Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Christoffer Ventus

A Probabilistic Content-Based News Recommender System

Master's Thesis
Helsinki, May 14, 2015

Supervisor:	Professor Heikki Saikkonen
Advisor:	Antti Rauhala M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Christoffer Ventus		
Title: A Probabilistic Content-Based News Recommender System			
Date:	May 14, 2015	**Pages:**	vi + 62
Major:	Software Technology	**Code:**	T-106
Supervisor:	Professor Heikki Saikkonen		
Advisor:	Antti Rauhala M.Sc. (Tech.)		

Digital content can be created and published in large quantities and at low costs. News publications in particular benefit from being able to reach readers without delay, and because the digital newspaper pages are virtually infinite in size, there is plenty of room for diversity. As the amount of content and variety increase, however, it becomes harder and harder for readers to find relevant news stories. Using knowledge about every news item that has been published and the preferences of readers, news recommender systems provide each user with personalized recommendations. They reduce user effort and make content more accessible and discoverable.

In this thesis the development of a news recommender system is explored with the primary target of finding a simple and extensible solution. Using probability theory as a framework, a model of user behavior is iteratively derived through experimentation and validation. It calculates the probability that an item is selected among a set of candidate items. The model consists of several smaller parts, each of which focus on different aspects of what makes news interesting. Some are personalized and adapt to the user's behavior, while others reflect general usage patterns. There are many challenges involved, and a major one is the cold start problem, which can cause bad results when assumptions are made based on insufficient knowledge. This challenge, and more, define additional requirements on the implementation.

By optimizing for minimal mean cross entropy against the empirical selection distribution of several users, accurate and composable models are found. The time and space complexities of the model are low and there is a straight-forward and mathematically sound way to extend the system further. Evaluation metrics such as item coverage and recommendation diversity provide insight into the large scale behavior of the system and confirm the utility of personalization.

Keywords:	recommender system, recommendation engine, content-based, news, cross entropy, perplexity, diversity, coverage
Language:	English

Aalto-universitetet
Högskolan för
teknikvetenskaper

Aalto-universitetet
Högskolan för teknikvetenskaper
Examensprogram för datateknik

SAMMANDRAG AV DIPLOMARBETET

Utfört av:	Christoffer Ventus		
Arbetets namn: Ett probabilistiskt innehållsbaserat nyhetsrekommenationssystem			
Datum:	Den 14 maj 2015	**Sidantal:**	vi + 62
Huvudämne:	Programteknik	**Kod:**	T-106
Övervakare:	Professor Heikki Saikkonen		
Handledare:	Diplomingenjör Antti Rauhala		

Digitalt innehåll kan skapas och publiceras i stora mängder och till låga kostnader. Nyhetsförmedlingar i synnerhet kan dra nytta av möjligheten att genast nå läsare och eftersom det digitala pappret är så gott som oändligt finns det rum för mångfald. Men allt som storleken på innehållet och variationen ökar blir det svårare och svårare för läsare att finna relevanta nyheter. Genom att använda kunskap om varje nyhet som publicerats och läsarnas intressen kan ett rekommendationssystem tillgodose varje läsare med personliga rekommendationer. De underlättar användandet och gör innehållet lättillgängligt och enkelt att få en översikt av.

I det här diplomarbetet undersöks utvecklandet av ett nyhetsrekommendationssystem vars främsta mål är att finna en enkel lösning som kan utökas. Med sannolikhetslära som grund härleds en modell av användarbeteende genom upprepade experiment och validering. Modellen räknar ut sannolikheten att en nyhet väljs bland en mängd valmöjligheter. Den består av flera mindre delar och var och en av dem fokuserar på olika aspekter av vad som gör nyheter intressanta. En del skräddarsyr resultaten och anpassar sig till användarens beteende och andra speglar allmänna användningsmönster. Det finns många utmaningar varav kallstartsproblemet är viktig, vilken kan leda till odugliga resultat när antaganden görs med otillräcklig information. Denna utmaning bland andra utgör ytterligare krav på implementationen

Genom att optimera för minimal medelkorsentropi gentemot flera användares empiriska valdistributioner skapas modeller som ger goda resultat och vilka kan sammanslås. Modellens tids- och rymdkomplexiteter är låga och det finns enkla och matematiskt korrekta sätt att vidare utöka systemet. Utvärderingsmått som täckning och rekommendationsmångfald beskriver modellens storskaliga beteende och bekräftar nyttan av personliga rekommendationer.

Nyckelord: rekommendationssystem, innehållsbaserad, nyheter, korsentropi, perplexitet, mångfald, täckning

Språk: Engelska

Acknowledgements

I would like to thank my supervisor Heikki Saikkonen for taking on the role without hesitation with short notice, Antti Rauhala for invaluable insight, expertise and proofreading and for your enthusiasm and drive, Paavo Punkari and Mikko Vihonen for making this project a reality and believing in the concept, Mark Voit for helping and staying positive despite all the additional work that piled up because of me, Maj-Len Ventus for proofreading and support, and everyone at Futurice for showing interest in my thesis and being an awesome bunch of people to work with.

Most of all I would like to thank my wife Sofia and my son Charlie for being understanding and supportive throughout this endeavor. Without your encouragement I would not have been able to see this through.

Helsinki, May 14, 2015

Christoffer Ventus

Contents

Chapter 1

Introduction

The Internet and the digitization of intellectual property has changed the way content is created and distributed. Music, videos, books, and more, which were once limited to the physical space, can now be distributed and sold online at marginal costs. Online distribution has the potential to target a global audience and can reach consumers in a small amount of time. Retailers can put their products on a virtually infinite display in the web browsers of its customers. The wide spectrum of tastes, colloquially called the long tail, can more easily be served when the threshold to create content is low. There is more diversity and greater potential to target niche interests.

But as the quantity and diversity increase, new problems appear. If the number of options is so great that not every one of them can be evaluated on their own, then it is difficult to get an overview or understanding of what is available. At a certain point it becomes unrewarding and even impossible to browse through everything and the value of diversity starts to decrease. The development of accurate search engines was a breakthrough for browsing the web, because suddenly it was possible to quickly find what you were looking for. But you do not search for what you are unaware of, and not every worthwhile thing can be promoted equally. In this case it does not matter that something is technically accessible.

Information overload is a problem of usability which makes it difficult to retrieve information from a system. Recommender systems are alternatives and complements to search engines for making large and unknown collections of items discoverable and manageable. With knowledge of what is in store, a recommender system picks out a selection of items that a particular user might like, and makes frequently accessed items easily available.

News is by its nature surprising, and unanticipated events cannot be searched for. Readers are unable stay informed if news that affect them reaches their attention too late. News media are in the business of spreading ideas and information and have the same opportunities to diversify their content as other content providers. They also face the same problem of information overload, and targeting news, which has traditionally been achieved through separating publications, can be done on a per-user basis in the digital age. With a recommender system every reader could have a tailor made news source which contained only interesting content from a wide variety of topics.

1.1 Problem Statement

Digital news publication allows a wide range of topics and special interests to be covered. Certain stories are only interesting to some readers and add no value to others. The intellectual capacity is limited and what readers are exposed to should primarily consist of interesting and relevant news and this can vary from person to person. In this thesis the development and implementation of a recommender system will be explored and applied in the context of news personalization. The primary question is how to attain a simple and scalable solution that provides recommendations of adequate quality and a means to extend it incrementally. The secondary goal of the thesis is to uncover what properties of news items make them interesting to some readers and not to others, and whether these properties can be measured and reasoned about in isolation.

1.2 Thesis Structure

The following two chapters set the stage for the rest of the thesis. In chapter 2 the field of recommender systems is introduced. Basic concepts and algorithms are covered with a focus on probabilistic methods. There are certain challenges that are inherent to any recommender system, and these are discussed at the end. Following the theory, chapter 3 describes a news aggregator service which provided the context for the thesis. In order to find an answer the problem statement, a recommender system was developed for a news application.

The field of information retrieval is rich, and many algorithms and methods are described in the literature. There are many ways to choose an implementation and in chapter 4 the principles by which the recommendation algorithm was developed are outlined. The implementation consists of several components, and in chapter 5 they are described in terms of how they were derived, how they work, and how they fit together to form a whole. The accuracy and quality of the implementation was continuously measured and the results of individual models and combinations of models are presented in chapter 6.

The thesis is wrapped up in the two final chapters. In chapter 7 the results are interpreted, the validity of the implementation is assessed, and future research is proposed. The overall work is also evaluated in light of the problem statement. The thesis is concluded with chapter 8 where the most important topics and results are reviewed.

Chapter 2

Recommender Systems

A recommender system is a tool that helps users find relevant and interesting items in a large body of content. Items are pieces of the content a service provides or distributes, for instance movies that can be streamed or products sold by an online retailer. Users consume items, and it is possible that they are unaware of a range of items that would suit them. A recommender system makes the content discoverable by putting a magnifying glass on the items the user is most likely to enjoy.

The study of recommender systems is a branch of *information retrieval* with the characteristic that users do not need to provide a query that states what they are looking for. Instead, the system predicts what a user might want based on context and knowledge about users and items. A high level description of a recommender system is that it is a piece software that selects a subset of the content that should be visible in the user interface. One common requirement is that recommendations are *personalized*, i.e. tailored to the needs and interests of each user individually, but an exact definition depends on the application and can vary. It is sometimes thought of as a filter that removes noise from a stream of content, and in other cases it is meant to help users compare similar items so that an informed decision can be made. [23]

2.1 Ratings

A rating is an expression of the quality or desirability of an item. At the same time a rating is a reflection of the interests of a user, and the items a user chooses not to use or rate is similarly telling. A rating is a form of *relevance feedback* which are used to measure the accuracy of an information retrieval system [24]. In a recommender system they are also used to learn about a user's preferences and how items relate to one another based on who gives them ratings.

An application can be designed to allow users to *explicitly* rate the content they use and consume. For instance, users can give star ratings or a thumbs up, or they might be able to write a review about a product. An explicit rating is a reliable indicator of what a user thinks about an item but explicit ratings put a burden on users because they are expected to do a small amount of work for an intangible reward.

Users also express their interests *implicitly* through their actions. When a user opens a web page and spends a long time reading it, the behavior can intuitively be interpreted as being caused by the user's interests in the contents of the page. Other implicit ratings include the act of explicitly rating an item (or choosing not to rate an item that was used), searching for something using keywords, bookmarking something for later consumption, and buying an item from an online store. People make mistakes, however, and sometimes an action doesn't reflect a user's interests. For instance, a link can be clicked by mistake, and it is not uncommon to buy gifts to give to friends. On the upside, implicit ratings are abundant and easy to collect. With enough implicit ratings the number of misinterpretations can be negligible. [23, 24]

The way ratings are collected restrict the values they can take. Explicit ratings can have an arbitrary scale, but implicit ratings can be hard to grade. For instance, when a user selects an item from a list for closer inspection the selection is an implicit rating. Whether or not the item turned out to be relevant to the user cannot be deduced from the selection alone; it is merely a sample of an item that the user selected. Items the user did not select are not necessarily uninteresting – there are many reasons the user might not closely inspect everything they see. On the other hand, an action with inherently negative connotations in terms of relevance, such as "remove item from view" or "open next page of results", could be used to indicate a poor rating.

2.2 Recommendation Algorithms

At the heart of a recommender system is an algorithm that generates personalized recommendations for each of its users. Its primary input consists of the ratings that users have made in the past. If a user prefers item a over item b then it should follow that the user's rating of item a is higher than that of item b. Based on this correlation the ratings that have been made in the past are used to predict how items will be rated in the future. The user's interest in a new item c can be guessed based on the user's ratings of items a and b.

Conceptually, there is a matrix $\boldsymbol{R}$, called the *utility matrix*, that has a row for every user and a column for every item. If $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ is the set of all users and $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ is the set of all items then the element r_{ij} in the matrix is the rating of item x_j by user u_i. In most cases a user has rated only a small fraction of the whole content which makes the rating matrix very sparse with many missing elements. In addition to ratings, specific knowledge about items or users might be available, such as the genre of a movie or the age of a user which can be used as additional input to make recommendations more accurate. [32]

Under the simplistic assumption that ratings are isomorphic to preferences, the task of a recommender system is to estimate the rating $\hat{r}_{ij}$ of each item x_j in a set of candidate items $\boldsymbol{\mathcal{X}}_c$ and list them ordered by their estimated rating. The set of candidate items can be the set of all items, or it can contain items that have been filtered for the user in advance. For instance, the language a book is written in should be one that the user understands and a user might not be very interested in a book that they have already read [15].

Many recommendation algorithms can be categorized as being based on either *collaborative filtering* or *content-based filtering*. They not mutually exclusive and some implementations are influenced by both approaches. Content-based algorithms find similarities among the items that a user has rated in the past. For instance, a user that frequently reads about sports is likely to do so in the near future as well. Items need to have distinguishable *features* that make them similar or different, and in the example there were a set of items that had been identified to share the "sports" feature. This can be represented as a content matrix $\boldsymbol{C}$ where each row is an item and each column is a feature. The matrix can be heterogeneous with both continuous valued and discrete valued columns. Documents, for instance, can be told apart by the words that they contain and different kinds of food have characteristic nutritional values. Difficulties arise when attempting to compare very different items, as one might in a web store. Purely judged by content, an apple and a book are very different, but given that a user has put a cook book in the shopping basket, it is possible that ingredients for an apple pie are relevant to the user. A recommender system that finds connections like these reduces effort and increases sales. [8, 23, 31]

Algorithms that use ratings made by many users to predict the rating of a target user falls under the collaborative filtering category. Groups are formed that consist of users that tend to give similar ratings to items. Items that some members rate highly likely to be interesting to the rest of the group, and vice versa. Items are distinguished and deemed similar based on how they are rated, and ratings are all that collaborative filtering algorithms need. There is no need for a content matrix, and this makes them applicable in many different domains. [11, 15, 19, 27].

Many algorithms have been invented for both collaborative filtering and content-based filtering, and some are applicable to both. Sometimes a distinction is made between *memory-based* and *model-based* algorithms [11, 28]. A memory-based algorithm stores all the data it needs, for instance the utility matrix, in memory and makes most calculation on the fly. For every calculation the latest state of the system is used which enables the results to be up to date. An alternative is to approximate the procedure through which ratings are made with a model. A model consist of an algorithm and model parameters and the same algorithm can be used to predict ratings in different cases, for instance, for users by specifying different parameter values for each of them. The utility matrix $\boldsymbol{R}$ and the content matrix $\boldsymbol{C}$ can be used to estimate model parameters which take up much less space. When the data changes the model parameters need to be adjusted or recomputed, which can be a computationally intensive process that takes a long time to complete.

2.2.1 Similarity

The k-nearest neighbors algorithm (k-NN) is a basic method for finding similar users or items. It is commonly used for search tasks and other kinds of information retrieval problems, such as recommender systems. The similarity, or distance, between users or items is defined by a metric, for instance, the Pearson correlation coefficient or cosine similarity, and by calculating the metric for every pair, the k most similar ones can be found. A naïve implementation of the k-nearest neighbors algorithm does not scale well as the numbers of users, items or features increase, but there are ways to reduce the amount of pairs that have to be compared [23, 32].

When the rating r_{ij} is estimated for user u_i in a collaborative filtering system, the neighbors are other users who have rated x_j and who have similar tastes. Users who have given items similar ratings show similar tastes and the distance between them is small. The estimate $\hat{r}_{ij}$ can be given by a linear combination of the nearest neighbors' ratings of the item weighted by their distance to the user. [11, 15]

Content-based similarity algorithms look for neighbors of item x_j among the items user u_i has rated in the past. The distance is measured in terms of how similar their features are, for instance, to what degree the vocabulary of two documents match. Then the rating r_{ij} can be estimated based on how user u_i rated the item's nearest neighbors [8, 23]. Collaborative filtering can also be applied in a similar manner by transposing the roles of items and users. The neighbors of item x_j are in this case other items which have received similar ratings from some users [27]. In one sense, the ratings that items receive become features and the utility matrix $\boldsymbol{R}$ plays the part of the content matrix $\boldsymbol{C}$. One difference is that the more users rate the more becomes known about items.

2.2.2 Manifest and Latent Variables

The words in a document and the ratings made by a user are observable quantities and are called manifest variables. Comparing the words in two documents or the items rated by two users can lead to a very shallow comparison. When estimating r_{ij}, it is possible that some users have rated many items that are similar to x_j except the target item itself. These users cannot be used as neighbors in a k-NN algorithm for user u_i even though they might otherwise have very similar tastes. In the same vein, two documents that cover the same subject do not necessarily have to use the same vocabulary and when comparing their use of words they appear different.

The semantic meaning of documents or the interests shared by users can be seen as abstract concepts that make two superficially different things similar. Only the words in a document are observed but its tone and writing style and the subjects it covers determines the words that are used. User interests have not been expressed explicitly but they can explain why ratings were made. These abstract concepts can be modeled as latent variables. Matrix factorization algorithms that are based on linear algebra, such as Singular Value Decomposition [25] and Principal Component Analysis [20], as well as topic models, such as probabilistic Latent Semantic Indexing (PLSI) [15, 21] and Latent Dirichlet Allocation (LDA) [10], have been used to find high level patterns among items and users in recommender systems. Some latent models output the estimate $\hat{r}_{ij}$ directly and others can be used for dimensionality reduction to find the nearest neighbors in the much smaller latent variable space.

2.2.3 Probabilistic Algorithms

Probability theory is a well studied field of mathematics and provides tools for dealing with uncertainty and modeling and predicting the behavior of complex systems. In a recommender system the rating r_{ij} can be modeled as a random variable R and a probabilistic model can be used to estimate its value. It might be assumed that the rating depends on the user and item in question and an estimate is then given by the expected value

$$\hat{r}_{ij} = \mathrm{E}_{ij}[R] = \sum_{r \in \mathcal{R}} r \, \mathrm{p}(R = r \mid X = x_j, U = u_i, \Theta = \theta) \tag{2.1}$$

where $\mathcal{R}$ is the range of values a rating can take, and θ represent model parameters and contextual information, such as the current time and date. In this case the event space is user ratings, and whenever a rating is made, values for R, X, U and Θ can be observed. When all random variables are discrete $\mathrm{p}(R \mid x, u, \theta)$[1] represents a probability mass function and the way it is defined determines the behavior of the recommender system.

Even if r_{ij} is estimated very accurately it is not necessarily true that a user is going to be equally interested in an item at every point in time. Probability theory provides a way to reason about recommender systems that are not based on ratings alone. This is not as straight forward to achieve when an algorithm based on, for instance, linear algebra or the average rating of the nearest neighbors. The problem of recommending items can be defined as the probability $\mathrm{p}(x \mid u, \theta)$ which is the probability that an item is *selected* rather than given a particular rating. When the event space consists of selections, they can easily be used as implicit ratings.

A probability distribution can be implemented by any non-negative function as long as its total sum or the integral over its domain is one. There are many traditional probability distributions, such as the Bernoulli and Binomial distributions, which have been examined in detail and are used to model various discrete phenomena. Functions that do not sum to one can be turned into a probability mass function by restricting its domain and normalizing it by a constant factor. If $f(z)$ is a non-negative function for arguments in set $\mathcal{Z}$ and it is greater than zero for at least one of them, then a probability mass function $g(z)$ can be defined as

$$g(z) = \begin{cases} f(z)/Z & \text{if } z \in \mathcal{Z}, \text{ and} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where $Z = \sum_{x \in \mathcal{Z}} f(x)$. [12]

Because only one article is selected whenever a selection occurs, the probability $\mathrm{p}(X = x)$ that article x is selected is easy to derive. It is related to an item's overall popularity and is proportional to the number of times it has been selected [13]. Recommending items based on their popularity does not

[1] As is customary, random variables are written in upper case and specific values they can take are in lower case. The notation $\mathrm{p}(A)$ represents the probability distribution of A and when the meaning is clear from context $\mathrm{p}(a)$ is used as a shorthand for the probability $\mathrm{p}(A = a)$ that A has the value a. User and item subscripts are omitted from here on.

provide personalization on its own because the overall popularity of an item is the same for every user, but due to its simplicity and ease of implementation, it can be useful nonetheless. In a news recommendation competition Lommatzsch [29] found that the most popular news item was often selected more frequently than the what more complex algorithms predicted. Item popularity is often used as a baseline to which other models are compared [15, 20] and, for instance, Breese et al. [11] calls it a zero-order collaborative filtering model because every user in the system is used as neighbors. The age of the selections affect the result of a popularity model, and it is often desirable that old selections play a smaller role than new ones, which should reflect the current popularity.

When there are more than one random variable, there are many possible combination of values and it can be hard to find the definition of the joint probability function. The joint probability can be reformulated, however, through Bayes' theorem, the chain rule, and discovery of (conditional) variable independence. A Bayesian network is one way to define a large joint probability function. It is a directed acyclic graph where each node is a random variable and dependencies among them are represented by edges. Each random variable is conditionally independent of every other variable except of its parents and conditional probabilities can be efficiently calculated [3, 9]. In a recommender system this can be put into use by creating a random variable for the rating of each item, and based on the ratings a user has given to some items, the expected rating of the other ones can be estimated. Breese et al. [11] created a collaborative filtering algorithm that used a Bayesian network to recommend, among other things, support pages from a knowledge base. They used an algorithm to find edges between nodes from a data set of user selections. They also developed a model using Bayesian clustering for comparison. Clusters represented user tastes and preferences and conditioned on the cluster variable C the item ratings $X_1, X_2, \ldots, X_N$ were independent. The number of clusters was chosen based on maximum likelihood and the cluster prior probabilities and the conditional rating probabilities were latent variables that were inferred using the Expectation Maximization (EM) algorithm [17]. The independence assumption that was used results in a joint probability that has the form

$$\mathrm{P}(C, X_1, X_2, \ldots, X_N) = \mathrm{P}(C) \prod_{i=1}^{N} \mathrm{P}(X_i \mid C) \tag{2.3}$$

which can be regarded as an extreme variant of a Bayesian network. In classification tasks where C is an observed class label it is often called the naïve Bayes due to its strong independence assumption. Billsus and Pazzani

[8] used a naïve Bayes classifier for modeling users' long term interests in a content-based news recommendation software agent. The presence or absence of important and distinguishing words in an article were used as binary features for classifying an article as interesting or uninteresting for the user.

PLSI is a latent variable model in which the probability of an observation $\langle u, x \rangle$ that user u selects item x is modeled as being caused by a latent class variable z, which makes u and x conditionally independent. For instance, the observation is caused by the user's interest in a topic, which z represents, and the fact that the item is strongly related to that topic. A user makes a selection with probability $\mathrm{p}(u)$ and shows interest in topic z with probability $\mathrm{p}(z \mid u)$. An item might cover more than one topic and the probability that item x is selected when a user is interested in z is $\mathrm{p}(x \mid z)$. By marginalizing out the latent class variable z the probability of the selection $\langle u, x \rangle$ is

$$\mathrm{p}(x, u) = \mathrm{p}(u) \sum_{z \in \mathcal{Z}} \mathrm{p}(x \mid z)\mathrm{p}(z \mid u). \tag{2.4}$$

The number of latent variables $|\mathcal{Z}|$ is a model parameter and since z is never directly observed both $\mathrm{p}(x \mid z)$ and $\mathrm{p}(z \mid u)$ should be chosen such that they minimize misprediction. A recommender system is concerned with the probability that a specific user selects an item which is easily derived from equation 2.4 with the definition of conditional probability and is

$$\mathrm{p}(x \mid u) = \frac{\mathrm{p}(x, u)}{\mathrm{p}(u)} = \sum_{z \in \mathcal{Z}} \mathrm{p}(x \mid z)\mathrm{p}(z \mid u). \tag{2.5}$$

If there are n users and m items there are a total of $|\mathcal{Z}|(n + m)$ parameters that have to be estimated and Hofmann [21, 22] proposes a method to apply the EM algorithm to find parameter values that maximize the log-likelihood function $\log \mathrm{p}(x \mid u)$ of every observed $\langle u, x \rangle$ pair.

In combination with two other models, Das et al. [15] used PLSI for collaborative filtering in Google News and showed how it can be implemented in terms of Map Reduce [16] to make it scale to millions of users and selections per day. Compared to the recent popularity $\mathrm{p}(x)$ of an item x they found that their combined model gained 38% more user selections on average. The PLSI model parameters were periodically calculated offline in a large computational cluster but some parameter values could be adjusted slightly after every selection.

Liu et al. [28] later extended Google News with a simple content-based model. Articles were classified based on their content into one of several categories. The system kept track of how often an article of each category was selected by users during a period of time T. Selection frequencies were used to estimate the probabilities $\mathrm{p}(c_k \mid u, t)$ and $\mathrm{p}(c_k \mid t)$ that a specific user or any user selected an item of category c_k at time t. The probability that a user is going to select an item of a certain category in the near future was derived using Bayes' theorem and has the form

$$\mathrm{p}(c_k \mid u, T = \text{now}) = \frac{\mathrm{p}(c_k \mid T = \text{now}) \sum_t \left(n_t \frac{\mathrm{p}(c_k \mid u,t)}{\mathrm{p}(c_k \mid t)} + K \right)}{\sum_t (n_t + K)} \tag{2.6}$$

where $\mathrm{p}(c_k \mid T = \text{now})$ was estimated from selections made the last hour, n_t was the number of selections the user made at time t, and K is a smoothing factor in case a category was never selected by the user. The fraction $\mathrm{p}(c_k \mid u, t)/\mathrm{p}(c_k \mid t)$ can be interpreted as how much a user's interests differ from that of the general public. The model parameters are very simple to estimate and update, and for each user only $\mathrm{p}(c_k \mid t)$ needs to be recorded which require storage that grows linear to the number of time periods that are kept for each user.

2.3 Challenges

Making useful and accurate recommendations is a difficult problem, but no matter the domain or implementation, there are other common challenges that any recommender has to be able to handle. Any system that is made to build user profiles is a cause for privacy concerns, and in the real world there can be malicious users that try to game the system in ways that can affect others [23]. These topics are very important to consider and it can be hard to find definite answers. There are also some purely technical challenges that are more straight forward to identify and address.

2.3.1 Cold Start

Recommendations are based on knowledge, but when a piece knowledge is missing a recommender system still needs to provide results. A collaborative-filtering algorithm bases its recommendations on the ratings made by other users. When a new item is introduced to the system nobody has had the chance to give it a rating. To rate an item a user first needs to find it, but if the algorithm is built with the assumption of complete knowledge it has no basis to recommend a new item to anyone. This is an instance of the *cold start* problem. If users have other ways than their recommendations to browse content the item might eventually be recommended to other users, but there can be a significant amount of delay between the time the item was added and when it has gained enough ratings to be widely recommended. [23, 28]

Another aspect of the cold start problem shared by both collaborative filtering and content-based filtering algorithms is the lack of knowledge about users that recently joined the system. When there is no knowledge about a user you can at best make a guess based on prior belief of what users like in general. The rate at which recommendations transition from being general to being personalized is crucial and is related to the confidence in the knowledge a system has about a user's true preferences. By personalizing too rapidly there is a risk that the ratings made in the past are not representative of the user's much wider preferences, and recommendations become too focused. This makes a large portion of interesting content less accessible. If the rate is too slow, however, the recommender system adds very little value until a user has invested enough time into it.

2.3.2 Scale and Latency

A key usability metric is responsiveness. If the loading of a page takes a noticeable amount of time to complete the user experience suffers and users start looking for alternative services. Delays can result in user experience issues that not even perfect recommendations could alleviate. Every request for recommendations needs to generate a useful personalized result for every user, and the result needs to be made visible within a small amount of time. One strategy to simplify this problem is to perform as much computation as possible offline so that requests can be responded to in linear or constant time [15, 20]. If the algorithm cannot be computed every time for every request then there is a trade-off between responsiveness and adapting to changes in users and content.

2.3.3 Concept Drift

Global and local trends affect how items relate to one other and the overall desirability of an item. For instance, two pieces of apparel that are worn in combination might not be a fashionable thing to do in the future, and in an news preference experiment Liu et al. [28] showed that the interests of users change over time. If a system recommends products that used to be but are no longer bought in combination, or if it assumes that a user is always going to be interested in something they liked a long time ago, then the recommendations will be poor and might even mislead users. When the statistical properties of a system change over time it is non-stationary, and this is often called concept drift. It causes a model to be less accurate over time and it implies that item correlations, rating and other model properties need to be interpreted in their temporal context.

Another example of dependencies among ratings is the anchoring effect. The anchoring effect is the human tendency to place too much weight on a first experience which is then used as a reference point by which future experiences are judged. For instance, a user is likely to give lower ratings to an average movie if the previous movie watched (i.e. the anchor) was rated highly than if the previous movie was disliked [18]. This makes ratings depend on the order in which they were made, and for instance, whether they were made during the same session or not.

2.3.4 Granularity

Recommendations cannot be made more specific than to what degree items can be told apart. Imagine there is a user who really enjoys *apples* and buys one every day. At the same time the user really dislikes the taste of *orange*. A recommender system might come to the conclusion that the user likes *fruit* because apples are categorized as such. However, if oranges also have that label then the recommender system will not be able to deduce why fruit is sometimes well received and other times disliked. The coarse grained labels hide the finer details.

On the other hand, if there are many features, and each of them are very specific, then there will likely be dependencies among them. For instance, a book about *art of the renaissance* have something in common with books about the *renaissance* which in turn share some concepts with books about the *age of enlightenment*. A rating given to the one says something about the interests in the others, but discovering to which degree individual features or combinations of features depend to one another can be very hard. Fine grained features might make it hard to see the forest for the trees. [23]

Chapter 3

News Aggregation

A news aggregator is a system that collects news from several sources and presents it to users in a uniform and consistent manner. User effort is reduced because users are exposed to the latest content from each source as soon as it is available, and they do not have to check up on every source individually. When a story is covered by several sources it is also easier to compare what different journalists and publications have to say about it. An aggregator can be selective in the sources it uses which can enable a user to find related news sources of high quality.

News is history unfolding itself and is an ever expanding body of information. The editors of a publication select which are the most important stories and which ones should be published each day. Articles in a newspaper are organized into sections and the number of articles is limited to make the scope manageable. If there are many sources and if the number of published news items is high then a news aggregator that simply acts as a funnel can easily overwhelm the user and the signal gets lost to the noise caused by the large volume of uninteresting articles. For this reason it is common for news aggregators to filter out less relevant content so that users are able to browse through the most interesting content.

Google News[1] is an example of a news aggregator service which links to news stories from thousands of publications world-wide [7]. It uses a recommender system to provide users with personalized news which is combines both collaborative and content-based filtering [15, 28]. Other services, such as Digg[2] and reddit[3], are based on crowd-sourcing for content and filtering. Users submit links to news stories which are collectively voted on. The highest rated recent items appear most prominently on each users' news feeds.

[1]`https://news.google.com/`
[2]`http://digg.com/`
[3]`http://www.reddit.com/`

3.1 System Integration

In order to answer the research question of this this master's thesis, a recommender system was developed and integrated into an online news aggregator service. When the thesis began the service was in active development and a beta version was available to a small number of users. Content was licensed from many different news publications, most of which published both online and in print. There were monthly periodical magazines and daily newspapers among the sources and some were somewhere in-between. News could be browsed based on categories, but to make the content more discoverable and to improve the user experience, there was also going to be a personalized feed which selected news for each user and adapted itself to their reading habits. The personalized feed was realized with the recommender system.

The service consisted of a server and various clients. The server exposed a REST API through which content and news feeds could be retrieved. A web front end rendered HTML for browsers, and native mobile applications used the REST API to create views for users of mobile devices. The server monitored the sources for new content and served the most up to date news whenever a client made a request for it. In addition to requesting news, the clients informed the server of user interaction events, such as when the user selects a news story to read.

A news feed is a list of news items, and typically a client would display it to users in the order that the server provided. Items in the list could be selected and would take a user to an article view where the text can be read. In the personalized news feed highly recommended items appeared in the beginning of the list and items the user was unlikely to select appeared farther down in the list. The better the recommendations the less the user had to browse.

3.2 News Recommendation Challenges

Recommending news is in some ways different from other domains, such as retail. The shelf life of news is often measured in days or hours, and even small delays between publication and delivery has detrimental effects. The cold start problem can be one source of delays and is a primary concern when recommending news. News is also made at a rapid pace and covers diverse range of topics. A story that is breaking might be carefully followed for a a few days after which it is no longer of interest. Other phenomena are periodic, such as the Olympic Games, and their relevance is likewise periodic and temporary. [15, 28]

Chapter 4

Method

In the literature there are numerous methods and algorithms to choose from and many have been applied in recommender systems with varying degree of success. What counts as the best one depends on domain, context, and what qualities are sought-after. Therefore it is important that the algorithm that is implemented is proven to work and that its results are interpretable and understandable. In this chapter a method to develop a news recommender system is described.

4.1 Model Validation

A model is used to emulate a process where the inner workings are unknown and only the effects can be observed. From the point of view of a news recommender system there is something that causes individual users to select some news items and ignore others. If there was a model that described this system, then it would be possible to predict what users are going to select ahead of time.

There are often many models that can describe an unknown process in their own ways, and in order to know whether a model is good fit or not it has to be tested. The effects of the system can be observed and the accuracy of a model is related how well it can predict the effects in advance. A common way to evaluate a model is to record the inputs and outputs of a system and tune a model to produce the same or similar output for an input. A recommender system that models user behavior takes at least a user and a point in time as input and predicts the item the user selects at that time as output. A probabilistic recommender system allows for some uncertainty and assigns to each item the probability that it will be selected when the choice is one of the items. When there is a data set of selections made by users, it is first split into two disjoint sets called the *train set* and the *test set*. The train set is used to tune model parameters and the model is tested by comparing the model's predictions with the selections a user actually made, which are the selections in the test set. [1]

Because selections are made sequentially over time it is possible to use online algorithms, in which the model parameters are adjusted after a selection has been observed. With enough data most models make good predictions [2], and the more is known about a user the better the recommendations can be. But in light of how the cold start problem can affect the user experience, as described in section 2.3.1, it is important that a recommender system also behaves well for users that have made very few selections. With an online learning algorithm the error of a model can be checked after each selection from the training set. If the error increases or has a high variance at small sample sizes, then a user might lose faith in the recommender system and stop using the service altogether.

4.2 Evaluation Metrics

The quality of recommendations need to be measured in order to know whether a change improves the results or makes them worse. A common accuracy metric used for various information retrieval tasks is *precision* and *recall* [30]. These are metrics of the fraction of items that a user found relevant among a set of item that was retrieved. They are generally applicable which makes comparison between models with different implementation possible. However, a small change in a model might not result in any change in precision or recall.

Probabilistic models can be examined on a finer level with tools derived from *information theory*. *Cross entropy* is a measure of how well a probability distribution $\mathrm{q}(x)$ fits a target distribution $\mathrm{p}(x)$ and for discrete distributions it is defined as

$$\mathrm{H}(\mathrm{p},\mathrm{q}) = -\sum_{x} \mathrm{p}(x)\log_2 \mathrm{q}(x) = \mathrm{H}(\mathrm{p}) + \mathrm{D}_{\mathrm{KL}}(\mathrm{p} \mid\mid \mathrm{q}) \tag{4.1}$$

where $\mathrm{H}(\mathrm{p})$ is the entropy of $\mathrm{p}(x)$ and $\mathrm{D}_{\mathrm{KL}}(\mathrm{p} \mid\mid \mathrm{q})$ is the Kullback-Liebler divergence [26] of $\mathrm{q}(x)$ from $\mathrm{p}(x)$. The minimum of $\mathrm{H}(\mathrm{p},\mathrm{q})$ is $\mathrm{H}(\mathrm{p})$ which is reached when $\mathrm{p} = \mathrm{q}$ because $\mathrm{D}_{\mathrm{KL}}(\mathrm{p} \mid\mid \mathrm{p}) = \sum_x \mathrm{p}(x)\log_2 \frac{\mathrm{p}(x)}{\mathrm{p}(x)} = 0$.

The smaller the cross entropy $\mathrm{H}(\mathrm{p}_u,\, \hat{\mathrm{p}}_u)$ of a user selection model $\hat{\mathrm{p}}_u$ the better it is at approximating the underlying system. Even a small improvement in the model is apparent in its cross entropy. The real distribution p_u is unknown, but some selections by the user $x_1, x_2, \ldots, x_N$ have been observed.

The sample distribution is a close approximation of p_u and it can be used to approximate the cross entropy for a model $\hat{\mathrm{p}}_u$ as

$$\hat{\mathrm{H}}(\mathrm{p}_u, \hat{\mathrm{p}}_u) = -\frac{1}{N}\sum_{i}^{N} \log_2 \hat{\mathrm{p}}_u(x_i). \tag{4.2}$$

and is known as the mean cross entropy. For every potentially selectable item the model predicts the probability of it being selected and the sum of probabilities must be one. The smallest mean cross entropy is achieved when $\hat{\mathrm{p}}(x)$ gives uniformly high probability to each item the user selects and very low probability to other items (so that the probability of selected items can be as high as possible) which makes the negative logarithm small.

Information gain is the difference in cross entropy of two models. Often the gain is compared to the uniform distribution. If there are M items from which recommendations are to be made, a uniform distribution assigns the probability $^1/_M$ to each of them, and its cross entropy is $\log_2(M)$. The information gain of $\hat{\mathrm{p}}$ over the uniform distribution is then $\log_2(M) - \hat{\mathrm{H}}(\mathrm{p}, \hat{\mathrm{p}})$. It is possible that a model has negative information gain if it gives high probabilities to items that the user does not select.

Perplexity provides an intuitive interpretation of cross entropy and is often used in natural language processing and information retrieval. It can be defined as

$$\textsc{Perplexity}(\hat{\mathrm{p}}) = 2^{\hat{\mathrm{H}}(\mathrm{p}, \hat{\mathrm{p}})} \tag{4.3}$$

and makes small differences in $\hat{\mathrm{H}}(\mathrm{p}, \hat{\mathrm{p}})$ more pronounced. The perplexity value N is related to the average number of alternatives the model has to consider. A perplexity of 1 indicates that a model always knew which item was going to be selected (and that the system can be described deterministically), and from a set of M items a uniform distribution has a perplexity of M.

4.2.1 Quality Metrics

A recommender system is a personalized window into the content, and if the model is biased the window contains the wrong items. While it might contain some items that are relevant it might be too focused and leave out interesting things. For instance, a user that has read a few sports related news items should still be offered other kinds of news until there is enough evidence that the user is never interested in anything else. The *diversity* of a set of recommendations is how different the items are. It is a measure of how personalized the results are, and in combination with perplexity it can reveal how different the users are.

In a list of recommendations for a single user the *intra-user diversity* is a measure of how similar the L first items in the list are. If the items are too similar, for instance only sports as in the example above, there is a chance that the recommender system is biased and places too strict boundaries on what the user is exposed to. If the items are very different the algorithm might have failed to discover any pattern in the interests of the user. The intra-user diversity depends on features that distinguish items from one another and a distance metric, such as the Jaccard distance

$$\mathrm{D_J}(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \tag{4.4}$$

and using the distance metric it is the average distance among the L first recommendations for every user. The average intra-user diversity can be expressed as

$$\textsc{Diversity}_{\text{intra}}(L) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{2}{L(L-1)} \sum_{i<j} \mathrm{D_J}(x_{ui},\, x_{uj}) \tag{4.5}$$

where $\mathcal{U}$ is the set of all users, x_{ui} is the recommendation at location i in the recommendations for user u, and $i, j \in \{1, 2, \dots, L\}$. The *inter-user diversity* measures how similar the L first recommendations are between two different users on average and is defined as

$$\textsc{Diversity}_{\text{inter}}(L) = \frac{2}{|\mathcal{U}|(|\mathcal{U}|-1)} \sum_{i<j} \mathrm{D_J}(X_{u_i}(L), X_{u_j}(L)) \tag{4.6}$$

where $X_{u_i}(L)$ is the set of the top-L recommendations for user u_i. When this number is close to one the users get very different recommendations, and when it's close to zero the recommendations are nearly the same (although the order might be different) which means that there is very little personalization. The average inter-user diversity of all user pairs and the intra-user diversity of every user for different L gives insight into the overall behavior of the algorithm. [13]

The coverage is a measure of how large a fraction of the content that can be found in at least one user's top-L recommendations and is given by

$$\textsc{Coverage}(L) = \frac{N_L}{N}$$

where N_L is the number of distinct items in the top-L lists of all users and N is the total number of items. The coverage is related to the diversity, and personalization in general. A small coverage indicates that many users receive the same items in their top-L recommendations. The minimum coverage is $^{L}/_{N}$ when every user get the same recommendations and can reach to $^{|\mathcal{U}|L}/_{N}$ when every user get unique recommendations.

4.3 Model Selection

When time and budget is limited and a recommender system is developed for an application for the first time the simplest models yield the greatest return of investment. While they are able to represent more detail, latent variable models such as LDA and PLSI are complex both in terms of comprehensibility and computability. In contrast, simple systems are easier to integrate and maintain. In the spirit of good software engineering practice, a complex model should only be implemented when it has been shown to improve the results of a simpler one that is currently in use.

One example where the improvements became smaller and smaller the more time went on is the Netflix prize competition. The goal of the Netflix prize was to create a model that predicts what rating a user is going to give to movies and television series. Entries were compared against a target model, called Cinematch, and the winner was the first one to improve upon it by having a 10% smaller error. The user rating that Cinematch predicted was 9.6% better than an item's average rating, which is similar to the item popularity in section 2.2.3. There were hundreds of competing teams and initial progress in the competition was very rapid. After a few months several entries achieved 6% improvements and after a year some had attained a model that had 8% smaller error. [6] However, further progress was slow and it took two more years to close in on the remaining 2%. [5] There were diminishing returns the more complex the models got and the even very simple models provided significant gains. For instance, it was shown that a model based on a linear combination of a the average rating a user makes and the average item rating was only 3% worse than Cinematch [32, p. 317].

In a paper outlining the methods and strategy used for winning the first Netflix progress prize, Bell and Koren [4] remark on the importance of using several different models in combination rather than optimizing a single one. Some models were able to find very local details, such as finding pairs of movies that often get a similar rating, while other models found more abstract similarities among groups of movies such as mood, theme or genre. No single model was able to find both abstract and concrete patterns, but together the models complemented each other.

Overall it is a common idea to build up a model out of smaller pieces, each of which focus on one particular aspect of what a user might want. Billsus and Pazzani [8] used two separate models in their news recommender system *News Dude*. One model captured the user's short term interests and another model the long term and they used very different implementation strategies. Together they a smaller error than either model used on their own.

As mentioned earlier, Google News also uses a combination of collaborative filtering models and a content-based model [15, 28]. In a news recommendation contest Lommatzsch [29] implemented several models and compared their performance and accuracy against each other. It was not clear that any single model was inherently better than others, and in different contexts some models worked better than others.

4.4 Experimental Model Development

With simplicity and modularity in mind, a model for news recommendation was experimentally developed and the starting point was to create an environment for evaluating models. The test bench was a simple software utility that loaded a data set containing user selections (described in appendix A) and compared the results of models in terms of perplexity and information gain. Step-by-step, a model made a prediction of what a user is going to select at a given point in time which was compared against what the user actually selected. At the end of each step the parameters were tuned with some of the selections a user has made. A good model should give high probabilities to items a user selects and low probability to other ones.

Using Bayes' theorem and the chain rule the probability $\mathrm{p}(a \mid u, \theta)$ could be split into smaller constituent parts. For instance, from the data set it was observed that some users tend to select articles that contained certain key words or articles that have recently been published, and these phenomena were modeled as aspects of the probability. The test bench made it possible to iteratively improve upon a model by testing it and observing its behavior. Based on the results it was hypothesized how it should be modified, and then it could be tested again to validate to discard the hypothesis. This test and modification cycle was fast. For most models it was possible to get the results of a change within seconds which encouraged experimentation. Comparing the results of different models also gave insight into how they relate to one another and where dependencies lie.

Chapter 5

Models

A recommender system can be based on a probabilistic model, as stated in section 2.2.3. When user selections are used as implicit ratings, it becomes natural to also specify the event space as user selections, and then the recommender system needs a model for the probability $\mathrm{p}(a \mid u, \theta)$, that article a is selected by user u in context θ. In this chapter a model is described which estimates this probability.

The only contextual information that was used in the model was time. This makes the target probability $\mathrm{p}(a \mid u, t)$. Using Bayes' theorem and the chain rule, the conditional probability can be restated a

$$\mathrm{p}(a \mid u, t) = \frac{\mathrm{p}(u \mid a, t)\mathrm{p}(t \mid a)\mathrm{p}(a)}{\mathrm{p}(u, t)} \tag{5.1}$$

which was used as a starting point for the algorithm. Disregarding the joint probability $\mathrm{p}(u, t)$ in the denominator for a moment, the user u is mentioned only once. This leads to an algorithm that consists of two primary parts

$$\underbrace{\mathrm{p}(u \mid a, t)}_{\text{personalized}} \; \underbrace{\mathrm{p}(t \mid a)\mathrm{p}(a)}_{\text{non-personalized}}$$

where the non-personalized probabilities $\mathrm{p}(t \mid a)\mathrm{p}(a) = \mathrm{p}(t, a)$ reflect the probability that an article is selected at the current time by any user in general, and the personalized part represents the probability that user u in particular makes that selection.

A simplifying assumption was made that the probability that a user makes a selection is conditionally independent of time such that

$$\mathrm{p}(u \mid a, t) = \mathrm{p}(u \mid a). \tag{5.2}$$

Even so, most users only read a news article once, which makes the probability $\mathrm{p}(u \mid a)$ useless once it has been observed. Instead it was modeled based on the content of the article, which makes the recommender system content-based. Each article was originally published in a magazine or newspaper, which is hereafter referred to the source s_a of the article. Additionally, for every article a there was a vector of binary features $\boldsymbol{f}_a$ where the value in each dimension indicate the presence or absence of a word or a combination of words.

Seen as random variables, the source S and feature $\boldsymbol{F}$ are fully determined when conditioning on an article and therefore the probability

$$\hat{\mathrm{p}}(u \mid a) = \mathrm{p}(u \mid s_a, \boldsymbol{f}_a) \tag{5.3}$$

was used in to estimate equation 5.2. This was further simplified by assuming that the article features $\boldsymbol{f}_a$ and the source s_a of an article were mutually independent and that conditioned on a user the features are independent of the source. By applying Bayes' theorem to equation 5.3, the personalized probability

$$\mathrm{p}(u \mid s_a, \boldsymbol{f}_a) = \frac{\mathrm{p}(s_a, \boldsymbol{f}_a \mid u)\mathrm{p}(u)}{\mathrm{p}(s_a, \boldsymbol{f}_a)} = \frac{\mathrm{p}(s_a \mid u)}{\mathrm{p}(s_a)} \frac{\mathrm{p}(\boldsymbol{f}_a \mid u)}{\mathrm{p}(\boldsymbol{f}_a)} \mathrm{p}(u) \tag{5.4}$$

is turned into two independent conditional probabilities of the article properties.

The algorithm consists of four models that each capture a particular aspect of what makes an article interesting. During a selection the value of the following five random variables can be observed:

A	The article that was selected, which is one of the set of articles in $\mathcal{A}$.
U	The user in $\mathcal{U}$ who made the selection.
T	The current time. It is used to determine the age of an article based on the time t_a when it was published.
S	The source of an article, which is one of the possible sources in $\mathcal{S}$. The source is determined when conditioned on an article.
$\boldsymbol{F}$	The article text feature vector. Each feature F_i indicate the presence or absence of a single word or a combination of words in the text. Just like the source, the features are determined when conditioned on an article.

In addition, every model algorithm implicitly depend on the selections other users have made which has been left out from the notation.

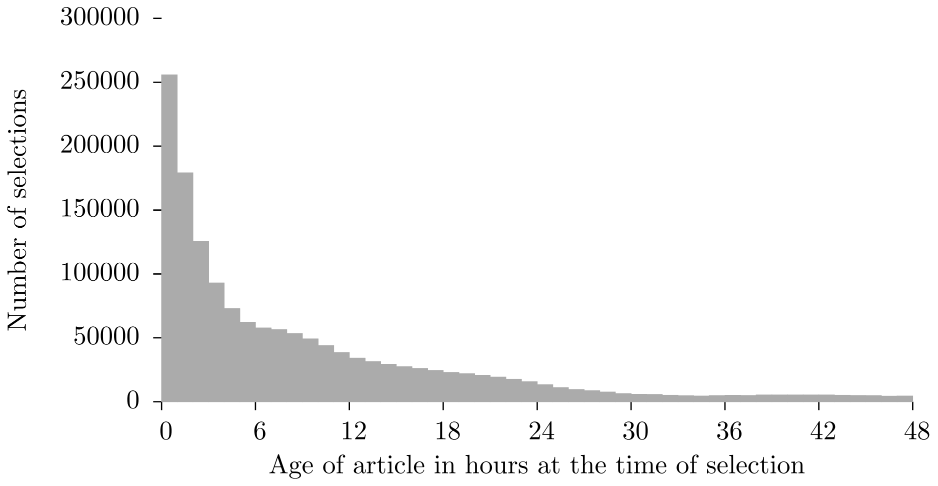


Figure 5.1: Histogram of user selections in the data set when the age of the article was less than 48 hours at the time of selection. Each bin is an hour wide. About 12% of the selections in the data set were of articles that were more than 48 hours old.

5.1 Age

The conditional probability $p(t \mid a)$ is the probability that a selection is made at time t when the selected article is a. Intuitively, an article becomes less interesting the older it gets because it ceases to be news and turns into history. It follows then that old articles are selected with a lower probability than recent ones and this is the subject of the *age* model.

Using the data set described in appendix A, which contained selections by real users, a histogram of article selections was made which is shown in figure 5.1. Each bin b_n in the histogram contained the number of selections in the data set that had occurred when the selected article was between n and $n + 1$ hours old. The number of selections per bin seemed to roughly follow a exponential or geometric progression with a few bumps here and there. Other histograms were made, each of which contained selections from only one source, and they were found to exhibit similar curves. For some sources the age of an article was less important and for others very few selections were made after an article was only a day or two old. While there are many factors that affects how long an article stays relevant, such as contents and context, magazine articles were generally more long lived than newspaper articles. Based on this pattern, the selection probability at a certain age was modeled to only depend on the source of an article.

If a user had an opportunity of selecting an article each hour with probability ϕ but didn't select it until the nth hour, then the geometric distribution can represent this scenario. The geometric distribution has the probability mass function

$$f(t; \phi) = (1 - \phi)^t \phi$$

for $t \in \{0, 1, 2, 3, \dots\}$ and $\phi \in (0, 1)$. When $t = 0$ an article is selected with probability ϕ and the closer to one it is, the more probable it is that it will be selected for small t. A small value of ϕ makes the progression flatter and the difference in probability at t and $t + 1$ is smaller.

Each source has its own parameter ϕ_s which can be estimated from selections in the data set. The maximum likelihood estimator of ϕ from the sample $x_1, \dots, x_N$ of non-negative integers (selection counts per hour interval) is

$$\hat{\phi} = \frac{N}{\sum_i x_i + N} \tag{5.5}$$

The parameter values of the sources that were estimated from the data set lie in the range $[0.02, 0.15]$.

The source $S = s_a$ is fully determined when conditioning on article a, which specifies the model parameter ϕ. The conditional probability $\mathrm{p}(t \mid a)$ is estimated as

$$\hat{\mathrm{p}}(t \mid a) = \mathrm{p}(t \mid t_a, \phi_a) = f(t - t_a; \phi_{s_a}) = (1 - \phi_{s_a})^{(t - t_a)} \phi_{s_a} \tag{5.6}$$

where t is the current time in hours and t_a is the time that article a was published.

While the support for the geometric distribution are the natural numbers, if t is allowed to also contain fractions of an hour the test results turned out to improve. This misuse can arguably be justified due to the way the probabilities are normalized, which is covered in chapter 6, but also when considering a related exponential variable $X \sim \mathrm{Exp}(\lambda)$. Its probability over a unit length (one hour) is

$$\Pr(t \leq X \leq t + 1) = \int_t^{t+1} \lambda e^{-\lambda x} \mathrm{d}x = e^{-\lambda t}(1 - e^{-\lambda}) = (1 - \phi)^t \phi$$

for $\lambda = -\ln(1 - \phi)$ and $t \in [0, \infty)$.

5.2 Popularity

As discussed in section 2.2.3, the popularity of an article a, i.e. the total number of times it has been selected, is an indicator of its overall quality and desirability, and is proportional to the probability $\mathrm{p}(a)$ that it is selected by any user. In equation 5.1 it is used as the prior probability of an article and is estimated by

$$\hat{\mathrm{p}}(a) = \frac{n_a}{\sum_i n_i}$$

where n_x is the number of times article x has been selected. This probability forms a categorical distribution over all the articles.

The view count of an article accumulates over time, however, which makes it difficult to compare the view counts of two items published at different points in time. This is an instance of the cold start problem since new articles have had little time to accumulate views. The task of the popularity model is to approximate what the value of n_a will be at a point in time far in the future, with the hope that old and new articles can be compared fairly.

One way to estimate the rate of accumulation is to assume selections are independent Bernoulli trials where each trial has a small probability of success q. During the time an article is relevant (i.e. when people still read it) there is a very large number n of chances that a selection will be made which is proportional to the number of users. The total selection count after a long period of time, after every trial has occurred can then be estimated with the binomial distribution

$$\hat{N}_a \sim \mathrm{Binomial}(n, q). \tag{5.7}$$

The age model suggests that after t hours an article has accumulated a fraction of its total number of selections that is proportional to

$$F_T(t) = 1 - (1 - \phi)^{t+1} \tag{5.8}$$

which is the cumulative distribution function of a geometric distribution with parameter ϕ. One property of the binomial distribution is that if X and Y are binomially distributed with the same probability p but different number of trials x and y respectively, then $X + Y \sim \mathrm{Binomial}(x + y, p)$. Using this property it's possible take the gradual accumulation into account by splitting $\hat{N}_a$ into two different variables

$$\begin{aligned} L_{at} &\sim \mathrm{Binomial}(\lfloor nF_T(t) \rfloor, q) \\ M_{at} &\sim \mathrm{Binomial}(n - \lfloor nF_T(t) \rfloor, q) \end{aligned}$$

where L_{at} is the number of times a has been selected up until until time t, and M_{at} is the number of selections that remain, and at any point in time $\hat{N}_a = L_{at} + M_{at}$. The expected value is $\mathrm{E}[\hat{N}_a] = \mathrm{E}[L_{at} + M_{at}] = nq$ and the variance of the binomially distributed variable is $\sigma^2_{\hat{N}_a} = \mathrm{Var}(\hat{N}_a) = nq(1-q)$.

If, however, q is very small then $\mathrm{Var}(\hat{N}_a) \approx nq$. Given the evidence that L_{at} has the value l_{at} at time t, the conditional expectation is

$$\mathrm{E}[L_{at} + M_{at} \mid L_{at} = l_{at}] = l_{at} + (n - \lfloor nF_T(t) \rfloor)q. \tag{5.9}$$

and because the parameter n is a large integer the error is very small when assuming that $n - \lfloor nF_T(t) \rfloor \approx n(1 - F_T(t))$, and by approximating $l_{at} \approx F_T(t)\mathrm{E}[N_a] = F_T(t)nq$ then $n(1 - F_T(t))q = nq - l_{at} = {}^{l_{at}}/_{T_F(t)} - l_{at}$. Using these approximations the conditional expectation can be written as

$$\mathrm{E}[L_{at} + M_{at} \,|\, L_{at} = l_{at}] \approx l_{at} + l_{at}\frac{1 - F_T(t)}{F_T(t)} = \frac{l_{at}}{F_T(t)} \tag{5.10}$$

which is an extrapolation of the observed value of L_{at}. As t grows $F_T(t)$ approaches 1 and the extrapolation becomes less and less significant and the expected value becomes the observed value. The conditional variance

$$\mathrm{Var}(L_{at} + M_{at} \,|\, L_{at} = l_{at}) = (n - \lfloor nF_T(t) \rfloor)q(1 - q) \tag{5.11}$$

approaches zero as $F_T(t)$ goes to 1. Intuitively, if the age model is accurate then most of the selections for article a has been made after a certain amount of time has passed, and the observed value at that time is close to the true value. By extrapolating and assuming $nq(1 - q) \approx nq$ the variance becomes

$$\mathrm{Var}(L_{at} + M_{at} \,|\, L_{at} = l_{at}) \approx \frac{l_{at}(1 - F_T(t))}{F_T(t)}. \tag{5.12}$$

At time $t = 0$ the denominator $F_T(0) = \phi$ in equation 5.12 is at its smallest, and during the first few hours the difference between any $F_T(k)$ and $F_T(k+1)$ will be the greatest. As time goes on the effect becomes less severe.

The extrapolation can have very chaotic results soon after an article has been published because the number of selections l_{at} is rapidly increasing. Until enough time has passed it's reasonable to guess that the view count will be similar to those of similar articles. The sample mean of the view counts of the articles $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ from source s is

$$\overline{n}_s = \frac{1}{m}\sum_{i=1}^{m} n_{\alpha_i}$$

and is used in the popularity model as the initial guess of what the value of n_a will be.

The two estimators $\hat{n}_{E_a} = \mathrm{E}[L_{at} + M_{at} \,|\, L_{at} = l_{at}]$ and $\overline{n}_s$ for n_a can be combined through a convex combination

$$\hat{n}_a = w\hat{n}_{E_a} + (1 - w)\overline{n}_s \tag{5.13}$$

which is a kind of shrinkage estimator where the weight $w \in [0,1]$ can be arbitrarily chosen, but should represent the level of confidence in $\hat{n}_{E_a}$. The variance of $\hat{N}_a$ decreases as time passes, which means that the confidence in it should increase. This suggests that w could be defined as a function of $\sigma^2_{\hat{N}_a}$.

Multiplying $\overline{n}_s$ by the standard deviation $\sigma_{\hat{N}_a}$ of $\hat{N}_a$ has the desired effect, but then the two estimators have different units of measure and an interpolation of the values does not make sense. This can be fixed by scaling $\hat{n}_{Ea}$ by the standard deviation of the distribution that $\overline{n}_s$ represents. This is essentially the same as normalizing random variables to have unit standard deviation and variance, which makes them comparable. To fulfill the requirement that $w \in [0,1]$ both standard deviations are divided by their sum, which gives the expression for w

$$w = \frac{1/\sigma_{\hat{N}_a}}{1/\sigma_{\hat{N}_a} + 1/\sigma_{\overline{n}_a}} = \frac{1}{1 + \sigma_{\hat{N}_a}/\sigma_{\overline{n}_a}} \tag{5.14}$$

which is normalized and approaches one as $\sigma_{\hat{N}_a}$ goes to zero. Using similar assumptions as above, if $N_{\alpha_i} \sim \text{Binomial}(\nu, \pi)$ where ν is very large and π is very small – the distribution from which N_a is initially assumed to be drawn from – then $\sigma_{\overline{n}_a}$ be approximated as

$$\sigma_{\overline{n}_s} = \sqrt{\nu\pi(1-\pi)} \approx \sqrt{\overline{n}_s}. \tag{5.15}$$

For a news aggregator that bases its selection counts on the selections made by the readers of each publication, there is another problem that has to be dealt with. Unless every publication have the same number of readers, the selection counts of items from large publications will generally be greater than even the most popular item from a small source, and the probability would be biased in favor of large publications. If a news aggregator want to treat publications equally then the selection counts need to be scaled to reflect the aggregator's users' behaviors. A normalizing factor κ_a is introduced which depends on the source s of the article and is defined as

$$\kappa_a = \frac{M_{gs}}{M_{ps}} \tag{5.16}$$

where M_{gs} is the total number of selections of articles from source s among the aggregator's users and M_{ps} is the total number of selections of items used by the aggregator that were made by the publication's users.

The normalizing factor combined with equation 5.13 can be turned into an estimator for $\mathrm{p}(a)$ and is given by

$$\hat{p}(a) = \frac{\kappa_a \hat{n}_a}{\sum_x \kappa_x \hat{n}_x} \tag{5.17}$$

5.3 Source

In equation 5.4 the probability that a user makes a selection conditioned on the article $\hat{p}(u \mid a)$ is modeled in terms of the article properties s_a and $\boldsymbol{f}_a$. The source model is an estimator of the conditional probability $p(s_a \mid u)$ and the source probability $p(s_a)$ and provides a first step of personalization for each user. Every time a user reads an article, the event is recorded in a user specific list of selections called the user's selection history. The selections a user has made in the past are used to predict future selections. Like the popularity model it suffers from the cold start problem because for new users there is very little evidence of what their true preferences might be.

For a new user not much can be assumed about the conditional probability $p(u \mid a)$. A first assumption might be to assume that the user has no particular preference for different sources and that it equals $p(u)$, i.e. that the probability of the user making a selection is independent of the article's source. In equation 5.4 it becomes evident that this can be achieved defining $p(s_a \mid u)$ such that it starts out from the probability $p(s_a)$ for new users and converges towards a personalized model as the user makes selections. In this form the quotient ${}^{p(s_a \mid u)}/_{p(s_a)}$ can be interpreted as the ratio by which the user's interests differ from the average [28]. For instance, if the quotient is 2 then a user is twice as likely to select an item from source s_a than the average user.

When the sample space consist of the selections then the probability of finding an article from source s in user u's selection history can be estimated from the selection frequency

$$p(s \mid u) = \frac{p(s,u)}{p(u)} \approx \frac{{}^{n_{us}}/_{n}}{{}^{n_u}/_{n}} = \frac{n_{us}}{n_u} = \hat{p}(s \mid u) \tag{5.18}$$

where n_{us} is the number of articles from site s in the selection history of user u, n_u is the total number of selections by the user, and n is the total number of selections by all users. There's a separate probability for every source and combined they form a categorical distribution and $\sum_s p(s \mid u) = 1$. This conditional probability is used to estimate whether a user is going to select an item from that source in the future.

The probability of a selection of source s in general is

$$p(s) = \sum_u p(s \mid u)p(u) \approx \frac{n_s}{n} = \hat{p}(s) \tag{5.19}$$

where $n_s = \sum_u n_{us}$ is the number of selections by all users of articles from source s. However, due to the small number of users the application has the variables n_s and n were approximated by the number of articles per site, i.e.

$$\hat{n}_s = |\{ \text{ articles from source } s \ \}|$$
$$\hat{n} = \sum_s \hat{n}_s$$

which were used in place of n_s and n, until the number of users increase. This has the effect that each article is equally likely to be selected and the prior probability for each site is based on its volume of articles.

Like the popularity model, $\hat{\text{p}}(s \mid u)$ and $\hat{\text{p}}(s)$ are combined with a convex combination defined as

$$\hat{\text{p}}'(s \mid u) = w\hat{\text{p}}(s \mid u) + (1 - w)\hat{\text{p}}(s) \tag{5.20}$$

where the weight $w \in [0, 1]$ again represents the confidence in $\hat{\text{p}}(s \mid u)$. The confidence is related to the standard error of $\hat{\text{p}}(s \mid u)$ which, if selections are i.i.d. Bernoulli trials, is inversely proportionally to $\sqrt{n_u}$. While the standard error can be estimated from the sample standard deviation, it is likely to be zero for many sources until the user has made enough selections, and therefore it is estimated by a constant. Based on experimentation w was chosen to be a simple function of n_u defined as

$$w = \frac{1}{1 + {}^{K}\!/_{n_u \hat{\text{p}}(s)}} \tag{5.21}$$

where n_u, the source probability $\hat{\text{p}}(s)$ and a constant K all decide how quickly w converges to 1. A large value of the prior probability means that a user is likely to select articles from that source and not too many samples are needed to be confident in $\hat{\text{p}}(s \mid u)$. A small prior, on the other hand, suggests that a larger n_{us} is needed before it is representative of the true probability. The constant K sets the overall convergence rate and in the model it is set to $^1\!/_4$.

5.4 Text Features

The text feature model is an estimator of $\text{p}(\boldsymbol{f}_a \mid u) = \text{p}(f_{a1}, f_{a2}, \ldots, f_{an} \mid u)$ and $\text{p}(\boldsymbol{f}_a)$ from equation 5.4, which is the probability that user u makes a selection given the state of a set of binary features $f_{ai} \in \{0, 1\}$ that are associated with article a. For two articles a and a' it is possible that $\boldsymbol{f}_a = \boldsymbol{f}_{a'}$ in which case they both result in the same probability. The features are based on keywords, such as whether the text contains the name of a famous person, location, or uses a technical term. The text feature model is personalized and reflects a user's tendency to select articles with certain features.

Like the source and popularity models, the feature model also suffers from the cold start problem and similar strategies are used to find a definition for $\mathrm{p}(\boldsymbol{f} \mid u)$ that works well for new users. For a new user the conditional probability $\mathrm{p}(\boldsymbol{f} \mid u)$ should not make any assumptions about the user's preferences and is equal to $\mathrm{p}(\boldsymbol{f})$. There are more features than sources which means that there is a greater number of variables to estimate. This in turn means that the rate of personalization for the text feature model is going to be slower.

If an article contains the words "cat" and "furniture" which are quite unrelated, it is reasonable to assume that a user who is interested in pets and home decoration is going to find the article particularly intriguing. If one feature does not imply another it is possible to make the naïve independence assumption on $\mathrm{p}(\boldsymbol{f} \mid u)$ such that

$$\mathrm{p}(\boldsymbol{f} \mid u) = \mathrm{p}(f_1, f_2, \ldots, f_n \mid u) = \prod_i \mathrm{p}(f_i \mid u) \tag{5.22}$$

which simplifies estimation because only n probabilities need to be estimated rather than one for each of the 2^n possible states the features can take. The probability distribution in equation 5.22 is a multivariate Bernoulli distribution, and for a single feature F_i there is one parameter θ_i such that

$$\mathrm{p}(F_i = k \mid u) = \theta_i^k (1 - \theta_i)^{1-k}$$

for $k \in \{0, 1\}$. The parameter θ_i is a user specific and can be estimated from the feature frequencies among articles in the user's selection history. If there are many different features the probability $\hat{\mathrm{p}}(F_i = 1 \mid u) = \hat{\theta}_i = {}^{n_i}/_{n}$ is small or even zero for most users. The probability distribution of feature F_i is calculated by marginalizing over the users

$$\mathrm{p}(F_i) = \sum_j \mathrm{p}(F_i \mid u_j)\mathrm{p}(u_j). \tag{5.23}$$

The feature model also gradually moves from $\hat{\mathrm{p}}(\boldsymbol{f})$ to $\hat{\mathrm{p}}(\boldsymbol{f} \mid u)$ using a convex combination

$$\hat{p}'(\boldsymbol{f} \,|\, u) = w\hat{\mathrm{p}}(\boldsymbol{f} \mid u) + (1 - w)\hat{\mathrm{p}}(\boldsymbol{f})$$

where, behaviorally similar to equation 5.21, w is defined as

$$w = \frac{1}{1 + {}^{K}/_{n_i \hat{\mathrm{p}}(f_i)}} \tag{5.24}$$

with K set to 4 based on experiments.

5.4.1 Feature Selection

If features are not independent and the assumptions in equation 5.22 do not hold, then the results will be biased. For instance, the words “yarn” and “knitting” are very related and convey much of the same information. If the words are taken to be independent then the probability will be biased for articles that contain both of them unless the user really thinks that an article that contains these two words is better than one that only contains one of them. In equation 5.4 there is also the assumption that features and sources are independent. One can imagine that a car magazine uses car related terms more often than a beauty magazine. Feature selection is a useful and often necessary step where only the best features are kept and the rest are ignored in the model.

Mutual information is an information theoretic quantity that can be used to compare the level of dependence between random variables. Probabilities for sources and features are needed for its calculation. The random variables $\tilde{F}_i$, $i \in \{1, 2, \ldots, m\}$ are augmented features, and in addition to all the binary text features, there are binary variables that indicate the source of an article. The event space of the probabilities of these features consist of articles. The probability that a single augmented feature is found in an article is $\mathrm{p}(\tilde{F}_i)$ which can be estimated from its frequency of occurring in articles in the data set. Similarly, it is possible to estimate the joint probability $\mathrm{p}(\tilde{F}_i, \tilde{F}_j)$ of text features and sources occurring in the same article. Sources are disjoint, however, and $\mathrm{p}(\tilde{F}_y, \tilde{F}_x) = 0$ when x and y represent different sources.

The augmented binary feature variables are discrete and binary and their pairwise mutual information is defined as

$$\mathrm{I}(\tilde{F}_i; \tilde{F}_j) = \sum_{x=0}^{1} \sum_{y=0}^{1} \mathrm{p}(\tilde{F}_i = x, \tilde{F}_j = y) \log_2 \frac{\mathrm{p}(\tilde{F}_i = x, \tilde{F}_j = y)}{\mathrm{p}(\tilde{F}_i = x)\mathrm{p}(\tilde{F}_j = y)}.$$

Feature selection can be performed using this equation by incrementally expanding the set of features $\mathcal{F}$ that are used in the model. Initially the set contains the source features, since every article have a source and the source model depends on it. Then one by one, starting from the most common text feature, the relative sum of pairwise mutual information with items already in $\mathcal{F}$ was calculated for each text feature i as

$$m_i = \frac{\sum_{j \in \mathcal{F}} \mathrm{I}(\tilde{F}_i; \tilde{F}_j)}{\mathrm{H}(\tilde{F}_i)}$$

where $\mathrm{H}(\tilde{F}_i) = \sum_{x=0}^{1} \mathrm{p}(\tilde{F}_i = x) \log_2 \mathrm{p}(\tilde{F}_i = x)$ is the entropy of $\tilde{F}_i$. If m_i was less than a specific threshold then the feature was added to $\mathcal{F}$. On the following iterations the newly added text feature $\tilde{F}_i$ affects whether subsequent features are included or not.

5.5 Generating Recommendations

The probability $\mathrm{p}(a \mid u, t)$ from equation 5.1 is calculated for every article a in the set of candidate articles $\mathcal{A}$ whenever user u requests a list of recommendations. The equation is estimated using using the models described in this chapter which yields

$$\hat{\mathrm{p}}(a \mid u, t) = \frac{\hat{\mathrm{p}}(u \mid s_a, \boldsymbol{f}_a)\hat{\mathrm{p}}(t \mid a)\hat{\mathrm{p}}(a)}{\hat{\mathrm{p}}(u, t)} \tag{5.25}$$

and after expansion

$$\hat{\mathrm{p}}(a \mid u, t) = \frac{\hat{\mathrm{p}}'(s_a \mid u)}{\hat{\mathrm{p}}(s_a)} \frac{\hat{\mathrm{p}}'(\boldsymbol{f}_a \mid u)}{\hat{\mathrm{p}}(\boldsymbol{f}_a)} \frac{\hat{\mathrm{p}}(u)\hat{\mathrm{p}}(t \mid a)\hat{\mathrm{p}}(a)}{\hat{\mathrm{p}}(u, t)}. \tag{5.26}$$

where, given that $\hat{\mathrm{p}}(u \mid t, a) \overset{\text{def}}{=} \hat{\mathrm{p}}(u \mid a) \overset{\text{def}}{=} \hat{\mathrm{p}}(u \mid s_a, \boldsymbol{f}_a)$, values for

$$\hat{\mathrm{p}}(u) = \sum_{\alpha \in \mathcal{A}} \hat{\mathrm{p}}(u, \alpha) = \sum_{\alpha \in \mathcal{A}} \hat{\mathrm{p}}(u \mid \alpha)\hat{\mathrm{p}}(\alpha) \tag{5.27}$$

and

$$\hat{\mathrm{p}}(u, t) = \sum_{\alpha \in \mathcal{A}} \hat{\mathrm{p}}(u, t \mid \alpha)\hat{\mathrm{p}}(\alpha) = \sum_{\alpha \in \mathcal{A}} \hat{\mathrm{p}}(u \mid \alpha)\hat{\mathrm{p}}(t \mid \alpha)\hat{\mathrm{p}}(\alpha) \tag{5.28}$$

are given. During one recommendation request they are both constants, however, and only normalizes the result so that $\hat{\mathrm{p}}(a \mid u, t)$ is a probability value. There is also a normalizing factor in the denominator of equation 5.17 that defines the article probability $\hat{\mathrm{p}}(a)$, and if all constant normalization is left out of equation 5.25 the expression

$$\hat{\mathrm{p}}(a \mid u, t) \propto \frac{\hat{\mathrm{p}}'(s_a \mid u)}{\hat{\mathrm{p}}(s_a)} \frac{\hat{\mathrm{p}}'(\boldsymbol{f}_a \mid u)}{\hat{\mathrm{p}}(\boldsymbol{f}_a)} \hat{\mathrm{p}}(t \mid a)\kappa_a \hat{n}_a \tag{5.29}$$

is proportional to the probability. If values that are proportional to the probability are enough, normalization can be left out. For instance, sorting a list of recommendations according to proportional probability values will place items in the same order as if they were normalized.

5.5.1 Using Models Individually

Each model can be used on its own, which is useful when inspecting the its behavior. The popularity model is straightforward to use by assuming that A is independent of both U and T and

$$\hat{\mathrm{p}}_{\mathrm{p}}(a \mid u, t) = \hat{\mathrm{p}}(a). \tag{5.30}$$

For the other models one can assume that the probability that an article is selected is uniform so that $\mathrm{p}(A)$ takes the form of a uniform distribution $\hat{\mathrm{p}}_{\mathrm{u}}(a) = {}^{1}\!/_{|\mathcal{A}|}$. By making convenient conditional independence assumptions equation 5.25 can be turned into several model combinations.

Using the uniform probability for the article prior, the age model $\hat{\mathrm{p}}_{\mathrm{a}}(a \mid t)$ can be used on its own when defined as

$$\hat{\mathrm{p}}_{\mathrm{a}}(a \mid u, t) \propto \hat{\mathrm{p}}(t \mid a)\hat{\mathrm{p}}_{\mathrm{u}}(a). \tag{5.31}$$

The probability that an article is selected is determined in this model by its age and the rate at which articles from its source loses their relevance. In a list of recommendations old articles are placed at the end and new ones at the front, but an item that stays relevant for longer can get a higher probability than a more recent item with a short shelf life. Using $\hat{\mathrm{p}}(a)$ instead of the uniform distribution creates the combined age and popularity model

$$\hat{\mathrm{p}}_{\mathrm{a}}(a \mid u, t) \propto \hat{\mathrm{p}}(t \mid a)\hat{\mathrm{p}}(a). \tag{5.32}$$

This combination is sometimes used as the definition of a base line popularity model because the selections are weighted by age.

The personalized models can be used individually as well if articles are equiprobable and independent of time. The source model $\hat{\mathrm{p}}_{\mathrm{s}}(a \mid u)$ is defined as

$$\hat{\mathrm{p}}_{\mathrm{s}}(a \mid u, t) \propto \frac{\hat{\mathrm{p}}'(s_a \mid u)\hat{\mathrm{p}}_{\mathrm{u}}(a)}{\hat{\mathrm{p}}(s_a)}. \tag{5.33}$$

It ranks articles by their source alone and sorts the set of candidate articles such that every item from the most probable source comes first, items from the second most probable source comes second, and so on. In a similar way the text feature model $\hat{\mathrm{p}}_{\mathrm{f}}(a \mid u)$ is defined as

$$\hat{\mathrm{p}}_{\mathrm{f}}(a \mid u, t) \propto \frac{\hat{\mathrm{p}}'(\boldsymbol{f}_a \mid u)\hat{\mathrm{p}}_{\mathrm{u}}(a)}{\hat{\mathrm{p}}(\boldsymbol{f}_a)}. \tag{5.34}$$

The exact behavior of this probability is harder to predict. Generally, articles near the top of the list of recommendations have features f_{ai} in states x_i such that $\hat{\mathrm{p}}(u \mid F_i = x_i) > \hat{\mathrm{p}}(u \mid F_i = 1 - x_i)$ and for articles at the bottom the reverse is true. More than one article can get the same probability and such articles might appear in any order. Together the personalized models form the source and text feature model

$$\hat{\mathrm{p}}_{\mathrm{sf}}(a \mid u, t) \propto \frac{\hat{\mathrm{p}}'(s_a \mid u)\hat{\mathrm{p}}'(\boldsymbol{f}_a \mid u)\hat{\mathrm{p}}_{\mathrm{u}}(a)}{\hat{\mathrm{p}}(s_a)\hat{\mathrm{p}}(\boldsymbol{f}_a)} \tag{5.35}$$

5.5.2 Complexity Analysis

Each source has a probability $\hat{\mathrm{p}}(s)$ and an age model parameter $\hat{\phi}_s$. The selections themselves are not needed at run-time and these parameter estimations can be performed offline. There is also a probability $\hat{\mathrm{p}}(f_i)$ for each feature, which can also be estimated offline with a data set of selections and articles. Every article has a source, $|\mathcal{F}|$ features and the current number of article selections l_{at}, which has to be kept up to date. The personalized models need user specific parameters for $\hat{\mathrm{p}}(s \mid u)$ and $\hat{\mathrm{p}}(f_i \mid u)$. They can be calculated from a single pass over the user selection history. Alternatively, the counts n_{ui}, n_{us} and n_u could be stored individually and incremented after every selection. The selections themselves are still useful to keep for possible extension of the model and for estimating ϕ_s, $\mathrm{p}(s)$ and $\mathrm{p}(f_i)$. In total there are $2|\mathcal{S}| + |\mathcal{F}| + |\mathcal{A}|(|\mathcal{F}| + 2)$ non-personalized parameters and variables, and $|\mathcal{S}| + |\mathcal{F}|$ parameters per user.

Equation 5.25 can be calculated in constant time since the sizes of $\mathcal{F}$ and $\mathcal{S}$ are fixed. When recommendations are generated for a user, the equation is evaluated for every article a in $\mathcal{A}$, which takes $O(|\mathcal{A}|)$ time. Then the list is sorted, which can be done in $O(|\mathcal{A}| \log |\mathcal{A}|)$ time on average using the quicksort algorithm [14]. A list of probabilities (and article ids) is needed to sort the articles which makes the memory requirements when generating a single list of recommendations $O(|\mathcal{A}|)$. Only the parameters of the user in question are needed to evaluate the equation, and these parameters could potentially be stored separately from the articles.

If there are too many article parameters to fit in main memory, these could be placed in secondary storage. With disk pre-fetching the performance will not be too much hampered if the article probabilities are calculated serially as they appear on disk. If the number of articles is extremely large, then the set of articles $\mathcal{A}$ can even be sharded onto k separate machines. The unnormalized algorithm in equation 5.29 can be run in parallel on each shard and the partial results can be merge sorted to get the top recommendations in $O(|\mathcal{A}| \log |\mathcal{A}|)$ time [14].

Chapter 6

Results

In this chapter the results of model tests are presented and superficially interpreted to highlight the most interesting aspects. A more thorough interpretation is given in chapter 7.

6.1 Perplexity

Every model and model combination in section 5.5.1 were evaluated in the environment described in section 4.4 by simulating user interaction from selections in the data set described in appendix A. The primary target was to optimize for minimal cross entropy which is presented here in the form of perplexity. In these tests 32 users were randomly selected from the data set. For each of them a test set of hundreds of selections was made. For every test selection x, the 1024 most recent articles were collected into the set $\mathcal{V}_x$ that represents the articles that potentially were visible to the user at the time. The item probabilities were calculated for every item in the test set, conditioned on the user having to choose from articles in $\mathcal{V}_x$. Using these item probabilities the perplexities given by equation 4.3 were evaluated.

The calculation is slightly erroneous because the probabilities of x_i were generally conditioned on a different set of recent items $\mathcal{V}_{x_i}$. For instance, for some test selection x_i the recent items might have contained a lot of very interesting articles, which would make the probability $\hat{\mathrm{p}}(a_{x_i} \mid u, t_{x_i})$ of a user model smaller than if $\mathcal{V}_{x_i}$ contained only very old and uninteresting items. With a large number of test selections and by averaging over many users the noise is reduced and the average behavior appears.

Once the perplexity was calculated, one test sample was moved to the set of training samples and the perplexity was calculated once again with the adjusted test set, which no longer contained the train sample. This procedure

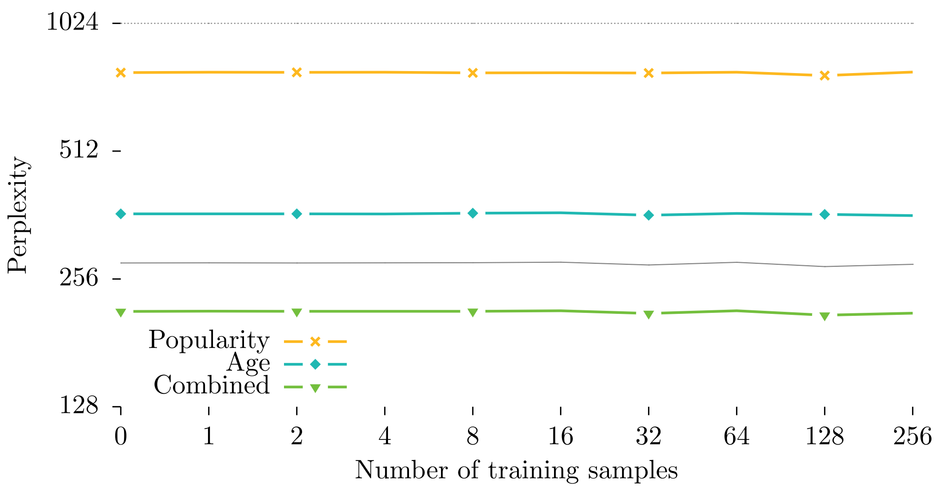


Figure 6.1: Perplexities of the two non-personalized models as well as their combination. The dotted line at the top is the perplexity of a uniform distribution, which gives each item equal probability, and the solid unadorned line in the middle shows the perplexity of a hypothetical model that has a total information gain that is the sum of the two models. Both axes are in logarithmic scale.

was repeated for every user and at every step more training samples were used. The order of the selections in the test set were randomized because there was a strong temporal dependence among them. It was generally very likely that if an article from source s was selected, then the next article will also be selected from source s. In the data set 99% of the selections were made when the article was less than 68 hours old which was the average age of the oldest article among the 1024 recent items.

In figure 6.1 the perplexities of the non-personalized models *age* and *popularity* are presented. These correspond to equations 5.30 and 5.31 respectively. Because the models do not adapt to user selections the curves remain flat after every training sample. The small fluctuations gives a small indication of the overall variance in the average set of recent items $\overline{\mathcal{V}}_{x_i}$. The age model is the probability that a user selects an article from source s that is t hours old. It had an average perplexity of 364 which is the best score out of any model used on its own. The popularity model is the general selection probability of an article, and it had an average perplexity of 783. The combined non-personalized model is the joint probability of equation 5.32, and the results show that the two models synergize well. It reaches a lower perplexity than the information gains of the two models added together which is shown with an unadorned solid line just above 256.

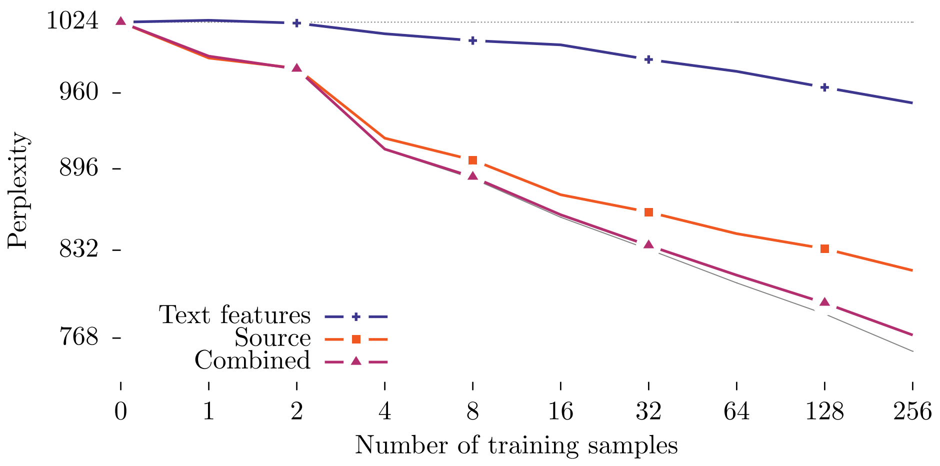


Figure 6.2: Perplexities of the two personalized models as well as their combination. The dotted line at the top is the perplexity of a uniform distribution, which gives each item equal probability, and the solid unadorned line at the bottom shows the perplexity of a model that has a total information gain that is the sum of the two models.

About 17% of the items in the data set had no features defined for them, and excluding those items the rest of them had 2.1 features on average. After feature selection, as described in section 5.4.1, the number of items with no features rose to 55% and the rest of the items had 1.4 features on average. Out of hundreds of features that were available only 80 features were found to be independent from the source and one other and were used in the algorithm.

The perplexities of the personalized *source* and *text feature* models are presented in figure 6.2, and they are defined in equations 5.33 and 5.34. Initially neither of the two models provide any reduction in perplexity over a uniform distribution because they are defined in terms of how the user's interests differ from the average. But the perplexity of the source model starts to decrease after only a few training samples – it starts to gives higher probability to articles from a source the user often reads, and lower to other ones. The text feature model actually has a marginally higher perplexity than the uniform distribution after one training sample, which means that there is some negative information gain, but soon thereafter it shows a slow but steady adaption rate. When combined the model $\hat{p}(u \mid s_a, \boldsymbol{f}_a)$ of equation 5.35 reached a final average perplexity of 770 after every training sample, which roughly corresponds to reducing the set of candidate items by a quarter, and this is comparable to the effect that the popularity model have.

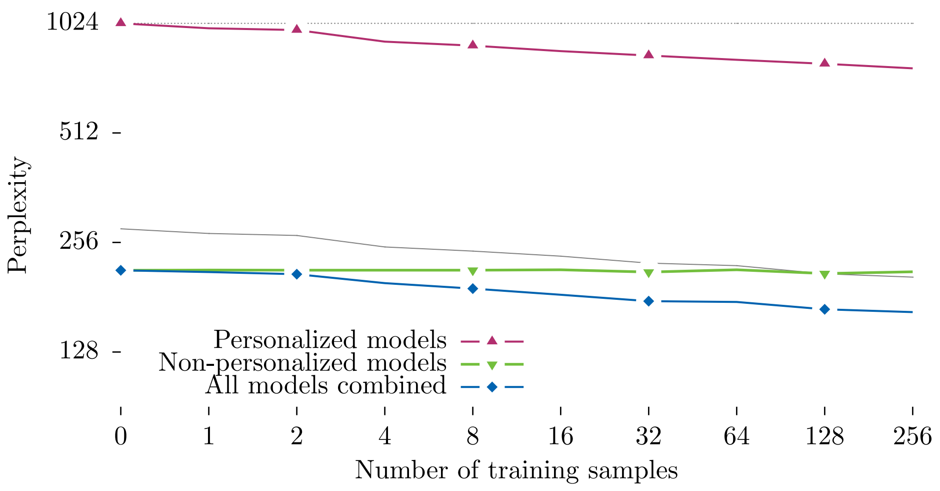


Figure 6.3: Perplexities of the combinations of the two personalized and non-personalized models as well as the combination of all four models. The dotted line at the top is the perplexity of a uniform distribution that gives each item equal probability, and the solid unadorned line in the middle shows the perplexity of a hypothetical model that has a total information gain that is the sum of every model. Both axes are in logarithmic scale.

However, the combination had a smaller information gain than the sum of two models individually, which indicates that there were some redundant information and bias in the two models. For instance, it is possible that certain text features were more frequent in articles from some sources than others and that the source model already could explain why an article was selected, or maybe articles from certain sources never had any features defined for them.

The perplexity of the complete model defined in section 5.5 is shown in figure 6.3. It also contains the results from the personalized and non-personalized models for comparison and scale. An effect similar to the one in the combination of the non-personalized models can be seen where the combined model has a lower perplexity than the total perplexity reduction of every individual model. After having been trained with every training sample the average perplexity of the model is 164.

6.2 Item Probabilities

Every time the probability $\hat{\mathrm{p}}(a_{x_i} \mid u, t_{x_i})$ of a test selection x_i was calculated, the probabilities of every other item in the set of visible articles $\mathcal{V}_{x_i}$ also had to be calculated in order to normalize the result. As an additional step, these probabilities were made into a list that was sorted in ascending order. The items at the end of the list are the ones that got the highest probability value and would be visible in the user interface. When averaged over every test sample and test user, it is possible to see what the average probability of the nth item in the list of recommendations was.

A defining feature of a model is how the mass of probabilities is distributed among the items. A uniform distribution gives equal probability $y_j = \frac{1}{1024}$ to each item x_j while any other model have increasing mass when going from the least to the most probable item. The sum of every probability value is one, and the location with the smallest ordinal m for which roughly half the mass $\sum_{i=1}^{m} y_i \geq \frac{1}{2}$ is on the left side and half is on the right side is similar to the median of a probability mass function. Because the list is sorted $512 \leq m \leq 1024$ and the greater m is the fewer the number of items with a high probability and the more discriminating the model is.

In addition, at every test step the position j of the selected item a_{x_i} in the recommendation list was recorded, which is related to how far the user would have had to browse before finding the item that was selected. The variable z_j contained the number of time the selected item was placed in the jth position in the list of recommendations. Because the site and feature models often gave the same probability value to several items, the position was randomized among every item with the same probability, i.e.

$$z_j = \sum_{i=0}^{N} n_{il} + \lceil u_i n_{ie} \rceil$$

where n_{il} is the number of items that got a smaller probability value than x_i, n_{ie} is the number of items with equal probability, $u_i \in [0, 1)$ is sampled from a uniform distribution, and N is the number of training samples. The result is that the selected item is assumed to appear in a random location among the other items that have the same probability. Histograms were made with the values of z_j to approximate the true probability that a user selects the jth item in each model. The histogram bins were formed based on the density of test selections such that each of them contained roughly K items. Formally the algorithm can be expressed as

$$\begin{aligned}
a_0 &= 0 \\
c_k &= \sum_{j=a_k}^{a_{k+1}} z_j \\
w_k &= a_{k+1} - a_k \\
h_k &= \frac{c_k}{N w_k}
\end{aligned}$$

where $a_{k+1} < 1024$ was the smallest number such that the bin count $c_k \geq K$, N is the number of test cases, and w_k and h_k is the width and height, i.e. probability, of bin k.

The average item probabilities of the models are shown as curves in figure 6.4, each with its corresponding sample distribution histogram in the background. The curve in figure 6.4a is the age model's average item probability and it is the average of equation 5.6 for different t and ϕ_s. The mass is focused in one end and about 100 items account for half of it. The histogram shows that a large fraction of the test items are placed near the top of the list which indicate that the selected item often was the most recent item in the range. Test selections occur even more frequently near the head of the list than what the model predicts which indicates that the value of some ϕ_s parameters could be greater which would result in a steeper peak. This would, however, put even greater emphasis on the most recent items and it is possibly that an exponential distribution is too simple a model.

The popularity model's average item probability curve in figure 6.4b shows a very uneven mass distribution with a small and slow ascent ending in a tall peak by the end. A small set of items are very strongly emphasized while the majority of the items get probabilities that smoothly decrease. The user selections in the histogram do not mirror this behavior and the histogram looks almost linear in comparison.

The article view counts in the data set consist of selections made by a much larger group of users with possibly differing interests and habits. It is possible that the normalizing constant, used by the model to scale the view counts up or down to more closely reflect the news aggregator's users, is too simplistic a transformation and it might introduce bias to the model. Ideally, the selection frequencies should be taken from the same users for whom the selections are used to make predictions.

When the popularity model is combined with the age model, as can be seen in figure 6.5a, its bias as well as that of the time model seem to partly cancel each other out. This effect can partly be understood by recognizing

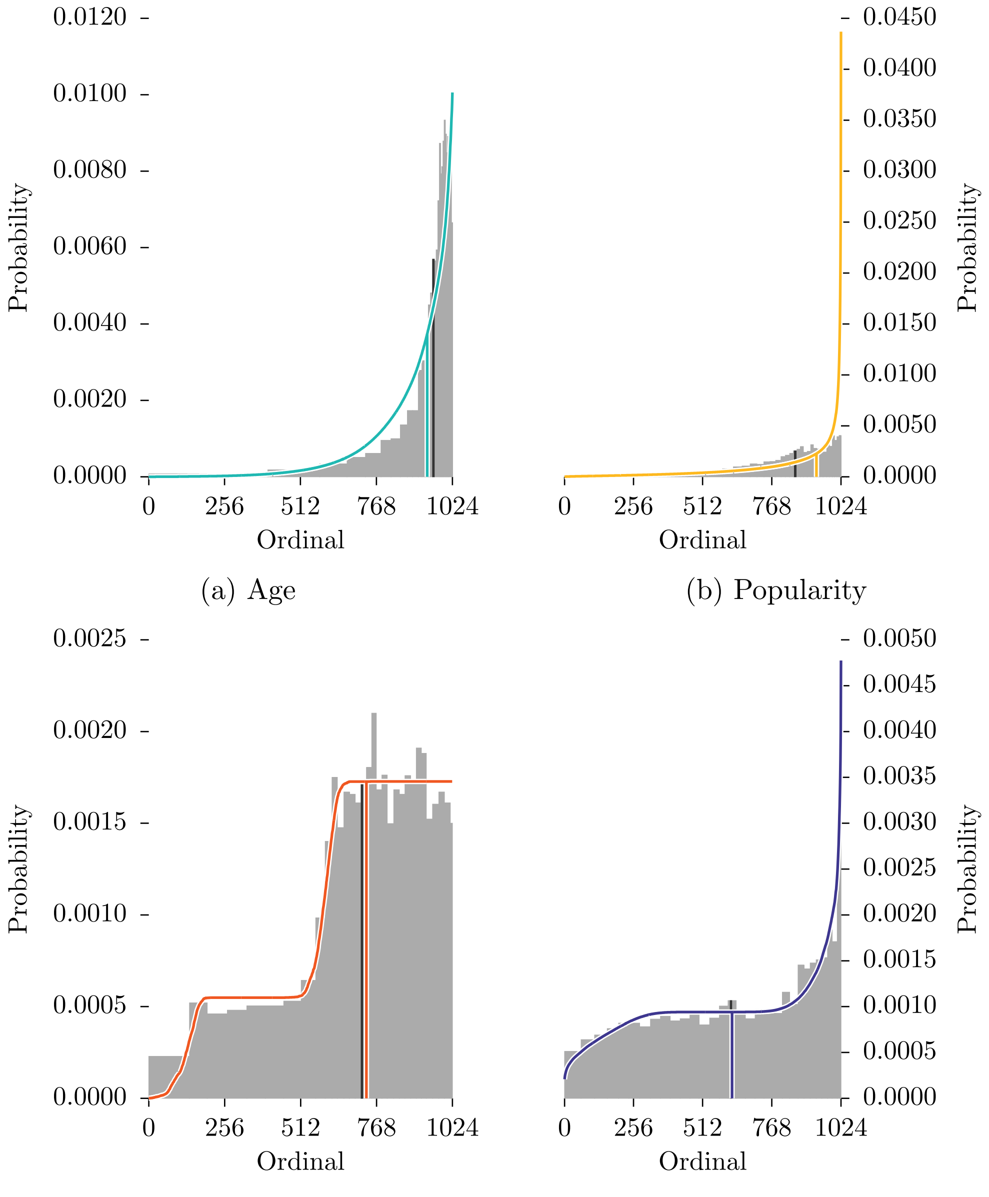


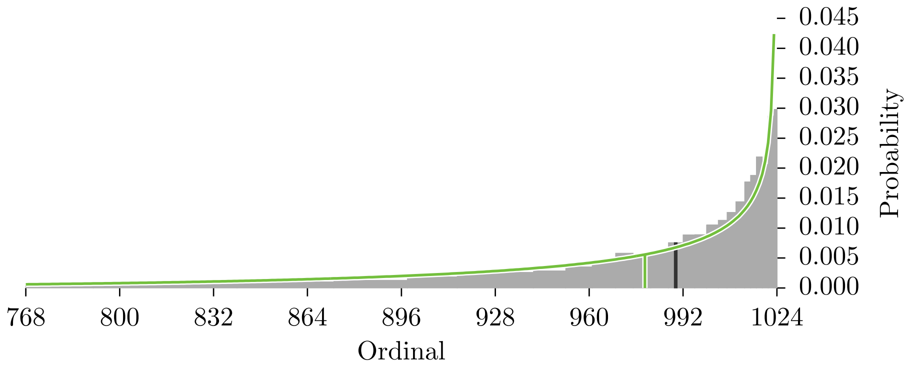
Figure 6.4: The average probability given to to items at different locations in the sorted list of recommendations. The personalized models were trained with every training sample. The histogram behind each curve shows the empirical frequency of selections at each location in the list. The first location where the cumulative sum of item probabilities or the histogram is greater than 0.5 is marked with a vertical line starting from the curve or overlapping the histogram.

that the two models share a dependency on time. Users tend to select new articles, and in the popularity model in equation 5.13 new articles start out from the site mean popularity when w is close to zero. Old popular articles have a more accurate estimate of $\mathrm{p}(a)$ and will be pushed to the front or the back of the list. Further, it is possible that an item that is popular has already been selected by the user and among the users in the data set it was rare that an item was selected more than once. When combined with the age model, items that the user has already seen are often older, and are less emphasized. Similarly, old articles that are rising in popularity and receive low probability scores by the age model get a boost from the popularity model. The combined model is still exaggerating the probability of a small set of items, but the shape of the ordinal probability curve and the histogram seem to agree better.

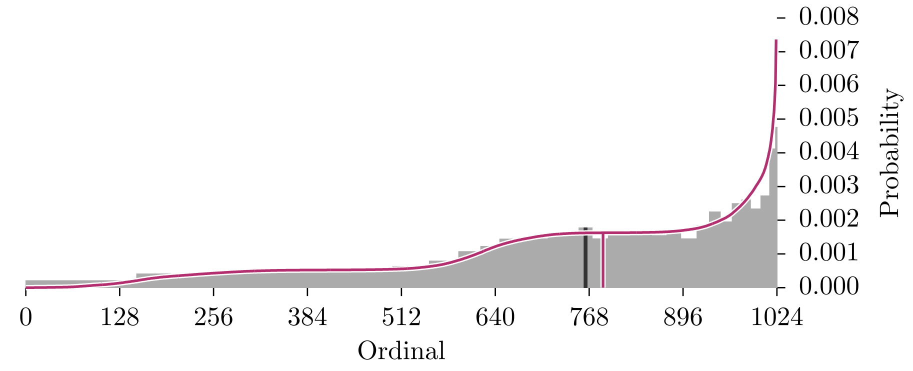
Compared to the non-personalized models, the source and text feature models have much less dense probability masses and their scales are smaller by an order of magnitude. The source model in figure 6.4c has an apparent discrete shape with two primary plateaus. When used on its own, the source model assigns the same probability to every item from the same source, and will thus rank every item from one source before every item from another one depending on the user's preference for it. The figure suggests that at any test, there were roughly an equal amount of items from two major sources, and users generally preferred one of them three times as much as the other. The smooth transition between them results from averaging the probabilities of several users and different numbers of items from each source. The histogram shows that some items are assigned almost zero probability even though selected items can be found in that end of the list.

A large portion of articles in the data set had no derived text features, and even fewer had any after feature selection. There is a wide flat section in the center of the item probabilities of the text feature model as can be seen in figure 6.4d. When items have no discerning features the model gives them the same probability for a user, although it can be lower or higher for different users. Articles that do contain features even after the feature selection procedure are recognized and given appropriate probabilities. There is a small amount of bias in the model because the predicted probabilities of the first items in the list are much higher than in the sample distribution.

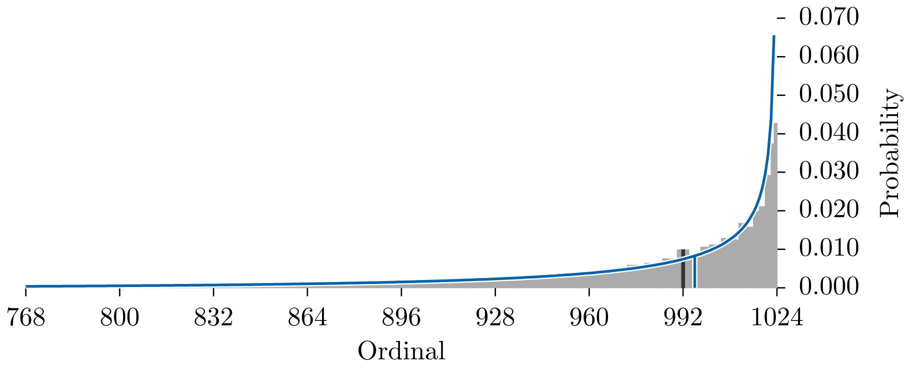
The result of the combination of the personalized models can be seen in figure 6.5b. Given the ordering of items by their source, the text feature model makes it possible to further distinguish items based on their contents. The shape of the text feature model is replicated for both plateaus and it is possible that some items are promoted from a lower stage to a higher one. It is still apparent in the figure that there are many items that lack binary



(a) Age and popularity combined



(b) Source and text features combined



(c) All four models combined

Figure 6.5: The average probability given to to items at different locations in the sorted list of recommendations. The personalized models were trained with every training sample. The histogram behind each curve shows the empirical frequency of selections at each location in the list. The first location where the cumulative sum of item probabilities or the histogram is greater than 0.5 is marked with a vertical line that starting from the curve or overlapping the histogram. In (a) and (c) only last 256 locations are shown which include more than 90% of the histogram mass.

features. Both the ordinal curve and the histogram share a peak at the end of the list, but the approximated probability is less extreme and the bias from the text feature model carries onto this result as well.

Figure 6.5c shows the item probabilities of the total combined model. It has the densest mass distribution with 29 items carrying half the probability mass on average. The observed synergy suggests that the non-personalized models begin by narrowing down the scope of items which makes the personalized models more focused. The biases from both the personalized and non-personalized models accumulate and overemphasize the first few items compared to what the histogram suggests but overall most of the selections in the test set were placed high in the list as the model predicted.

6.3 Coverage and Diversity

Perplexity and cross entropy was used to evaluate and compare the predictive accuracy of models and model combinations, and to guide the development effort of the recommender system. It shows to what degree a model can predict the items a user will select, which is the primary goal of any recommender system. Diversity and coverage was measured as described in section 4.2 to gain further insight into the large scale behavior of the system.

The data set contained article selections of 1000 users that used several different online news sites over a period of six months. A large scale test was performed that simulated the recommender system by generating recommendations for every user once per day using each model. Once every recommendation had been generated the user profiles were augmented to include the test selections that had occurred during that day and the model used them as training samples. This was repeated for every day and performed in chronological order and resulted in roughly 200 coverage and diversity measures.

The set of items to recommend was limited to the 1024 most recent items on each day when the recommendations were generated. Similar to the perplexity tests, this amounted to about two or three days' worth of articles on average and was meant to reflect the items that were visible to users at the time. The top $L \in \{10, 25, 50\}$ items with the highest probability in each users' list were used to calculate coverage and diversity values. This represents the first few pages of results a user might see in the user interface. In case a model assigned the same probability to two different items, the more recent item was placed closer to the first spot in the list. The number of users, items and choice of L can affect these results and also help to interpret them. For instance, since the number of items and users are almost the same, if every user got a unique item in their first L recommended items then the coverage would be the maximum of 1 and if every user got the same item as their first recommendation then the top-1 coverage would be $^1/_{1024}$.

The mean and standard deviation of the coverage can be seen in figure 6.6. The non-personalized models give the same recommendations to every user and the size of the union of every users' lists is $^L/_{1024}$ every simulated day. While the source model has a lower perplexity than the text feature model, the figure suggests that it tends to recommend many of the same items to several users. At $L = 50$ the coverage of the feature model is close to 0.5 which means that about every other user gets a unique item in their first 50 recommendations while for the source model this number is only 0.15. However, by combining the personalized models the coverage is increased. The mean coverage of the combination of every model shows that while the age and popularity models narrow down the scope of items to recommend the personalized models manage to find different items for different users and increases the coverage by more than three times.

Figure 6.7 shows the inter-user diversity which is the mean similarity among the top L items in the users' recommendations. The result can vary a lot depending on the distance or similarity measure that is used. During the test, items were compared solely based on their content using the Jaccard distance in equation 4.4 where A and B were sets containing the source and

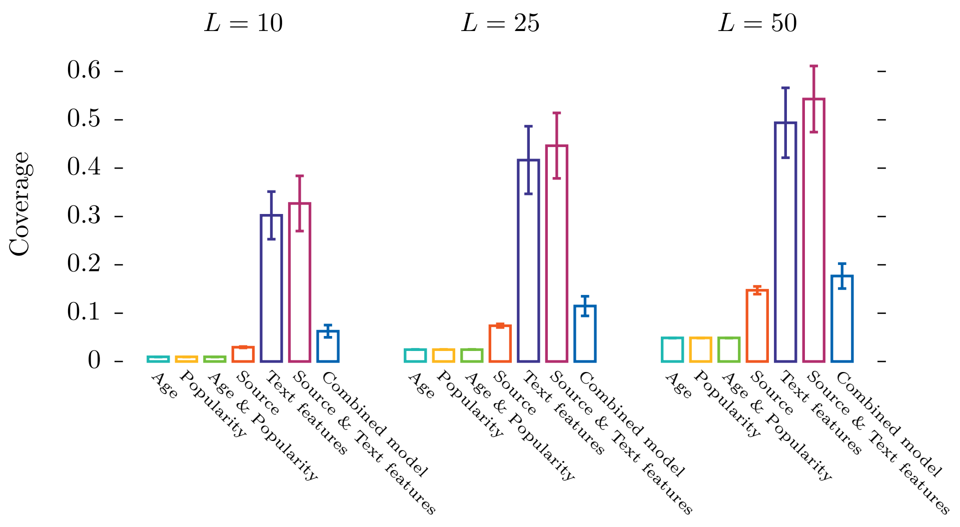


Figure 6.6: Average content coverage of the 1024 most recent items in the first L items for 1000 users for $L \in \{10, 25, 50\}$. The whiskers show the sample standard deviation in the 200 samples.

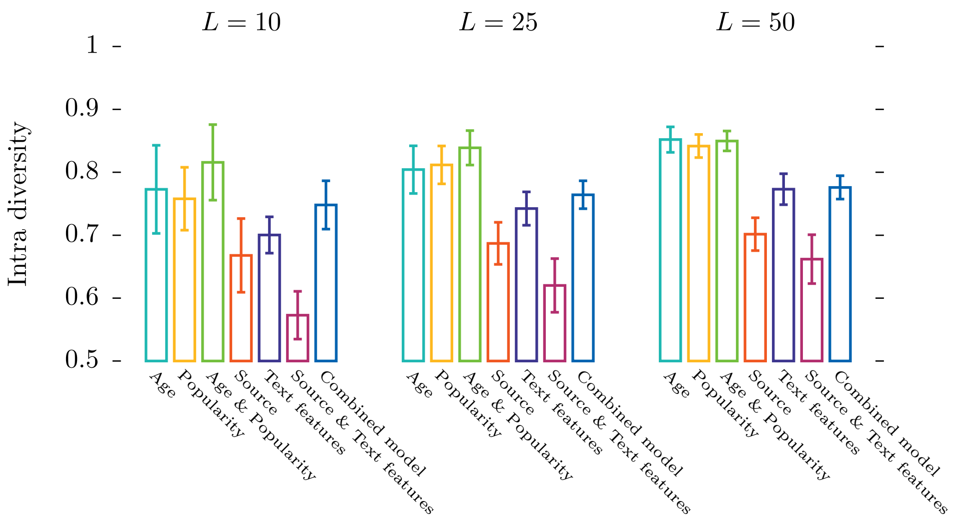


Figure 6.7: Average intra-user diversity in the first L items for 1000 users and $L \in \{10, 25, 50\}$. The whiskers show the sample standard deviation in the 200 samples.

text features of two articles. Overall there is very little difference in the mean as L increases but the standard deviation becomes smaller. The age and popularity models do not discriminate based on content directly. A wide range of items can be popular and the time an item is published determines its age probability score. The non-personalized models provide the most diverse items to users and the personalized ones makes the recommendations more narrow. The source model places every article coming from the user's favorite source ahead of every other, then the items from the second favorite source, and so on. While the text feature model personalizes the results, it retains a higher degree of diversity among the items in a list. For instance, items from different sources can have similar content and cover the same stories and this would increase the diversity because of the distance measure used. When the combined personalized model is used on its own, the results are about 40% similar on average which might not be desirable and there is a risk that the results have become too specific. The final model, however, does not suffer from this problem.

The insight from the intra-user diversity is a little limited because items are compared on a very coarse grained level. The inter-user diversity gives another point of view that is independent of the structure of items. The inter-user diversity compares the fraction of items that are shared between two users' top-L recommendations on average, as defined in equation 4.6. In

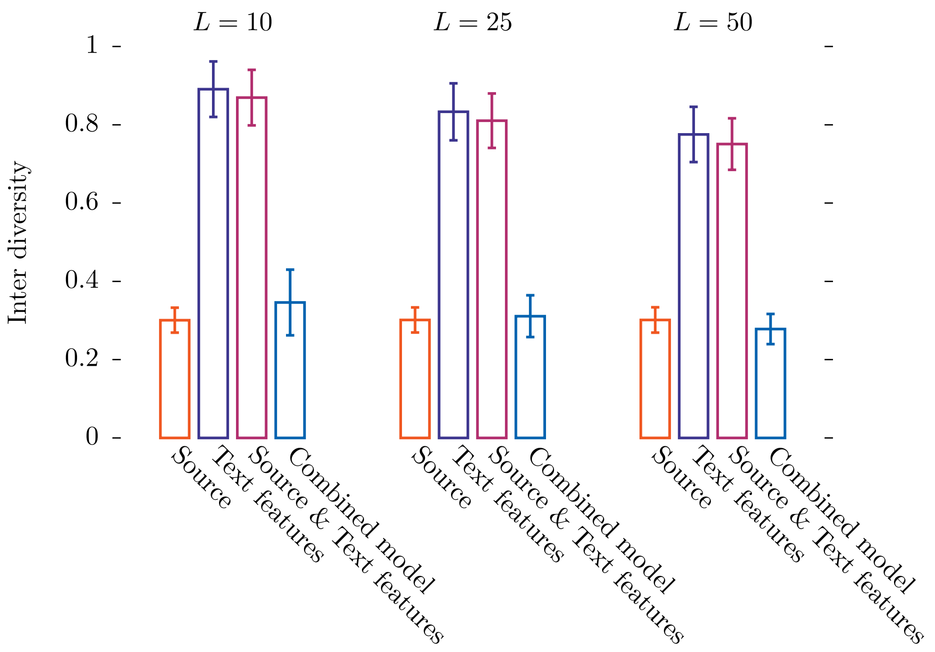


Figure 6.8: Average inter-user diversity in the first L items for 1000 users and $L \in \{10, 25, 50\}$. The whiskers show the sample standard deviation in the 200 samples.

this definition the item order is not taken into consideration and it is possible that several top-L lists are different even though they contain the same items.

The mean inter-user diversity is shown in figure 6.8 along with its standard deviation. Only the source and text feature models are presented since the non-personalized models provide no diversity. A general trend is that diversity decreases the more items that are considered. The text feature model provides the most unique results and only one of a user's top-10 items can be found in other users' top-10 recommendations on average. The source model has little diversity and it is almost constant at 0.3 in the ranges of L that were tested. This number correlates well with the distribution of selections in the data set where 30% of the selections were of items from one source and more than 60% were from another. It is possible that the users who preferred one source over the other were similarly distributed which would make the source model give only items from source A to 30% of the users and items of source B to the rest. If there were at least L items from each source in any given test then on average two users that preferred B would have no diversity among each but a user that preferred source B and another one that preferred source A would have no items in common. The variance could indicate users that were on the edge reading articles from both sources equally often.

Figure 6.9 shows the 200 top-10 inter-user diversity statistics plotted against time and provides additional insight into the rate of learning and personalization. The period of learning in the beginning of the curves amounts to some of the variance in the results. The source model shoots up early on and approaches a diversity value close to 0.5, but after roughly two weeks it stabilizes and has very little change in diversity thereafter. The text feature model and the combined personalized model also reach a level between 0.8 and 0.9 after only a few days. This means that people shared one or two of their top-10 items with other users on average. They have higher variance than the source model because features allows items to be different in more than one way and makes more personalization possible. The non-personalized models and the source model reduce the number of items that can be personalized among the top-10 and the diversity of the combined model varies between 0.2 and 0.5, but seems to have an upward trend.

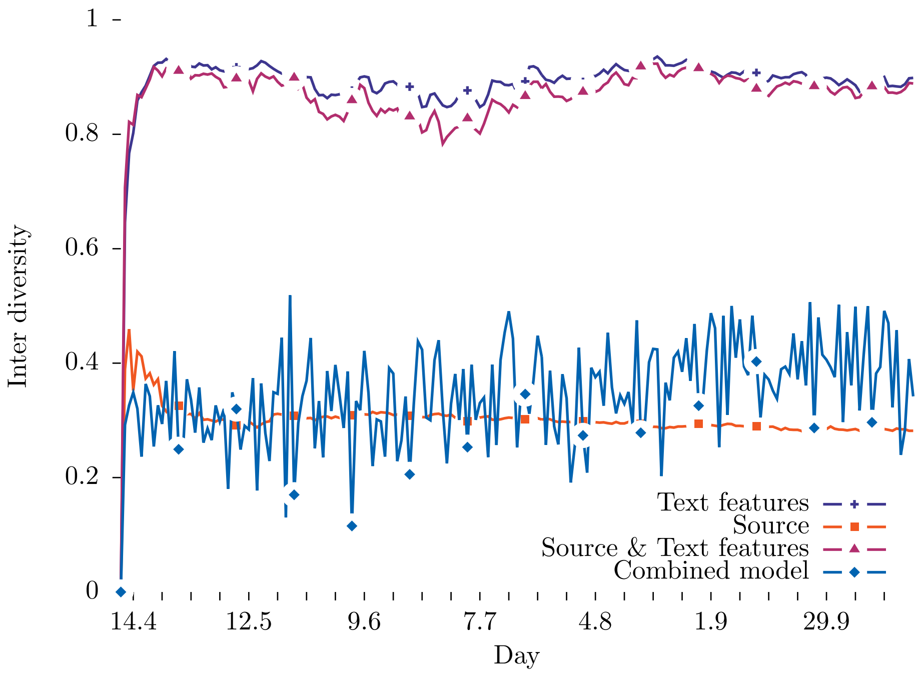


Figure 6.9: Sample inter-user diversity in the top 10 items in chronological order.

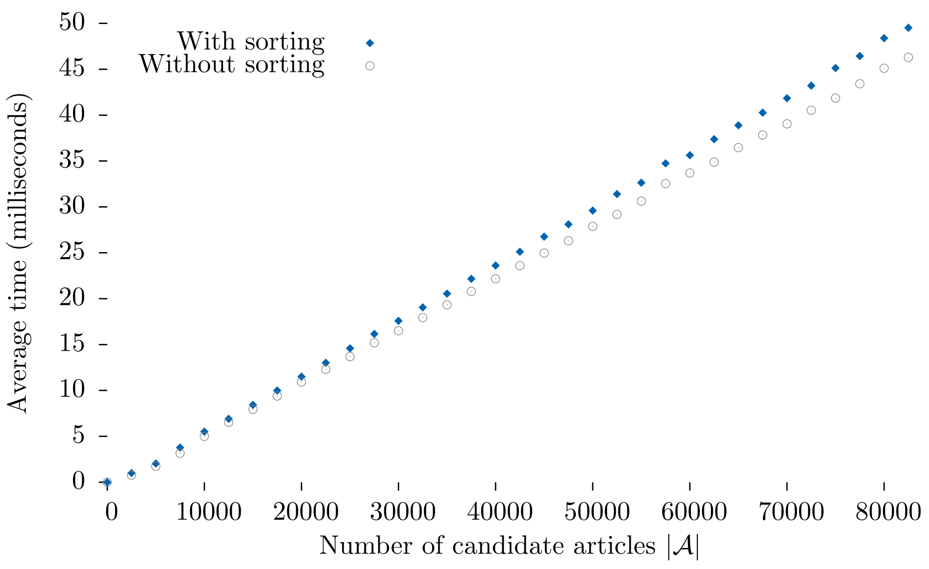


Figure 6.10: Average time to recommend items for a single user when the list of recommendations is sorted and when it is not from 50 measurements.

6.4 Time Complexity

The analysis in section 5.5.2 shows that that the algorithm calculates the item probabilities in linear time and that the list can be sorted in linearithmic time. In order to discover the constant factors and the scalability of the algorithm in practical use, equation 5.29 was implemented and used to generate recommendations with varying number of items n for a single user from the data set. The running time was measured and repeated 50 times and ran on a commodity computer. The article and user parameters all fit in roughly a hundred megabytes of memory even though the data structures had not been optimized for space and contained redundant and unused data, such as features that had been filtered out. Figure 6.10 shows the average time to generate recommendations when the list of recommendations is sorted and when it is not. The difference between the results provide an indication of the relative size of the constant factors in both operations. The constant factor in the probability calculations clearly dominate the sorting operations when the number of articles lie in this range – when the number of candidate articles $|\mathcal{A}| = 80000$, sorting the list takes less than 7% of the total time.

Chapter 7

Discussion

With a solution based on probability theory, the problem of predicting user behavior could be distilled into four small models. Each model could be tested and used in isolation, and together they formed into a combination where each part contributes in its own way. The test bench, which gave constant and immediate feedback when exploring model implementations, allowed the development process to be iterative and incremental. The results of the model perplexities shown in figures 6.1, 6.2, and 6.3 and the ordinal probabilities shown in figures 6.4 and 6.5 indicate that there is a strong connection between these two properties. The lower the cross entropy got, the closer the model was to the true distribution and the better it was able to predict what users are going to select. The diversity and coverage metrics provide additional understanding and insight into the overall behavior of the algorithm and give reasons to believe that the results are personalized and useful.

The non-personalized models, and the age model in particular, seem to reflect much of the underlying system. Online newspapers display the most recent news on the front page. Old news items are gradually pushed farther and farther down the list, which makes them harder and harder to find. In this sense the age model is almost personalized, because it hides items that the user has likely seen before. The more accessible an item is the more likely it going to be selected and this favors articles that are new. While the number of times an article has been selected says something about how interesting or important it is, it is also true that some articles are placed in more prominent locations and stay there for a long time, and it is natural that a head line story, which is visible to all users, is going to be selected more often than others. Overall, the age model provided the lowest perplexity of any individual model and its great effectiveness is probably related to this domain. When content does not lose its relevance as quickly as news do, time might not be as easy to factor in and the effect of removing items that has been seen before might have to be modeled in another way.

During the tests, the item a user selects was found half of the time among the first 35 items in the combined non-personalized model, as can be seen in the histogram in figure 6.5a. When the personalized models are included the perplexity decreases, which is shown in figure 6.3, and the center of mass of the empirical item probability distribution pushed forward by 3 places, which can be seen in figure 6.5c. The change in distribution of mass is small on a large scale, but among the first few items it becomes much more concentrated and the probability that a user selects the first recommended item grows by over a third. The improvement can be understood in light of the inter-user diversity shown in figure 6.8. Around 30% of the top-25 and top-50 items that are recommended to a user are relatively unique. This means that some of the items that would have been recommended by the age and popularity models alone are pushed further down and some older and less popular items are brought to the front of the list. When compared to the empirical item probability histogram of the age model in figure 6.4a it becomes apparent that the combined algorithm is over four times better at recommending items for users than simply ordering items by their age.

Users evidently prefer some sources over others. When used on its own, the source model counters some of the effect of news aggregation by hiding items from sources the user does not like. In combination with the text feature model, items with interesting content can appear before items from the user's favorite source. The inter-user diversity shown in figure 6.8 suggests that there is enough diversity to make personalized recommendations even if items from one source have a higher precedence than others. There is also a correlation between the occurrence of words in the text and a user selecting it. Synonyms and homonyms can cause major problem in key word matching, but in a probabilistic model the occurrence of a single word does not immediately make an item seem interesting. Instead, the words only increases or decreases the probability of the item being selected and this probability is but one among many factors. The intra-user diversity in figure 6.7 and coverage in figure 6.6 indicate that the key word feature granularity was on a meaningful level. Had there been a small number of features that occurred in almost every item then both the intra-user diversity and coverage would have been much lower.

Unlike many state of the art recommender systems, the algorithm doesn't look directly for correlations among individual users or items. This means that it does not recognize or make use of certain kinds of patterns in how the content is used. However, the algorithm is small, efficient and easy to implement. There are few dependencies among the variables, which makes the algorithm flexible. User and item parameters can be updated incrementally and in isolation, and if necessary, there are ways to improve the scalability of an implementation. Finally, its modular probabilistic nature provides was to extend it in new ways.

If there were any reason to doubt the results, it would be the applicability of data set. The users were not using news aggregators, and the selection behavior is not directly comparable. The behavior depends in part on the environment which dictates what the options are (i.e. an online newspaper), and if users had been exposed to these recommendations the items they would have selected would likely have been different. A follow-up report on how well the system works in production would be illuminating, and any future model improvement should be based on data from the aggregator's users. The number of users in the data set was small and, although the number of selections and items was substantial, every article had been selected by at least one user. This reduced the volume of the content, which affects the accuracy of most models. In the model tests, the latest 1024 items were enough to represent several days' worth of articles, but if thousands of articles were published every day then the probability mass distribution of the age model would be much more even and approach the uniform distribution, and features would have to be more detailed to find unique properties in items. On the other hand, the difficulty of any information retrieval task is proportional to the size of the content.

7.1 Future Research

There are several opportunities to expand on the algorithm. In many cases independence assumptions were made and enforced. New variables and models, and existing variable dependencies could be further examined to generated even more accurate and diverse results.

Time was assumed to be conditionally independent of the user in the algorithm. One hypothesis is that some users read news every day while others read several days' worth of news less frequently. This could be tested by turning the age model parameter ϕ_s into a personalized one. Mathematically this can be expressed through the symmetry $\mathrm{p}(u \mid t, a)\mathrm{p}(t \mid a) = \mathrm{p}(t \mid u, a)\mathrm{p}(u \mid a)$ to make the age model depend on the user. In addition to article age, time could be used for other things. For instance, people read different things during week days and week ends [29] or during working hours and evenings, and other time based models model could be created. Dependencies among these aspects could also be explored. For instance, on a Monday it could be useful to have a different value for ϕ_s because many of the most recent articles might be weekend specials, and this change would affect the estimated popularity.

Other contextual parameters could also be used in models, such as location. User interests vary from place to place [28] and location could be used to make the out of box experience more personal by recommending local news. Many mobile devices have GPS and similar location aware capabilities which could be used as additional input, given the user's consent. People living abroad might also be interested in current affairs of their home region and interest in specific locations could be derived from the articles that users have read.

Once an article has been read, a user might want to learn more about the same subject. Content-based similarity is a common information retrieval problem and algorithms such as k-nearest neighbors, PLSI and LDA could be used for this purpose. The similarity would necessarily be made on a more detailed level than the text features, or at the very least before any features have been filtered out. The most similar articles could then be ranked a second time based on user preference. Article similarity could also be used to improve the popularity model. Currently, to handle the cold-start problem, it makes an initial approximation of the total selection count based on the average popularity of articles from the same source. A better approximation could be made based on articles that have similar content and are from the same source.

Most publications organize their content into sections such as domestic and foreign news. This categorization could be combined with the source model to make certain articles from one source more probable than others from the same source. Then the sports section of one source could get a higher probability than the finance section of another source, even though the user prefers the latter source overall. When there are similar sections in several sources the knowledge of a user's interests in one could be used to predict the interest in the other even though the user has never selected anything from that source yet.

An unfortunate side effect of enforcing the assumption in equation 5.4 that sources and text features are independent, is that features that did depend on a source had to be removed. This means that words that described general things like cars or technology were removed because they were tied to particular sources. One obvious problem is that users are likely going to be interested in articles about cars whether they are published in car magazines or not. By properly considering the dependency between sources and features it would be possible to identify that car words can be significant unless it is published in a car magazine.

Finally, large gains in accuracy might be achieved by incorporating collaborative filtering algorithms into the model. There are many options to choose from and ones that can be tested in comparison and combination with the current models are especially desirable. If the algorithm is based on probability theory, it might be easier to see how its parameters and variables relate to the existing ones.

Chapter 8

Conclusions

A recommender system built out of probabilistic models was developed for a news aggregator service. The models were mainly content-based and each of them reflected different aspects of the selection behavior of users. Personalized models adapted themselves to the specific behavior of individual users and non-personalized models captured the general trends among all users. The cold start problem, which can cause models to act erratically when little or nothing is known about an item or a user, was a primary concern. It can lead to news arriving too late or new users getting a bad out of box experience with recommendations that do not make sense. With the help of a custom software tool, different models were iteratively tested, evaluated and optimized based on their their accuracy when the level of knowledge about a user gradually increased. The primary optimization target was cross entropy, which can be interpreted in the form of perplexity. The development method yielded models with clear inter-dependencies and fairly low bias, and their algorithms were easy to understand, measure and implement with simple space and time complexities.

One model focused on how time affects the probability of an article being selected and it was found that, as one might expect, the age of the article has a great impact. The general popularity of an article was the target of another model and articles that many users select is likely to be selected by other users as well. These two models are non-personalized and could be used as the basis for recommending items to new users. They reduced the scope of the content to consider so that personalization could be more specific in other cases. Certain properties of an article's content were found to be correlated with selections. First, people tend to prefer articles from some publications more than others, for instance, a publication might have a characteristic style, tone or topic that is appealing, and this model formed the first level of personalization. The range of topics covered by articles from a single publication can be very large, and key words were used as a finer grained level of personalization. When combined, the four models each contributed in their own way and could predicted with high accuracy what users were going to select.

When content is abundant and diverse, a recommender system can make the subjectively interesting parts discoverable and accessible. The problem has been tackled in various circumstances and solutions can be made arbitrarily complex. Guided by the principles of simplicity and modularity, it has been shown a lot of gain can be achieved even by very simple means.

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning.* MIT press, 2004.

[2] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[3] David Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[4] Robert M Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[5] Robert M Bell, Yehuda Koren, and Chris Volinsky. All together now: A perspective on the netflix prize. *Chance*, 23(1):24–29, 2010.

[6] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[7] Krishna Bharat. And now, News. The Official Google Blog, January 23 2006. URL `http://googleblog.blogspot.com/2006/01/and-now-news.html`. Accessed February 12 2015.

[8] Daniel Billsus and Michael J Pazzani. *A hybrid user model for news story classification.* Springer, 1999.

[9] Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[11] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the*

Fourteenth conference on Uncertainty in artificial intelligence, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[12] George Casella and Roger L Berger. *Statistical Inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[13] Pablo Castells, Saúl Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. In *International Workshop on Diversity in Document Retrieval (DDR 2011) at the 33rd European Conference on Information Retrieval (ECIR 2011)*, pages 29–36. Citeseer, 2011.

[14] Thomas H Cormen, Charles E Leiserson, et al. *Introduction to Algorithms.* 3 edition, 2009.

[15] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[16] J Dean and S Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, 2004.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[18] Jordan Ellenberg. This psychologist might outsmart the math brains competing for the netflix prize. *Wired Magazine*, pages 114–122, Februrary 2008.

[19] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[20] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[21] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[22] Thomas Hofmann. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems*, 22(1):89–115, January 2004.

[23] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction.* Cambridge University Press, 2010.

[24] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. In *ACM SIGIR Forum*, volume 37, pages 18–28. ACM, 2003.

[25] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[26] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[27] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[28] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[29] Andreas Lommatzsch. Real-time news recommendation using context-aware ensembles. In *Advances in Information Retrieval*, pages 51–62. Springer, 2014.

[30] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

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

[32] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets.* Cambridge University Press, 1st edition, 2011.

Appendix A

Data set

When models were tested in the test bench, a data set was used which contained selections and articles. Selections were represented as triples of the form $\langle$*user-id, article-id, time-stamp*$\rangle$ and there were two million selections that had been done over a period of seven months, 83000 items from 15 different online news sites, and 1000 users. Most of the users frequently read articles from more than one site. Except from the selections they made, there was no distinguishing information about users such as age, life style or location, and they had been reduced to an anonymized identifying number in the range $[0, 999]$. The text of each article was available and there were also binary indicators of the presence or absence of key words. The key words had been hand-picked ahead of time and some articles contained none of the key words. The news sites had different kinds of content and a varying numbers of users. Every article in the data set had selection counts from every reader of the publication.