is through this, that on top of the other opinions received, the conclusion can be made on whether the system itself is influencing the opinions of the recommendation methods built within them . Along with this survey, a questionnaire is provided regarding the feelings towards the system itself. These questions are used to determine the user's motivations behind their interactions with the system as well and identify the reasons behind why the system reacted the way it did. The **User Survey** can be viewed in the appendix**.**

4.1.1 Interface


[?]
[?]
CHANGE: Harry Reid Rebuked Amy Klobuchar For Mistreatment Of Staff
NEXT: See a choir of drunk people absolutely nail the Backstreet Boys hit 'I Want It That Way'
[?]
[?]
REFRESH CATEGORIES
[?]
Save Settings
[?]
Load Settings

[?] **username:** experimentuser

Toronto's chair thrower is a symptom of a bigger problem
2019-02-13T01:22:33Z
https://www.thestar.com/opinion/star-columnists/2019/02/12/torontos-chair-thrower-is-a-symptom-of-a-bigger-problem.html

[?] Topic Set: Set(56, 61, 78)

The most shocking thing about the mystery-woman-throws-chair-off-balcony video that appeared on Instagram this week (besides of course, the fact that someone threw a chair off a balcony, on camera, like it was no big deal) is that no one was injured or killed in the aftermath. A chair plummets toward a busy street and you assume someone is going to get hurt. But the only person hurting as a result of this event is presumably the chair-thrower herself, a woman likely in her twenties who police say they have identified but whose name they have not released. If you havent seen the viral video posted to Instagram alongside the caption good morning it depicts a leather-jacket clad young woman with blond hair picking up a cheap looking patio chair and hurling it over a highrise balcony railing onto the downtown streets below. Almost just as outrageous as the act itself is the time of day at which it is believed to have occurred. Police say the video was filmed at 10 a.m., which seems rather early for mischief causing danger to life. (Police also said the woman is believed to have thrown two chairs off the balcony, among other items.)But people who do very bad things on balconies are not deterred by the morning sun. Nothing it seems can deter them from making life miserable for those who live below, beside, and directly above them, no matter what time of day it is. Of course, Torontos mystery chair thrower deserves the scolding shes getting online and whatever legal woes are coming her way because no matter how much she may regret launching that chair into the air, she could have killed somebody, or several people. Shes lucky she didnt. Read more: Police say they have identified woman who allegedly threw a chair off downtown highriseToronto police investigating reckless video of woman throwing chair from downtown highriseBut its worth noting, while our entire city is on the subject of bad balcony behaviour, that our mystery chair thrower is a symptom of a bigger problem. The Toronto skyline is a panorama of glass towers and a great number of those towers have balconies. And a great many residents of and visitors to them do not behave well out on deck. These veranda villains range from the rude (partying at all hours) to

Variables	Impact [?]
your read history [?]	3
item-to-item similarity [?]	3
twitter trends [?]	3
database trends [?]	3
other user Influence [?]	3

Figure 4. 1: Interface

The user interface contains the standard set of required features with the additional user modifications to provide them some control over their results. The standard features are: a next button, a change topic button, and an article sample that can be clicked to reveal all of the article's text and count the article towards the user's interactions. **Figure 4.1** shows an example with the presented article having been interacted with. The additional customizable features of the interface are a way for a user to modify the way the main features react. These are modifiers for the user's read history, modifiers for trending news in social media trends and publishing trends, and the influence other users will have on recommended results. The impact is on a scale from 0 to 5 for these parameters and the weights for these parameters are calculated as a percentage of the total as in **Algorithm 3.3**.

4.1.2 Interactions and Instructions

For this experiment, a total of 17 users were tasked with interacting with the system. The users were mostly university students or peers who keep up with current events. All have regular access to devices that can run the application. During the experiment, all actions and settings are logged. These logged actions can then be used to provide information such as click rates, frequency of changes, and how these factors change over time based on the influences of the environment on the user's results. 5 of the users are tasked with setting their profiles to be only user-profile influenced as a baseline measure. Users interact with the system in at least 5 of the experiment days, for about 5 minutes per interaction session. If the article interests them, they were instructed to click on the middle text. They were also instructed to click change several times per session as well as next a few times per topic set in order to get an coherent idea of the topic set before changing to a new topic. When testing has completed, the results of user implicit and explicit preferences are collected and compared, along with the results they received during the experiment.

4.2 Dataset

The news dataset for recommendations is collected from a number of domains: The Weather Network, TSN, National Geographic, Space, Wired, Washington Post, Reuters, BBC, EOnline, Global News, Vox, Toronto Star, Live Science, Science Daily, NY Times, Digital Trends, CP24, CTVNews Toronto, Polygon, Engadget, Mashable, Huffington Post, Forbes, The Guardian, The

Globe and Mail, Sportsnet, Science News, RT, CBC, and CNN. Some of these domains were extracted with exact keyword matches in their respective RSS feeds, while others were extracted via categorical RSS searching. For example, searching for “sports news” in the “Canada region” will yield results for TSN without over-inundating the scrape with hundreds of very specific articles that site publishes daily. This also was done to eliminate over-populating the daily articles on apparently popular to publish subjects such as politics and sports, due to the global encompassment of the topic that would also create an over-bias to topics in that area in VMM-Tree building. Over the course of the experiment, from March 10th to March 29th 2019, 12838 unique articles were scraped from these sites. Prior to experimentation, 51907 articles were scraped and classified. All 64745 articles were available to be recommended to users during experimentation.

In the current scraper, an average of 642 unique articles are scraped daily. The number of articles scraped depended on the day, like there being less scrapes over weekends compared to weekdays, and if any over-encompassing event has occurred in the experiment period that is heavily written about. This group of sources also provided a good balance of subject matter between both local and world-wide news.

As a user interacts with the system, their interactions are logged to gain information on click rates and topic change rate. These pieces of information are logged with the user’s settings to know which of the users’ actions are done under which recommendation contexts. **Table 4.1** is an outline of how the information for user logging is stored.

Table 4. 1: User Logs Format

Field	Description
UID	User’s unique ID
Action	Start (initial click) /next /change /interact
Date	Time of article access
News ID	Unique Article URL (News ID)
Topics	Current Node’s topics

With respect to the rebuilding of the tree, the tree is remade every thirty minutes to insure an as up-to-date clustering of topics and files as possible. The tree is serialized and stored in the database. It is deserialized whenever necessary to traverse the tree to a node.

The User Profile is updated when an item is interacted with. As the experiment occurred over the course of 2 weeks and so not too many items would be used in determining the profile, the all items will be used within that window of time to update the user’s profile in a batch. For more precise relevancy calculations, the profiler runs based on minutes since the last read between articles. Time decay **Equation 3.2** is run with base 2,880, two days in minutes, to give a higher bias to more recently read articles and apply slow decay to articles afterwards.

4.3 Implementation

4.3.1 Scrape and LDA Training Set

In this section, any static parameters that need to be acknowledged will be covered. These include the LDA parameters, VMM-Tree parameters, and recommendation parameters. These parameters ultimately dictate how the system will run and greatly influence the outputs revealed to users of the system. LDA training determines how well defined the topics are both with respect to the number of words that define a topic and how well or loosely these words create the cohesiveness of a topic. The Tree parameters will determine how specific the topics may end up being in the nodes, and the minimum number of articles required for a node to be considered valid with respect to the number of articles it holds. Why the parameters are set the way they are is crucial in understanding the final evaluation of results.

4.3.2 Parameters: LDA and NER

The LDA model was trained from the sites on 20,000 articles spanning from September 2018 to December 2018. This model was used throughout the experiment for classification. Each article classified by the LDA model is represented by at most the top 10 topics that represent the article, with a minimum of 4 topics representing the article, where the total of the normalized values, with respect to the top 10, add up to at least 0.67. This typically results in articles being represented by about 6 topics. This was done as there was no definite definition of what normalized value of a topic would properly represent a article. For example, a topic value of 0.05 for one article may represent the article when observing the subject matter of the words that make up the topic. The topic may not have a relatively high score as there may be a single topic that dominates the normalized percentage, such as a topic-score of 0.8. This topic may still be relevant to the article but has a reduced score because of a single prevalent topic. In another article the defined topics could be more evenly distributed, such as the top 5 topics given scores of 0.15, so their added value would be 0.75. A value of 0.05 is given to a topic in this situation would likely indicate little to no relevance, as well as being partly in factor that LDA provides a topic score greater than zero, even if there is no relevancy between article and topic. By setting a percentage and a minimum number of topics, it guarantees that each article will have a decent number of topics in common with other articles as well as have a high probability that all topics that define the article are covered by the classification, while minimizing undesirable topics that do not accurately represent the article.

In creating the LDA model, the number of topics created through training is 120, the number of words per topic is 20, and the number of training iterations for the model is 50. These were numbers found and adjusted to provide both depth to the VMM tree, thus more specific topics per leaf node, as well as avoid training biases enough while providing well defined topics with articles that visually fall under them after extensive review and retraining.

In addition to training the LDA model Named Entity Recognizer (NER) words are identified in the article for promotion. These words are identified through the “Stanford Named Entity Recognizer”. It uses pre-defined models to recognize and list out named-entity keywords which are then used in the “Article-Dependent Named Entity Promoting” **Equation 3.1** that increases the frequency of these words appearing in their respective articles. This algorithm is also applied to the title and URL of the article. The non-letter characters are removed from both the title and URL. They are then appended to the article the same amount of times as the non-stop-word in the article with the highest frequency. This further promotes categorical words, like Science or Sports, and other titular words in the modelling and classification process .

4.3.3 Parameters: VMM-Tree

The VMM-Tree has a few parameters which are important to its construction. The minimum number of articles per topic set is proportional to the number of nodes created in the tree and how cohesive a set of topics is with regards to relevancy to their respective articles. If less articles are required to make up a node, there will be more potential for nodes defined by more topics to appear. However, a low article count per node risks not presenting enough consecutively relevant articles as the user iterates through the node.

However, this decrease in required items risks reducing the depth of the tree with many topics with low article counts being placed early in the tree. Too many items for the minimum number results in too few topics being used and reduces the database coverage. For this, a minimum article count of 5 was deemed sufficient, with at least 2 articles published in the last 4 days, and results in usually around 300 nodes present in the tree at a time.

Time decay is applied when constructing the tree. The rate of decay for articles when building the tree is 2 days represented by a 2880-minute base value in **equation 3.2**. This was found to give enough leverage to more recent articles and rapidly decrease the value to less than half for older articles. Articles selected for tree building were all articles in the past 30 days. This range was chosen as it provided a larger selection of articles for combinations of topics to appear from and provide a better guarantee of historical articles appearing for users if they decide to remain on a topic set for extended periods.

Ideally both topic relevancy and recency are considered in the construction of the tree. Requiring 2 articles to be published in the last 4 days under a certain topic combination aims to provide topic sets that have recently published information. Additionally, the requirement of 5 articles per node allows for historical articles under a topic set to be provided to the user if they suit the user's interests. There is likely a balance in the desired recency-relevancy tradeoff in the construction of the tree. This is in combination with the tendency for Markov Models to have flexibility in allowing for less accuracy based on the current set of information and providing space for combinations of topics which may be relevant in the near future .

4.3.4 Parameters: User Profile

The user profile is built from a set of articles they had interacted with. The articles selected for this were the last 100 articles interacted with by the users. Time decay of 10080 minutes, 7 days, is applied to the interactions of the articles. The 10080 minutes of decay also applies to the calculation of the collaborative vector between similar users. This was chosen as it would provide an appropriate short-term profile for the users within the days of the experiment.

4.3.5 Parameters: Recommendations

Time decay is the only application-set parameter for recommendations. Like the User Profile, the time decay applied to the similarity in **Algorithm 3.3** is based from 10080 minutes of decay.

4.3.6 System Architecture

Each module of the system was written in Scala, compiled and run using Scala Build Tool (SBT) both locally and on a DigitalOcean web server. LDA model creation and classification through LDA and NER was run on 2 Intel Core i7 cores @ 2.8GHz with 8GB RAM, with 50GB space on a virtual machine. For storage, all data was placed on a DigitalOcean server Elasticsearch database. The DigitalOcean server also hosted the website for the application. The specifications for the webserver are 6 Intel Xeon CPUs @2.2GHz with 16GB of RAM and 50GB disk space. Interface for users was created in HTML.

4.4 Results and Analysis

4.4.1 Analysis of system: Corpus

Defining topics should result in a corpus of topics that are as unique as possible. In creating distinct topics, the articles defined by them will have a coherent definition when those topics are applied to them.

By applying the Document-Dependent Named Entity Promoting algorithm in **Equation 3.1**. This would bring up not necessarily frequent terms in articles, but words that have meaning within the article. The result of implementing this should be a corpus of LDA topics that was built on meaningful words in collaboration with frequent ones. Thus, the overlap between topics should be minimized and the topics used to represent articles should be as unique as possible. Analyzing the exclusivity between topics ensures that when topics are combined within the tree, together the topics define a more unique set of items. If the defined LDA topics are not unique, then the combination of topics that define a node could be topics that are more-so the same thing, so the topics defining an article may be less than perceived.

The results of the trained corpus using the NER-LDA combination displayed an example of a potentially high-quality LDA model. Exclusivity of a model is the average percentage of words unique to topics within the model. The higher the exclusivity, the better defined the topics are. The exclusivity of the model is 89.25%, the lowest uniqueness of all topics being 55%, and 16 of 120 topics being 100% unique. This implies that on average 18 out of 20 words that define each topic only appear in that topic. This means that as topics are added to sets, they add value to the node. Topics with low exclusivity would result in limited to no added value if topics are added to a subset. While there is a high amount of uniqueness within the system, there is no guarantee that based on this stat that the topics are coherent. This means that the topics would be understandable by a human being that was reading them. In combination with the natural separation of articles through node clustering, the experiment subjects' survey results will be an assessment of whether sequential articles provided were related to one another. If articles within a topic set are relevant to one another, as judged by users, the topics should be cohesive. As LDA provides topic scores no matter what, topics would be assigned to articles, regardless of actual relevancy. In the event that articles are related, it will be reflected in user the user surveys.

4.4.2 Analysis of system: Tree

The analysis of the tree involves analyzing the nodes that the user will be interacting with. The objective of the system is to provide the user unique articles as they cross between topics. This objective was critical to giving the user fresh results on each jump, rather than providing transitions to a topic the user had recently interacted with. In this section the tree will be analyzed with respect to the user's interactions with the nodes of various sizes, followed by the analysis of the tree's structure.

4.4.2.1 Tree Analysis: User Interactions

In order to understand the significance of the size of these nodes, the size of the nodes traversed during the sessions must be considered. The average size of the nodes hit amongst all users is roughly 2.7 topics in the node being recommended across all sessions. The way this number was acquired was through the averaging the average size of topics provided to users during the experiment period. This average is a better representation of the average as some users had extensively or minimally changed within sessions, so they would have a different experience in the number of nodes they are exposed to. For example, a user with at least 20 changes in a session will have at least 20 topics in the sideline at any time, which would minimize the potential size of the topics they are interacting with. From this number, we can see that most nodes met for all users is mostly 3. Across all users, **Table 4.2** shows the percentage of node sizes presented to users and the percentage of interactions on those sizes. For example, if of 100 nodes recommended to all users: 35 were size 2, 45 were size 3, and 20 were size 4, then the percentage of recommendations would be 35%, 40%, and 20%, respectively. Also, if across all users: 100 interactions were in nodes of size 2, 150 in nodes of size 3, and 75 in nodes of size 4, then the percentage of interactions would be about 31%, 46%, and 23% for those respective sizes. Any omitted sizes were not recommended during the sessions. i.e. Nodes of size 1 were never recommended with in any user's session.

Table 4. 2: Tree Recommendation Rate to Interaction Rate Comparison per Node Size

Node Size (# of Topics)	% of Recommendations	% of Interactions
2	30.6%	29.2%
3	64%	67.1%
4	5.4%	3.7%

4.4.2.2 Tree Analysis: Nodes

The analysis of the nodes is important in determining whether there is a reasonable difference in the way articles are defined. If the nodes have an average low level of uniqueness between them with respect to the articles they contain, rather than defining using words that make up a corpus, it can be assured that the topics are not adequately defined. If there is a low average uniqueness, then the jumps between nodes that are supposed to be of different topics would be weak and that would be from either the topics not being well-defined enough and encompassing too many articles, or an overwhelming bias to certain topics that are defined in the multiple ways across topics. For example, if the model happened during an event that spanned a long time, it could bias the definition of topics.

For this analysis, the similarity between nodes on the same level of the tree will be compared. This is so that the parent nodes are not compared to their multiple children with the subsets of their articles. As jumps between children and parents will result in zero exclusivity and are impossible given the standards for sidelining in this experiment. This comparison in nodes will provide an idea of the potential difference in subject matter when jumping from node to node. The exclusivity is measured as the average percentage of shared articles between all nodes, including nodes with the same parent. It was measured this way as opposed to determining the number of articles that do not belong to other nodes, like in the corpus exclusivity evaluation, as sibling nodes that share topics tend to contain a subset of similar articles, and thus the exclusivity of the articles themselves would be greatly decreased. The intent of sidelining is to eliminate these sets of similar articles anyways. Likewise, it is not practical to eliminate nodes with similar parents, as it is difficult in this model whether node definitions like <A,B,C> and <C, D, E> contain a common parent. Topic C may be the parent both of these nodes stem from and it may not. If it is not, then C will have a smaller sideline and be brought forth sooner in the recommendation pool again. This is the reason for this slightly different variance in exclusivity definition. It will also give a better idea of how similar the jumps in exclusivity will be. The average min and average max exclusivity give an idea of the minimum uniqueness and maximum uniqueness between nodes when changing topics.

Table 4. 3: Node Exclusivity for Different Node Sizes in Latest Constructed Tree

Node Size	Exclusivity	Average Min Exclusivity	Average Max Exclusivity	# of Nodes on Latest Tree Constructed
2	89.6%	86.7%	100%	175
3	99.37	88.57%	100%	130
4	100%	100%	100%	5

It is evident that for the most unique experience, jumping between nodes of size 4 will likely result in absolutely no overlap in articles and thus topics as well under any circumstance. Nodes of size 3 also have a nearly perfect average exclusivity between them and will also probably result in unique experience. What is noteworthy, however, is that the average min exclusivity is 88.57%, which still brings up the potential for moving between topic sets that are similarly related. For nodes of size 2, the average similarity is not much greater than this, at 89.6% and the minimum is not much lower, at 86.7%. This is due to topic sets being more weakly-defined than the previous two sizes, so there are more articles per node and therefore more overlaps between node articles. This nearly 10% difference in exclusivity between size 2 and 3 nodes shows that nodes of size 3 are more well defined, though still provide the potential for switching to nodes of similar subject matter considering the 88.57% minimum exclusivity. The increased definition of topics, along with a similar capability to smoothly transition between topics could explain the increased interaction percentage in **Table 4.3** when compared to the seen nodes seen by users in topics of size 3, whereas the interactions decreased in comparison to the nodes seen of size 2. In nodes of size 4 the number of topics is results in a narrower list of articles that are likely very specific in the subject matter they are covering. This high-level of specification likely results in disinterest by the user when seeing articles that are too-related to one another and are too-narrowed down on what users are seeing based on their set preferences. So, nodes at this level provide too much of a specification when being traversed.

4.4.3 Analysis of Users

The goal of the survey is to gain insight into what users from the two groups who took part experienced with the application. Firstly, the surveys received will be examined whether or not their experience with the system reflected a usable system that did not impede on their involvement of the experiment. The next step is determining whether the recommended articles within the same topic set were cohesive in subject matter based on user feedback on relevance of subjects between articles. Finally, the users' perceived experience with the system will be taken and considered when being compared to statistical results obtained from program logs across all users. Through this an analysis can be made of user engagement and the reasons behind them with respect to the trends that are seen.

4.4.3.1 Usability

From 17 users, 8 surveys were submitted. With respect to usability, the John Brooke's System Usability Scale (SUS) was utilized in the survey to determine if there was any sort of difficulty that may have impeded the experience of the user. This is reflecting of general user confusion in how the interface works, whether the system gave the users issues and may have negatively influenced how they interacted with the system, and so forth. Generally, the average system gets a score of 70. Of the 8 users, 2 did not submit a result for the SUS survey portion.

The scores from the system usability study, from greatest to least, are: 95, 95, 95, 92.5, 80, and 75. The average of the SUS survey is 88.75, with none of the scores dropping below 70. There doesn’t appear to be any consistent concern based on any one of the scores from the completed SUS surveys. Another intent of the system was to be as minimal as possible and be as user-friendly within the confines of the experiment via a simple interface and buffering of articles so that the user wait time is minimal. The users were also provided with simple instructions in written and video format, which may have also mitigated any issues they may have had with the interface.

4.4.4 Survey Results Analysis: Baseline Users

Baseline users are users who were designated to use the system with their only parameter in their settings used to influence results being their read history. 4 of the surveys retrieved were from baseline users. With exception to the first two set of topics presented to them, as the system presents a topic and buffers the change before any interaction takes place, all topic sets presented to them are based on the topics that make up the articles they have interacted with, with no other factors.

In this section we will analyze the baseline user’s answers to the Application Results Survey. Here we aim to get an idea of the perception of the user’s results from the experimental period. Regarding having their results becoming more relevant to their reading tendencies, the average baseline user agreed that this is most of the time. If the subject matter they were interacting with was cohesive, this is an indication that there was consistency of articles presented on a day-to-day basis as well as the articles being properly defined within their set of topics. Likewise, when asked about the rate at which subject matter changed for them, it was between being a steady change and being mostly different. This is likely reflective of articles not necessarily only being defined by the topic set they are located in. In addition, given the typical average minimum exclusivity in most of the nodes interacted with being between 86.7% and 88.57%, it is possible that users jumped to nodes that contained similar subjects to the set of articles they have read, with consideration of the numerous topics. Regarding cohesiveness, the question of whether the articles in a set were relevant to one another, users had answered that 4 out of 5 articles as typically the number of articles they felt were relevant to one another within a set. Regarding the users in the baseline, all felt that the recommender provided results both with respect to their preferences as well as popular news. This is in part to the application having a popular news bias for topics in the initial articles given to the user, so those topics that are commonly written about would appear in the user's profile. Despite this, the consensus is that it provided results relevant to their interests by the end of the experiment.

4.4.5 Survey Results Analysis: Balanced Users

The other users being analyzed provided results under balanced user settings that included all of the possible settings at about an equal amount of influence.

The results of the application survey provided by the users will be taken into consideration in this section. Similarly, with baseline users, users in this section also felt that their interactions were becoming more relevant per new session most of the time, with the most active of the sample of users feeling it was becoming relevant every new session. These users also felt that changes brought mostly different results, so new topics provided new sets of articles. Of all users, all but one placed the corpus as having 4 of 5 articles being related to one another in a set. From this, it can be inferred that the corpus is coherent as subsequent articles would not have relevance if the words that comprised the topics did not make sense. When about the way users felt the system was catering to them, they felt that novel news and results based on past reading history were most prominent.

4.4.6 Log analysis

Log analysis will observe the trends in logs between baseline and balanced users. There were 17 users in the experiment in total. Of these, 5 were baseline users, 10 were identified as balanced users, and 2 fit in neither category. Links regarding how each set of users' surveys compare to their experiences shown in the logs will be observed and further analysis with how users interacted with the system involving topic distribution analysis will be observed.

4.4.6.1 Log Analysis: Interactions

In this section the differences between baseline and balanced users will be quantified. The main trend of interest is in the average interaction frequencies between users and topic sets the course of the 5 sessions they were assigned with. Each of the coloured lines in the following graphs represents a different session.

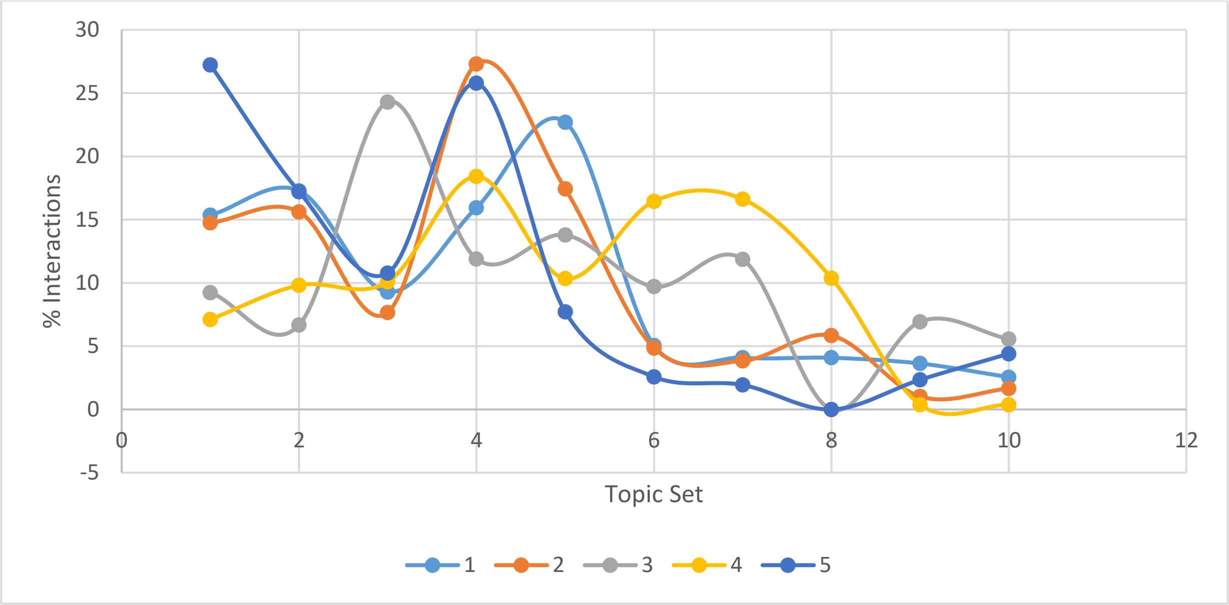


Figure 4. 2: Balanced User's Averaged Interactions per Session

Figure 4.2 shows the trends of balanced users over their first 5 sessions. Each line was calculated as the average percentage of interactions per topic set for each user for their first 10 interactions. For example, if there are two users, where one user has 100% of their interactions at topic set 1 and the other has 20% of their interactions at that point, it would average out at 60%. Not every user completed 10 topic sets per session. These percentages are calculated as being out of the total of all percentages. In this example, user 1 only interacted with topic set 1, so let us assume that user 2 interacted with topic set 2 the remaining 80% of their time. As user 1 did not interact on a second topic set , it will account for 40% of the interactions. In the chart above, the typical user had about 6 topic sets per session. So it should be noted that users who went beyond that would influence the latter topic sets to increase in value, but also decrease earlier topic sets as their percentage distribution would be lower for them. For users who went beyond 10 topic sets, their statistics were normalized to their first 10 topic sets. If a user had 10% of their topic sets on topic set 9, but 80% of their interactions were in the first 10 topic sets, then that 10% is normalized out of 80% and its normalized worth is 10%/80%, which is 12.5%. The 12.5% is the number which would be the user’s influence on the chart.

From **Figure 4.2** it is evident is that balanced users typically had a notable peak in interest around topic set 4, afterwards interest in interacting with the articles presented rapidly declined. What is evident is that for these readers, there was always a single topic in the session that acquired most of their attention and afterwards their interest declined. There are a few possibilities regarding this. There is a balance between the user’s profile, other similar users, and trends in social media and publishing trends. The earlier articles likely pique their interest less and the articles afterwards are not catered to the users’ interests enough afterwards to maintain user interaction. The balanced surveys indicated that users were split between being brought novel news and reading history. To further this conclusion, an observation on topic distributions for these users will need to be made.

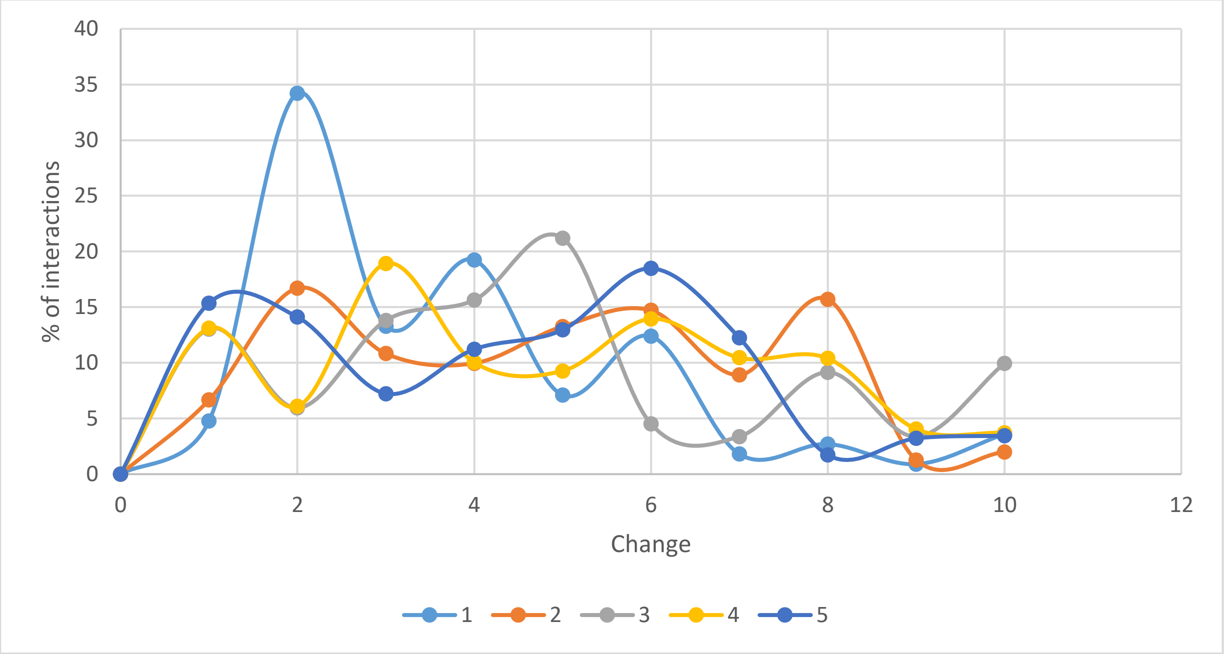


Figure 4. 3: Baseline User's Averaged Interactions per Session

Figure 4.3 reflects the trends of baseline users. It is calculated in the same manner as **Figure 4.2**. Topics presented to these users have more of an equal distribution in interest of subject matter. The first session is similar to all other users, in the sense that there is a peak in topic interest on the second topic set, followed by a rapid decline. This is similar to balanced users' trend. They would expose themselves to the popular topic of the day in earlier topic sets, which would change to a more novel topic based on their interests, thus the peak and rapid decline in reader interest. For baseline users, they are afterwards not as susceptible to popular topics on the changing days, rather all articles they are exposed to day-to-day, respecting the 2-day logarithmic topic decay, would have a say in what is recommended in their next sessions. This results in a more linear distribution of interactions with articles between topic sets. **Figure 4.4** shows the aggregated percentage of the above chart. After the first session, the interactions between topic sets becomes almost linear and very similar on a day-to-day basis. A maintained level of interest from the beginning to the end of a session is a good indication of mitigating a cold-start issue as well. This trend is in opposition to having topics that peak user interest at later points of the recommendation process. This is not beneficial in the long-term. By recommending similar items to users over a long period of time, user retention may suffer. This is because user interests could change over a long period of time or constant exposure to particular topics may wane their interest in them. Ways to introduce more randomness to recommendations while maintaining this session-to-session trend is to be desired .

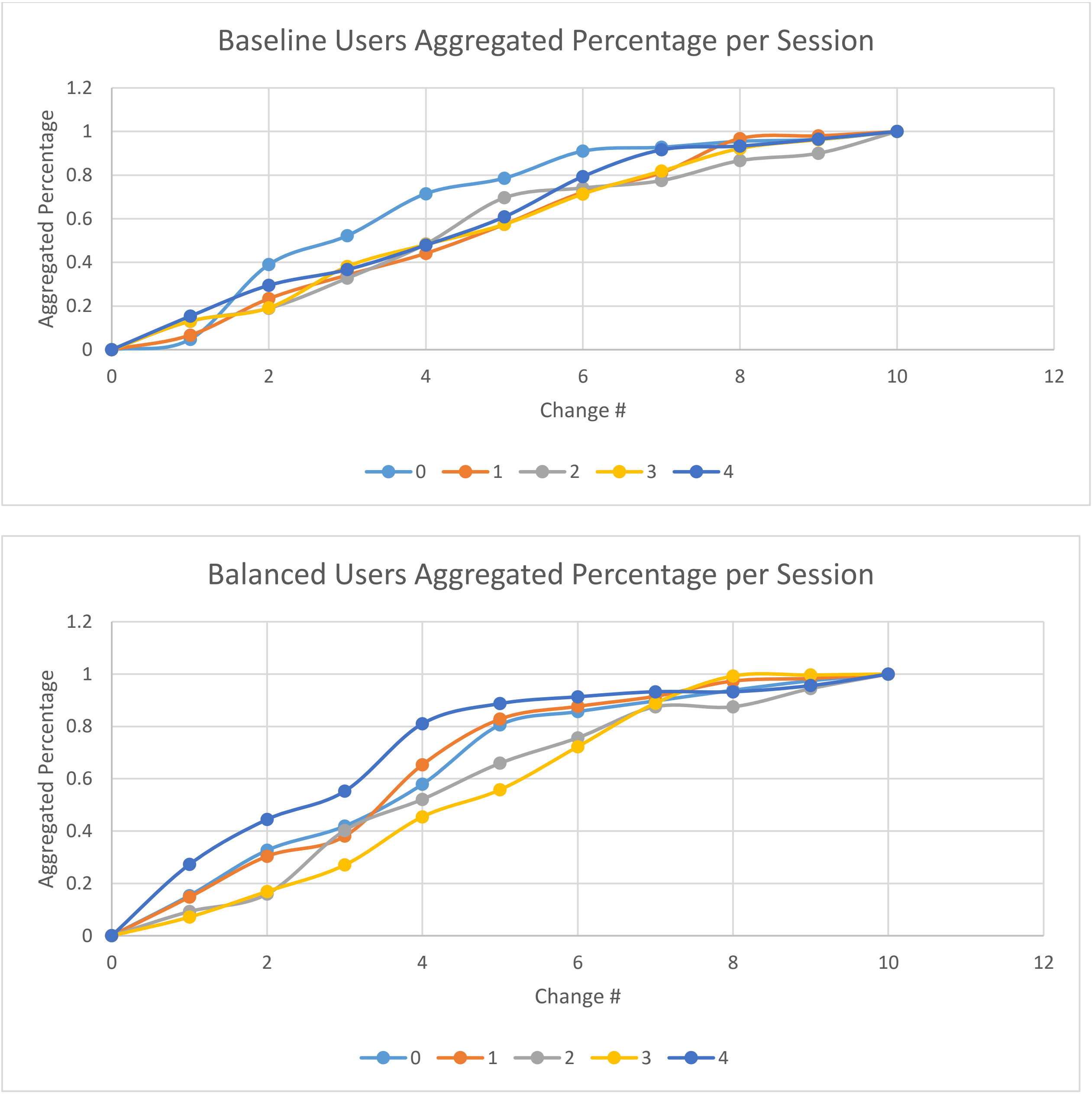


Figure 4. 4: Aggregated Interactions per Session

There is a noticeable difference between this trend in comparison to the aggregated percentage trends in the chart provided by balanced users, who tend to have a single topic which peaks their interest per session. In the **Figure 4.4** Baseline chart, after the first session the trend of interactions almost linear in comparison to the Balanced chart, which tends to have more interactions at particular points than the Baseline chart. The interaction trends will be better observed when averaged out across sessions.

The averages of all of the daily percentages will be used to show the trend in peaking popularity for topics amongst balanced users against the more equal nature of the baseline users.

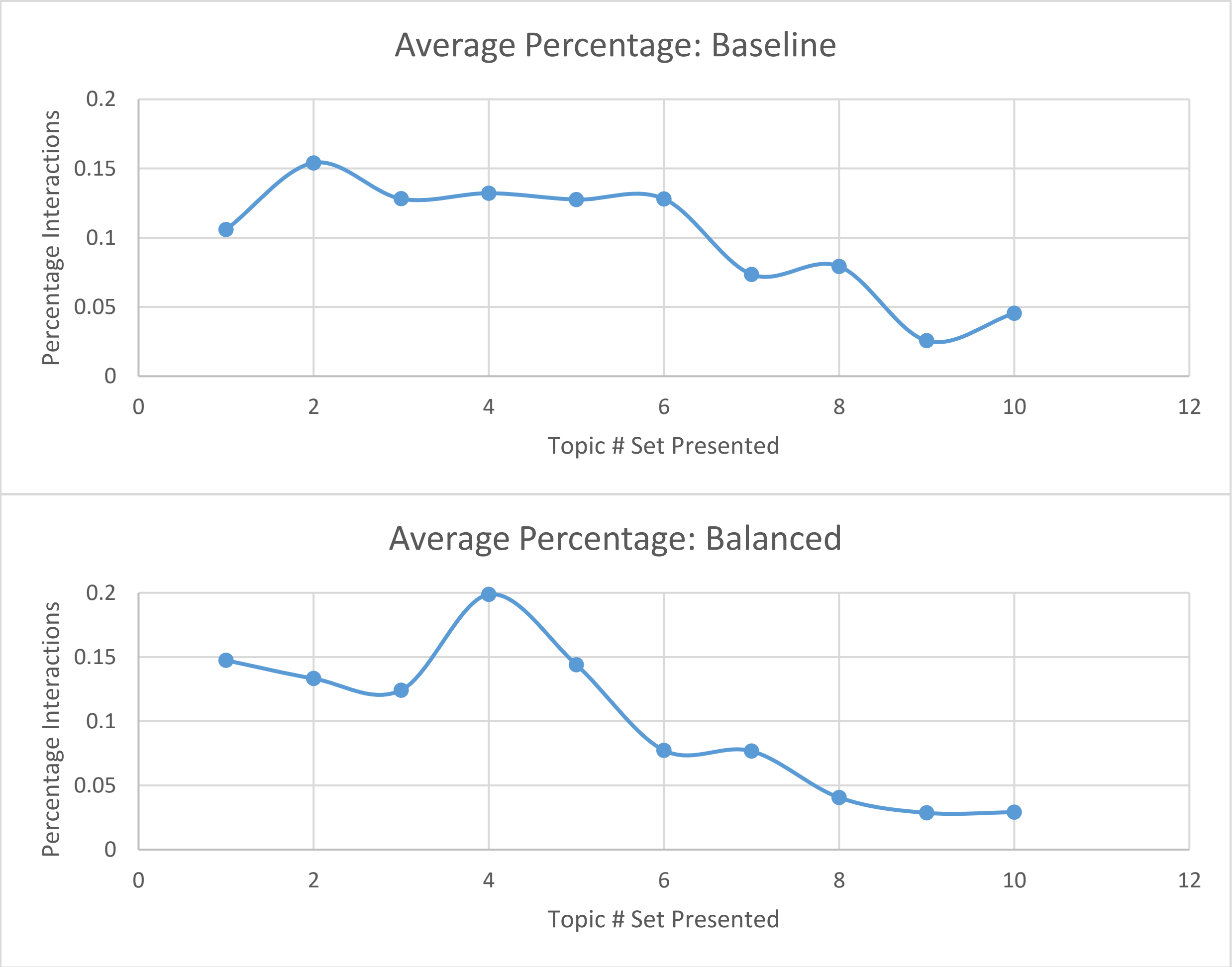


Figure 4. 5: Average % Interactions Across All Sessions

Taking into consideration that the typical user interacted per session with 6 topic sets, we will observe the differences between the two above graphs showing the averaged interaction percentages between them. For balanced users, there is a consistent single peak in reading activity that shows a roughly 66% increase in interaction peak in interest for the fourth topic interacted with followed by a rapid drop to about the same level of interest as before. This is then followed by an even more significant drop in interactions by the readers. With respect to baseline users, there is an immediate peak, followed by a roughly 20% drop in interest, which remains consistent up until the 7th topic, which falls beyond the typical user. This initial peak is reflective of the first sessions users had undertaken. As shown in **Figure 4.5**'s Baseline chart, the second set of topics the users were initially exposed to contains double the interactions in comparison to any of the remainder of the 5 sessions.

4.4.6.2 Topic Interactions: Distributions

Considering the interactions with the trees, it is important to identify the distribution of interactions between topics. The Gini index will be used for this calculation. The Gini index is used to measure the inequality of distributions. In this case it will be used for showing the frequencies of certain topics appearing more often than others. This will be done with respect to a topic appearing in a node when the user arrives at the topic. For example a user is placed on node <A,B,C>, then the next time they are placed on node <C, D>. The frequency at this point for the user of going on topic A is 1, B is 1, C is 2, and D is 1. The goal of the Gini index is not on a per-user basis, but collectively have topics evenly interacted with in the system. The intent is to have as many possible topics included in the recommendation process as possible. The more topics considered during recommendations, the less wasted effort there is in relation to classification. If there is an equal spread amongst topic interactions, it would imply that the system is effectively catering to all kinds of users.

We aggregate the results of these interactions for balanced and baseline users. A Gini index of 0 indicates completely equal distribution of topics amongst all users, while an index of 1 indicates all of the interactions are limited to a single topic. A 0.5 Gini implies that 50% of the topics make up 25% of the recommendations. While there will be a set of topics that do get the most attention, most of the topics will fall within reasonable exposure to one another. A score lower than 0.5 would indicate less bias towards certain topics in the system. A score above 0.5 would indicate more bias across all users to particular topics. The higher the score above 0.5, the less topics there are taking most of the recommendations and the more wasted effort and bias there is in the system.

Considering the limits of the experiment with regards to number of users and limited time, the Gini index will be observed from 4 perspectives: The Gini index including all topics and all interactions, using only topics interacted with, all topics with first 6 interactions, and only interacted topics in first 6 interactions. This is due to the limitations of the number of users in the number of days they interacted with the application and the limited time window which may not have included all of the possible topics. Observing this way, comparisons can be made between the indexes and assumptions of any possible missing information from the experiment’s limitations can be made.

Table 4. 4: Gini Index Comparing User Sets to Topic Sets

Userbase	All Topics	Recommended Topics Only	First 6 Topic sets; All Topics	First 6 Topic sets; Recommended Topics Only
Balanced	0.511	0.485	0.571	0.524
Baseline	0.531	0.455	0.582	0.415
All Users	0.479	0.463	0.526	0.492

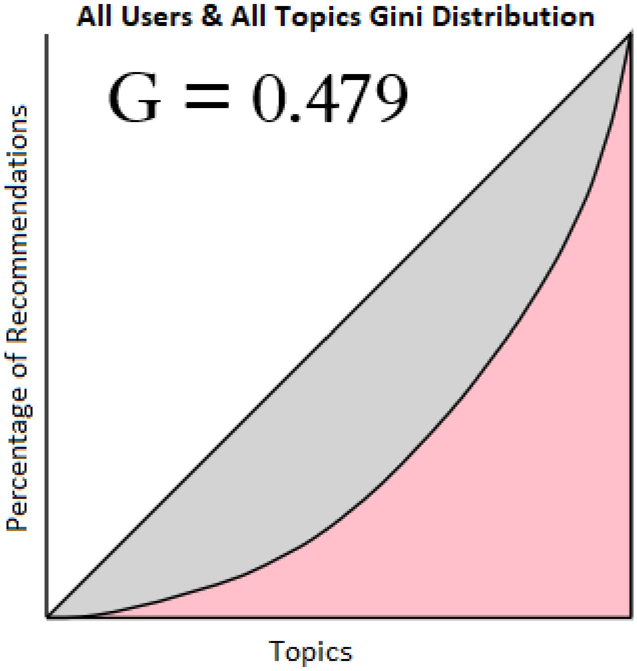


Figure 4. 6: Gini Index All Users

Observations will be made on the Gini indexes in **Table 4.4**. First an observation will be made on the balanced userbase. A 0.511 Gini indicates that across all balanced users there are certain topics that appear more often than others. There are approximately 50% of the possible topics encompass 25% of the data. These less likely to be recommended topics are likely due to not being as popular in trending data within the limited time window as balanced data would have that as an influencer. With more time, more topics would likely appear in both publishing trends and social media trends from twitter. The ItemRank model used is a popularity model that favours articles of higher scores that originate from trending factors. This factor in bias towards popular topics becomes apparent when looking at only the first 6 topic sets recommended to each user with respect to all possible topics. The 0.571 Gini in strictly using the first 6 topic sets, the recommended topics users would typically get on an average session, indicates that there is a noticeable preference of the algorithm to recommend a certain set of topics before sidelining would bring forth other recommendations.

Now a reflection of Gini on the baseline users, users who were only influenced by their profiles, will be made. Again the Gini index is only slightly above 0.5, which implies that for the most part, though uneven, the vast majority of topics are provided to the user. When looking at the first 6 topic sets only per session, the Gini raises similarity as the Balanced users to a Gini of 0.582. What is substantially different here is that the Gini for strictly recommended topics in the first 6 recommended topic sets is 0.415. This is a difference of -0.167 in Gini in comparison to the Balanced user drop from 0.571 to 0.524, which is -0.047. This Gini index difference in the recommendation of topics can be reflected with using the user interaction samples in **Appendix A**. While both balanced and baseline users interacted with a large set of topics in these charts, baseline users consistently interacted with certain topics between sessions. Balanced users would interact with less consistency between sessions, it more often occurred that an interacted topic would reappear in a later session or never. Likewise this trend in equal interactions and recommendations can be seen in **Figure 4.5**'s "Average Percentage: Baseline" chart. During the first 6 topic sets the users typically had an equal balance in interactivity before the number of interactions dropped. As baseline users interacted with these first 6 topic sets more, the variety in these sets would drop as their profiles updated. The even interactions between these set changes would result in a more balanced Gini index when only observing them and increase the Gini when observing other topics as they are less present in the user's profile.

The balanced users typically were more usually provided a set of recommendation of topics that were not previously provided to them. As they interacted with the system, they received results that were also not directly relevant to only their interests. Across all users, each user was given more of a variety of results and thus had a larger variety of interactions. Balanced users, like the baseline users, do still heavily interact with certain topics more than others when provided with them over multiple sessions, which can be reflected on in **Appendix A.1**, **A.2**, **and A.3**. The difference in interactivity is the less frequent emersion of these topics for balanced users in comparison to the baseline users who are given the topics they have interacted with more than others more frequently across sessions.

When combining all users, the Gini coefficients become distributed, for all topics, 0.479 as shown in **Table 4.4** and 0.526 when considering only the first 6 topic set changes per session. This drop in Gini could be an indication that the system would have a more even distribution of topics with additional users. It would be interesting to see the effects on the Gini index on both user sets with substantially more users in either. That putting together the user sets results in a lower index may also hint at the potential in eventually developing a hybrid system that alternates between pure user and trending topics in order to provide a broader spectrum of recommendations without having to go beyond the first six topic sets and strictly relying of the prolonged sidelining of prominent topics as done in this system.

4.5 Summary

This Chapter explained the design of the experiment through the interface and user instructions, the news websites used to populate the database, and the parameters of the system with respect to training the LDA model and building the VMM-Tree. The system was analyzed as well as a comparison between a set of users using balanced settings against a set using setting based strictly on their interactions. In analyzing interactions with the trees, with respect to LDA models in news recommendation systems, the conclusion can be made that a depth of 3 topics representing a node would be optimal in providing engaging articles to users. The users with results based on their profiles tend to have more balanced engagement throughout the topics being recommended to them before interest wanes, whereas balanced users have a rapidly declining rate of interest in the articles provided, with exception to a notable spike in interest for a single topic set per run. With respect to the balanced users, it was found that results reflected what is novel and their reading history. The users who had their profile as the only factor found that the system was narrowed down onto their interests after a few sessions. In reflection with averaged Gini indexes, baseline users displayed a slightly narrower recommendation of topics, though a more even distribution of the top topics interacted with than balanced users, who tended to interact more with similar topics. When all users are combined under a single index, there is a noticeable tip towards there being a more even distribution of topics. This likely shows that if a hybrid of the two systems were recommended, the system could potentially provide to users an experience that delivers relevant and novel news, without sacrificing what interests the users the most. A hybrid of the two systems would consist of dynamic adjustments to the variables that differentiate balanced users who are more exposed to novel articles, and baseline users who are given articles that specifically target their interests.

4.6 Threats to validity

The concerns for validity revolve around the conclusiveness of the two groups of users. The main concern is the timeframe of the experiment. With 17 users using the application in 5 days out of a time window of 20 days, there is concern regarding the kinds of topics users at certain points were exposed to as opposed to others who committed to their 5 days of interactions at a different time. This distribution of data likely has an influence on the overall topics that users were exposed to. If the experiment occurred over the intended 10-day period, it could also be said that potentially not enough topics would have had the opportunity of becoming prominent in tree building, thus the recommendation process for trending topics. If there were substantially more users accessing the application across the experiment period, more concrete data on topic distributions could be made. Despite this, all users displayed similar interaction patterns with regards to their designated user group. Likewise, the user group for baseline users was not large, as it was 5 users.

The application itself also is a threat to validity of the experiment. The sites selected for scraping were designated to be localized to the news users in the locale would be exposed to, so the result may not yield similar results if the sites were chosen based on a broader, global approach to news sources. Finally, the stream-based nature of the news topics being recommended to users means this approach may not be as well reflected when used in a non-streamed environment.

Chapter 5

Conclusion

5.1 Conclusions

In this work, our goal is to provide a simple system with a streamlined news feed using a general VMM-Tree to cater to all users in the system. Based on how the tree is built and how users interacted with the system, the system's goal was to provide the users with engaging and relevant news, without sacrificing factors like novelty and serendipitous results.

Sidelining was a key component used in both tree creation and topic recommendation to create a larger variety of nodes that users would interact with on a per-session basis and make each new change in topic to a unique new set of results that may interest them. The ItemRank model in this experiment used a set of variables that were static, with users either dedicating themselves to only use their profiles or balance their preferences against their reading history, other similar users, and trending topics in twitter and the scraped articles. As users interacted with the system, their profiles were updated with the attributes of the articles they have read. This would further cater to their interests. Implementing popular-topic vectors provided a good starting point for users to be provided their stream of news.

In this Experiment, the system was interacted with by two sets of users: baseline users, users only using their profiles to get their results, and balanced users, who used an even combination of factors to get their recommendations. As sessions passed, the baseline users were eventually provided results that were strictly catered to their previous interactions and their interactions were exposed to a more limited set of topics. The balanced users felt the effects of their profiles, popularity factors, and collaborative vectors and experienced more range in their recommendations as sessions progressed. The former provided more equal interactions with the topics they were given, whereas the latter would regularly single out a topic of their interest for reading during their sessions. In both cases, however, users were provided with topics of their interests and their interactions with their recommendations reflected that.

With respect to balanced users, the results have shown a decline in interactions as sessions lengthened, with exception to a single spike in topic interest. The baseline users showed this with respect to their first session, but future sessions displayed a balanced interaction amongst topics presented to them. The baseline users displayed a leaning towards a maintained level of interactivity that is desired for user retention. However, the baseline users leaned towards being recommended a specific subset of topics, which threaten long-term user retention from lack of variability assuming the users do not have prolonged session where sidelining would eventually bring more serendipitous results. Despite this, between the user bases and amongst all users, the system provides the potential to have less wasted effort with regards to the distribution of topics recommended amongst all users.

The system provided news catered to the two userbase readers’ interests despite the VMM-Tree being dynamically built regularly for a large group of users. The tree also provided two noticeably different trends in interactions between the two subsets of users. From these comparisons of users, we can come to a few conclusions to the contributions this experiment provided to future research:

1. Generalized VMM-Tree models can be used to provide catered news experiences to a locale without compromising user experience with respect to providing topical news, news relevant to the user’s interests, or novelty. In combination with sidelining it can also provide the potential for novel topic combinations, despite the tree being built based strictly

from Database publishing trends. Furthermore, though the userbase and the time-span of the experiment was short, the system also displayed a potential to reduce wasted effort with regards to topics being assigned to articles and those topics being recommended to the users via the ItemRank and sidelining techniques.

2. While VMMs have previously been done on a per-user basis, the tests have shown that using a generalized model can be influenced by user-based variables and user profiles to provide a catered experience. This significantly simplifies the process of recommending as it completely removes the need to maintain VMM-trees on a per-user basis. Maintaining the system is limited to user profiles and the environmental variables.

3. The baseline-users displayed balanced-user behavior for the first session, where a single set of topics greatly grabbed their interest in comparison to the others and quickly balanced out for latter sessions. This is in part due to: the system recommending series of articles defined by 3 or more of the same topics nearly 70% of the time (**Table 4. 2**) and the attempt to reduce the number of irrelevant LDA topics per article, and the 4 to 10 topic flexibility each article had in classification which would provide a better definition of the user's preferences beyond the node's topic set. From these factors the effects of narrowing down the user's interests were quick and after a couple of sessions the user's experience was significantly more catered. This is important in mitigating the cold start problem as soon as possible with regards to the user's profile.

5.2 Future Work

While the system has proven it can be effective in efficiently providing catered experiences to users, further exploring of the effectiveness of the system in comparison to other models and dynamics would be ideal in identifying the optimal approach to implementing such a system. The system is proven to work with respect to a standard ItemRank model, but it would be ideal to compare the system under the influence of other commonly utilized approaches as well.

The first of the other approaches would be to see the effectiveness of the system using a Bayesian Personalized Ranking (BPR) model. This model is the opposite of the ItemRank model. The goal of this is to introduce fresher items to a user by providing them with recommendations that do not fall within their current interactions. This is done by selecting the topic branch with the lowest score after the scores for sequential nodes are calculated. For example, if topic set <D,F> has a similarity score of 0.8 and <D,A> has a similarity score of 0.65; <D,A> will be selected. The intent of such a system would be to introduce the users to sets articles that are novel to their interests. This would broaden the number of topics that users are exposed to in comparison to the ItemRank model, while maintaining the same approach in user profiling.

The other approach would be a probabilistic approach to providing topics to users. This method involves using the scores calculated to determine the probability of a branch occurring . The branch is then randomly selected with the higher chance of selection given to those with a higher score and those with a lower score respectively have a lower chance of being selected. For example, if topic set <D,F> has a similarity score of 0.8 and <D,A> has a similarity score of 0.65; <D,F> will have a 55.17% chance of being selected and <D,A> will have a 44.83% chance of being selected. This works similarly to the implemented ItemRank system, but introduces an aspect of serendipitous results to topics that interest them as well as the chance of being recommended novel topics.

It would also be beneficial to compare this project to one using tree models built on Naïve Bayes. Through doing this it would be possible to compare the significance of the novelty in this model and highlight the importance of novel and serendipitous results where a Naïve Bayes tree would focus on accuracy.

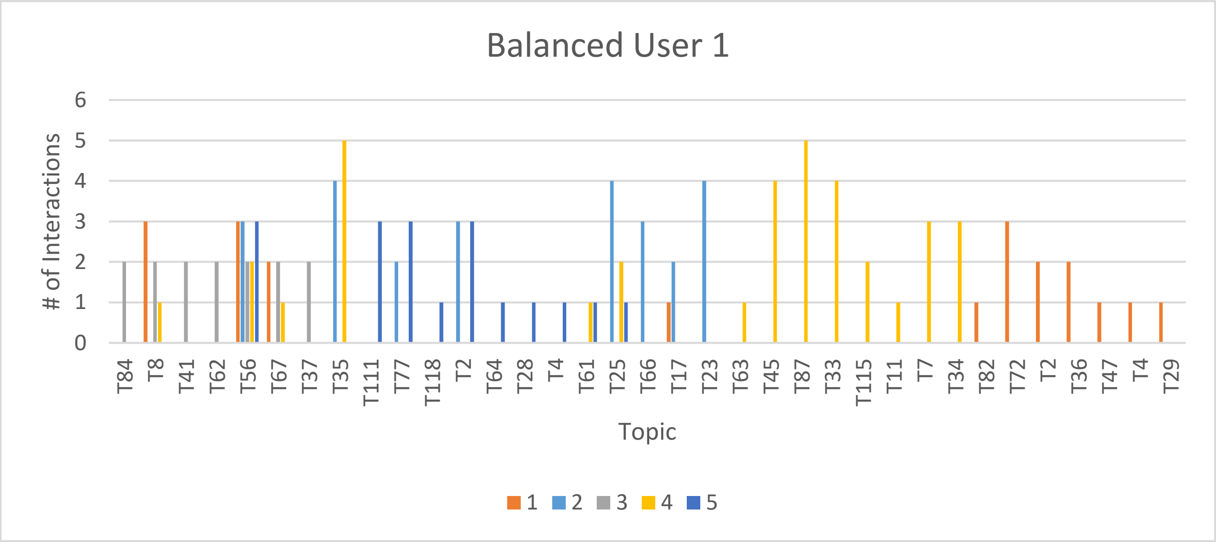
With the comparison of ItemRank, BPR, and Probabilistic results, an optimal approach to recommendations can be made. In addition to this, dynamic variables can then be implemented on top of the ideal model. This is with respect to the content-based and collaborative filtering variables that influence the baseline and balanced users. Much like VMM-Trees have been used previously, dedicated experts can be used to dynamically optimize which factors regarding user profiles, other users, social media trends, and publishing trends will influence the users in each step of the tree. Through this a method should be identified to provide users with novel information that engages them, while maintaining a balance in the stream of topics when the set of articles recommended to users is strictly based on their interests.

Finally, it would be of interest to see the effects of this system in a larger userbase to identify the limits of it. We have identified the potential in optimizing metrics such as Gini, though it would be beneficial towards further evaluation if conclusive results were made in that metric's regard as well as in reflection of BPR and Probabilistic techniques.

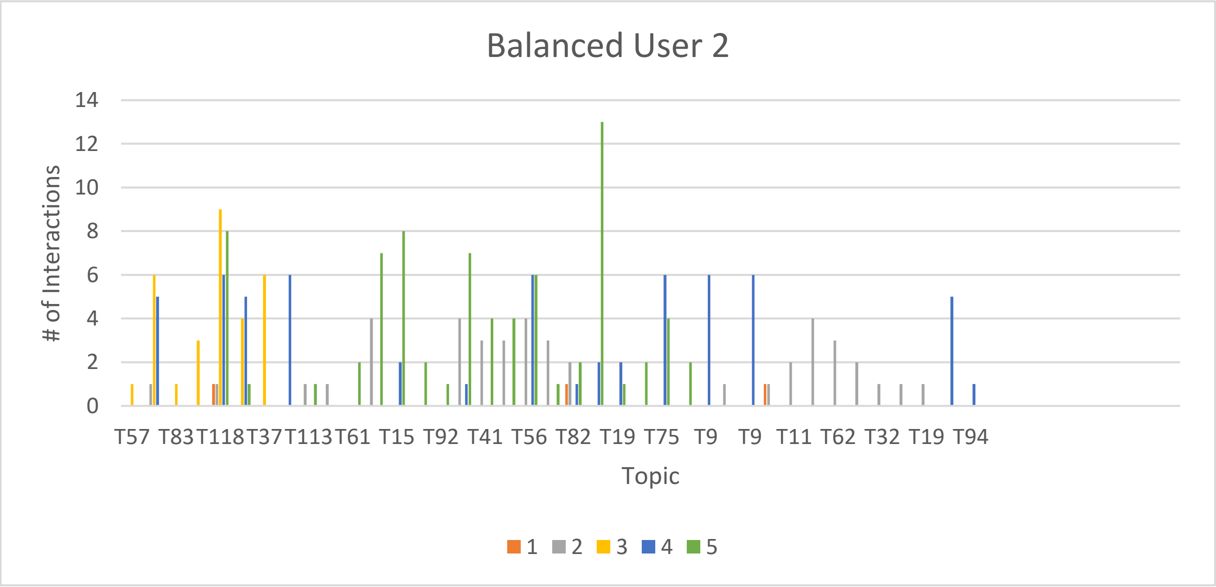
Appendices

Appendix A: User Topic Interactions Sample

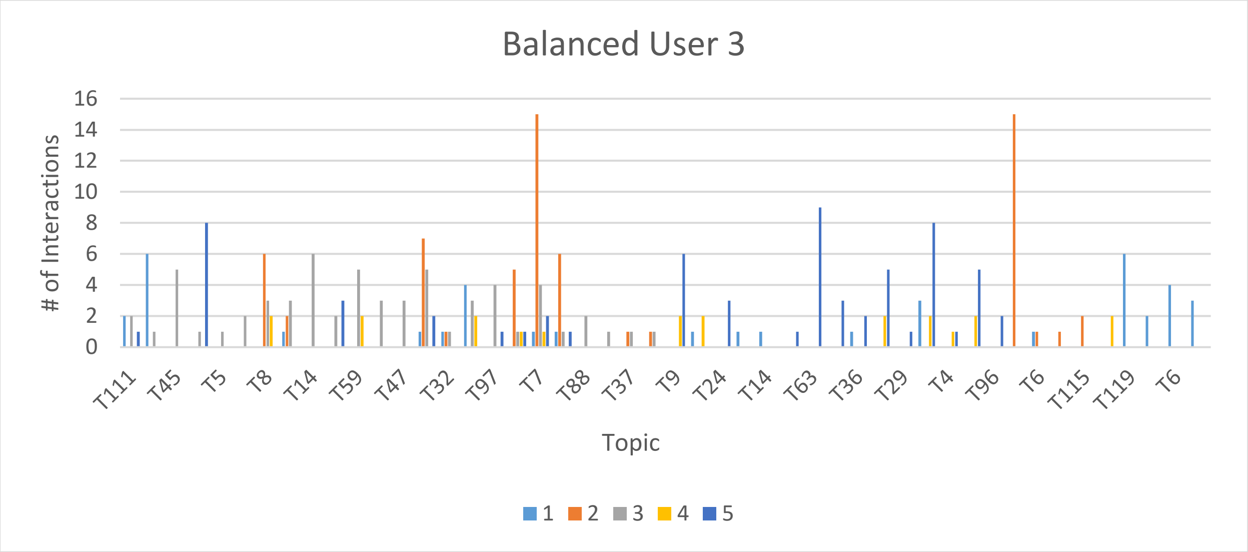
A. 1: Balanced User 1



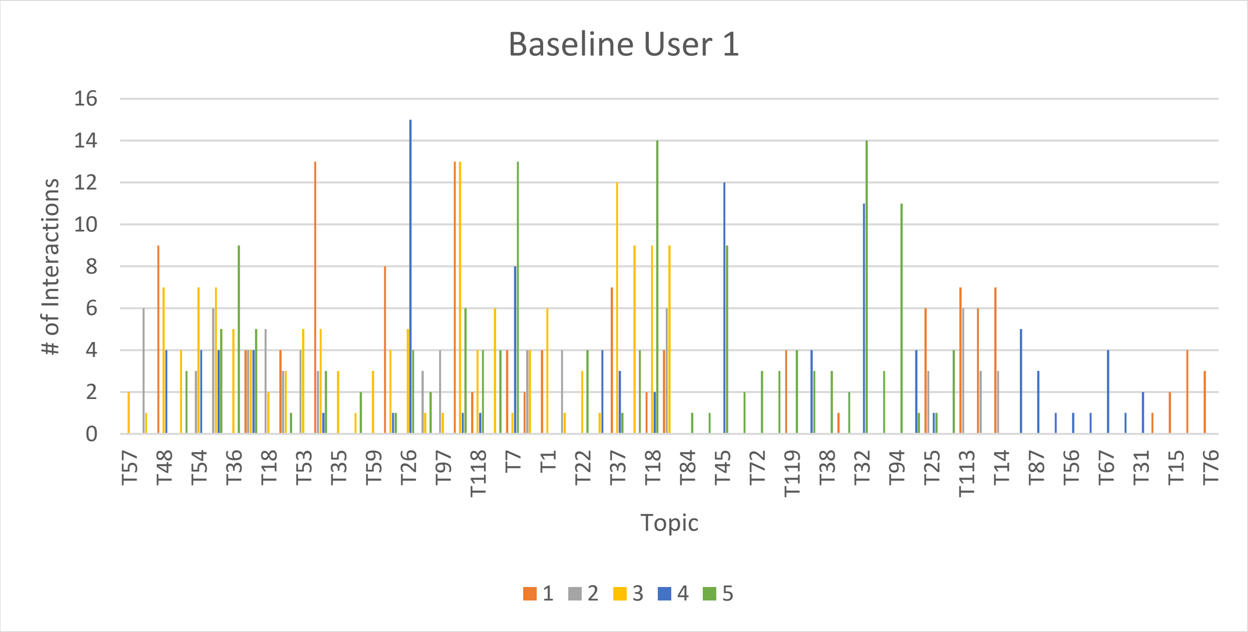
A. 2: Balanced User 2



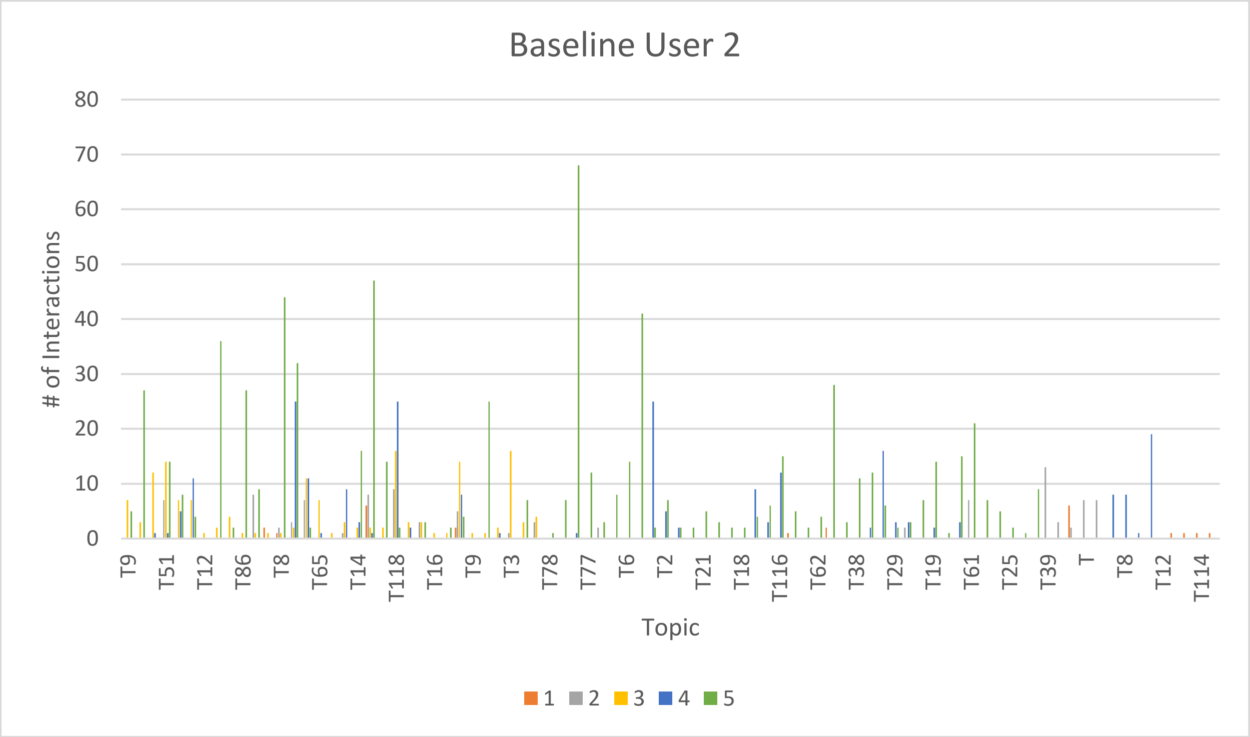
A. 3: Balanced User 3



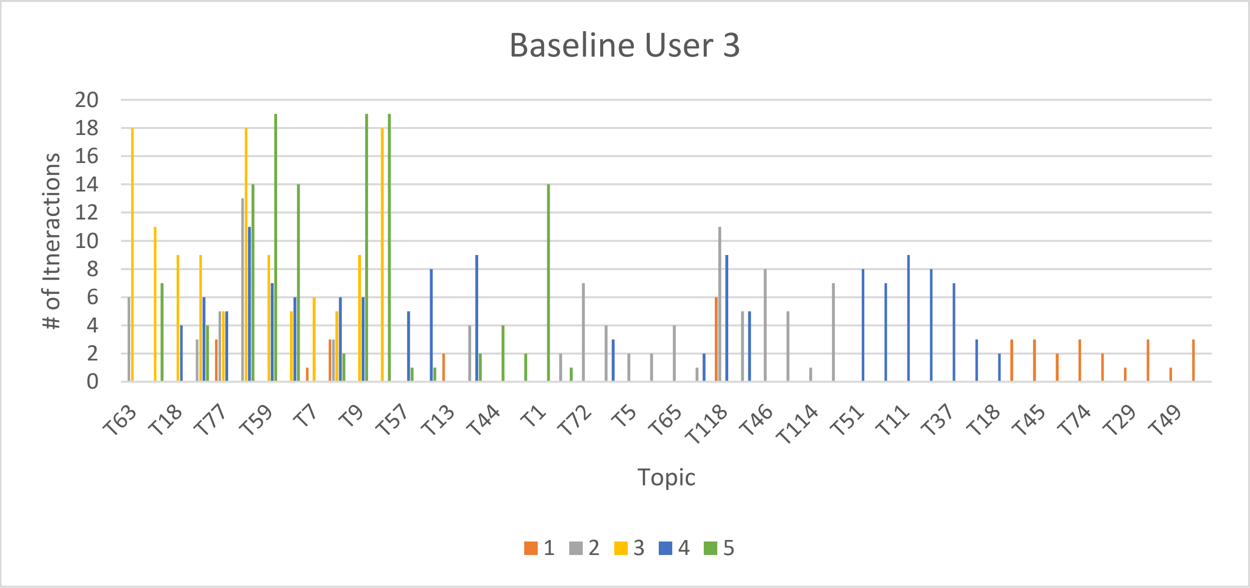
A. 4: Baseline User 1



A. 5: Baseline User 2



A. 6: Baseline User 3



Appendix B: User Survey

Application Results Survey (Highlight/bold your answer or just remove the ones that aren't relevant)

1. Did you feel that results more relevant to your settings day-to-day, as you interacted with the application? e.g. if user profile was noticeably becoming relevant or if you feel setting trending as a high factor was noticeable in the first few topics of your session.

1. Not at all **2.** Infrequently Noticeable **3.** Half and Half **4.** Most of the time

5. Nearly every change was becoming more relevant

*. Include this as well if you feel changes may have been a placebo effect to your settings

2. List order of importance in your settings (Not applicable to designated baseline users). 1 being most important, 5 being least. Put the same number if factors are tied and no number if you generally avoided giving it a value.

_ read history _Article Similarity _Twitter Trends _Database Trends

_Other Users

4. Briefly explain how you generally aimed to set your preferences during the experiment (3 sentences at most). Leave blank if you kept it at the default or were a designated baseline user.

3. What kind of change in topic did you witness as you clicked change typically.

1. No change **2**. Minimal change in subject matter **3**. Steady Change **4.** Mostly different

5. Consistently Different

5. How many of the first 5 news articles would you say were understandable why they were put under the same topic set when clicking the NEXT button typically (e.g. same subject/ topic/ locality/ etc.)? Feel free to put decimals if you feel it's a better representation as well as any comments you may have regarding this.

6. If you felt the change in topic was noticeable, did you value being given articles in new or novel topics to read as you went on? Why if any reason? (2 sentences at most)

7. Which factor(s) do you feel the application succeeded at bringing.

1. Articles Relevant to me **2.** Popular topics in the news

3. Novel news I wouldn’t have looked for

8. Regarding #7 Anything it did particularly better/worse?

9. Thank you for taking part in the experiment and this survey. Feel free to either write here or email the researcher (dejan.spanovic@ryerson.ca) if you have any further questions, concerns, or suggestions.

Survey (Highlight/bold your answer or just remove the ones that aren’t relevant)

System Usability Scale Survey

1= Strongly disagree; 2 = disagree; 3 = neither agree nor disagree; 4=Agree; 5= Strongly Agree

1. I think that I would like to use this system frequently

1 2 3 4 5

2. I found the system unnecessarily complex

1 2 3 4 5

3. I thought the system was easy to use

1 2 3 4 5

4. I think that I would need the support of a technical person to be able to use this system

1 2 3 4 5

5. I found the various functions in this system were well integrated

1 2 3 4 5

6. I thought there was too much inconsistency in this system

1 2 3 4 5

7. I would imagine that most people would learn to use this system very quickly

1 2 3 4 5

8. I found the system very cumbersome to use

1 2 3 4 5

9. I felt very confident using the system

1 2 3 4 5

10. I needed to learn a lot of things before I could get going with this system

1 2 3 4 5

References

[1] H. Wang, F. Zhang, X. X and M. Guo, "DKN: Deep Knowledge-Aware Network for News Recommendation," in *Proceedings of the 2018 World Wide Web Conference*, Lyon, France, pp.1835-1844, 2018.

[2] M. McCrue and E. Doll, Flipboard, Palo Alto, California, 2010.

[3] H. L. Borges and A. C. Lorena, "A Survey of Recommender Systems for News Data," *Smart Information and Knowledge Management*, no. 260, pp. 129-151, 2010.

[4] A. G. Adomavicius, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering,* pp. 734-749, 2005.

[5] E. Rich, "User Modeling Stereotypes," *Cognitive Science,* vol. 3, no. 4, pp. 329-354, 1979.

[6] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction,* pp. 331-367, 2002.

[7] A.D. Fidjestol, X. Su, H.N. Castejon, J.A. Gulla, "Implicit User Profiling in News Recommender Systems," in *WEBIST*, Trondheim, Norway, 2014.

[8] A. Chakraborty, S. Ghosh, N. Ganguly and K. P. Gumma, "Optimizing the Recency-Relevancy Trade-off in Online News Recommendation," in *Proceedings of the 26th International Conference on World Wide Web,* Perth, Australia, pp.837-846, 2017.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[10] C. Desrosiers and G. Karypis, "A Comprehensive Survey of Neighborhood-Based Recommendation Methods," University of Minnesota, Twin Cities, Minnesota, 2011.

[11] S. A. Munson, D. X. Zhou and P. Resnick, "Sidelines: An Algorithm for Increasing Diversity in News and Opinions Aggregators," in *ICWSM*, San Jose, CA, USA, 2009.

[12] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *WWW '01 Proceedings of the 10th international conference on World Wide Web,* Hong Kong, Hong Kong, 2001.

[13] J. S. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, 1998.

[14] C. C. Aggarwal, J. L. Wolf, K.-L. Wu and P. S. Yu, "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering," in *5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, 1999.

[15] A. Tuzulin and G. Adomavicius, "Context-Aware Recommender Systems," University of Minnesota & New Tork University, New York, NY, 2009.

[16] K. Stefanidis, E. Pitoura and P. Vassiliadis, "A Context-Aware Preference Database System," *International Journal of Pervasive Computing and Communications; Bingley,* vol. 3, no. 4, pp. 439-460, 2007.

[17] R. Sankaranarayanan, S. Senm,A. Tuzhilin, G. Adomavicius, "Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach," *ACM Transactions on Information Systems (TOIS),* vol. 23, no. 1, pp. 103-145, 2005.

[18] L. Li, D. Wang, T. Li, D. Knox and B. Padmanabhan, "SCENE: A Scalable Two-Stage Personalized News Recommendation System," in *SIGIR*, Beijing, China, 2011.

[19] K. Krasnashchok and S. Jouili, "Improving Topic Quality by Promoting Named Entities in Topic Modeling," in *ACL 2018*, Melbourne, Australia, 2018.

[20] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," Yahoo Research and AT&T Labs, 2009, pp. 42-49.

[21] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM,* vol. 53, no. 4, pp. 89-97, 2010.

[22] E. Frolov and I. Oseledets, "Tensor Methods and Recommender Systems," Skolkovo Institute of Science and Technology, Moscow, Russia, 2016.

[23] D. Godoy and A. Analia, "Interest Drifts in User Profiling: A Relevance-Based Approach and Analysis of Scenarios," *The Computer Journal,* vol. 52, no. 7, pp. 771-788, 2009.

[24] D. Kosir, I. Kononenko and Z. Bosnic, "Web USer Profiles with Time-Decay and Prototyping," *Applied Intelligence,* vol. 41, no. 4, pp. 1081-1096, 2014.

[25] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu and Z. Chen, "CubeSVD: a novel approach to personalized Web search," in *WWW '05 Proceedings of the 14th international conference on World Wide Web* , Chiba, Japan, 2005.

[26] H. Ge, J. Caverkee and H. Lu, "TAPER: A Contextual Tensor-Based Approach for Personalized expert Recommendation," in *Recsys*, Boston MS, USA, 2016.

[27] B. Chandramouli , J. Levandoski , A. Eldawy and M. F. Mokbel , "StreamRec: A Real-Time Recommender System," in *ACM SIGMOD International Conference on Management of Data (SIGMOD 2011)* , Athens, Greece, 2011.

[28] R. Begleiter, R. El-Yaniv and G. Yona, "On Prediction Using variable Order Markov Models," *Journal of Artificial Intelligence Research,* vol. 22, pp. 385-421, 2004.

[29] G. Zheng, F. Zhang, Z. Z, Y. Xiang, N. J. Yuan, X. Xie and Z. Li, "DRN: A Deep Reinforcement Learning Framework for News Recommendation," in *Proceedings of the 2018 World Wide Web Conference*, Lyon, France, pp. 167-176, 2018.

[30] F. Garcin, C. Dimitrakakis and B. Faltings, "Personalized News Recommendation with Context Trees," in *Recsys*, Hong Kong, China, 2013.

[31] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[32] S. Wang, B. Zou, C. Li, K. Zhao, Q. Liu and H. Chen, "CROWN: A Context-Aware Recommender for Web News," in *ICDE*, Beijing, PR China, 2015.

[33] J. A. Konstan and J. Riedl, "Recommender Systems: From Algorithms to User Experience," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 106-116, 2012.

[34] F. Garcin, F. Galle and B. Faltings, "Focal: A Personalized Mobile News Reader," in *RecSys*, Foster City, Silicon Valley, USA, 2014.

[35] X. Su, O. Ozgobek, J. A. Gulla, J. E. Ingvaldsen and A. D. Fidjestol, "Interactive Mobile News Recommender System: A Preliminary Study of Usability Factors," in *SMAP*, Trondheim, Norway, 2016.

[36] L. Jannachl, I. Kamehkosh and M. Jugovac, "What Recommenders Recommend: An Analysis of Recommendation Biases and Possible Countermeasures," *User Modeling and User-Adapted Interaction*, vol. 25, no. 5, pp. 427-491, 2015.

[37] P. Adamopoulos , "Beyond Rating Prediction Accuracy: On New Perspectives in Recommender Systems," in *RecSys '13 Proceedings of the 7th ACM conference on Recommender systems* , Hong Kong, China, 2013.

[38] P. Castells, S. Vargas and J. Wang, "Novelty and Diversity Metrics for Recommender Systems: Choice, Discover, and Relevance," Universidad Autonoma de Madrid, Madrid, Spain, 2009.

[39] S. Vargas and P. Castells, "Rank and Relvance in Novelty and Diversity Metrics for Recommender Systems," in *Recsys*, Chicago, Illinois, USA, 2011.

[40] Elasticsearch B.V., "Elasticsearch Reference," Elasticsearch B.V., 26 February 2019. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html. [Accessed 26 February 2019].

[41] M. Girolami and A. Kaban, "On an Equivalence between PLSI and LDA," in *SIGIR '03 Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* , Toronto, Canada, 2003.

[42] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research,* vol. 3, pp. 993-1022, 2003.

[43] L. Zhang, F. Li, P. Xia, "Learning similarity with cosine similarity ensemble," *Information Sciences,* vol. 307, pp. 39-52, 2015.

[44] D. Fleder and K. Hosanagar, "Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity,"
Risk Management and Decision Processes Center The Wharton School, University of Pennsylvania, Philadelphia, PA, USA , 2009.

[45] D. Billsus and M. J. Pazzani, "User Modeling for Adaptive News Access," *User Modeling and User Adapted Interaction,* vol. 10, pp. 147-180, 2000.

[46] J. Brooke, "SUS - A quick and dirty usability scale," *Usability evaluation in industry,* vol. 189, no. 194, pp. 4-7, 1996.

[47] J. R. Finkel, T. Grenager and C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," in *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005),* Ann Arbor, Michigan, 2005.

[48] B. Padmanaban and S. Prawesh, "Probabilistic News Recommender Systems with Feedback," in *Proceedings of the sixth ACM conference on Recommender systems*, Dublin, Ireland, 2012.

[49] A. Zimdars, D. M. Chickering and C. Meek, "Using Temporal Data for Making Recommendation," in *UAI'01 Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, Seattle, Washington, USA, 2001.

Glossary

BPR	Bayesian Personalized Ranking
BPTF	Bayesian Probabilistic Tensor Factorization
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
NE	Named Entity
NER	Named Entity Recognizer
PLSi	Probabilistic Latent Semantic Indexing
RSS	Rich Site Summary
SBT	Scala Build Tool
SUS	System Usability Scale
TAPER	Tensor Based Approach for Personalized Expert Recommendation
VMM	Variable Markov Model