Federal State Autonomous Educational Institution of Higher Education
National Research University Higher School of Economics

International Laboratory for Applied Network Research

Dmitry Donetskov

Making Recommendation to Online Learner

Master Thesis

Field of study: 01.04.02 «Applied Mathematics and Informatics»

Programme: «Applied Statistics with Social Network Analysis»

Reviewer:
Marjan Cugmas

Scientific Supervisor:
Prof. Dr. Janez Demšar

signature

signature

Moscow – 2019

Contents

Annotation

Recommender systems is an important applied field of Data Science and use many of techniques from Computer Sciences, Data Mining, Machine Learning and Decision Science. They first appeared in the early 1990s motivated by unprecedented data availability because of the Internet era start, and quickly involved into useful instruments for both of their users and owners. They help to mitigate the weight of information overload and *de jure* is the standard of Internet sites for successful businesses and institutions.

The current work focuses on designing a recommender system for a real web site taking into account all aspects important for its successful implementation which is unique because major part of literature covers only specific parts excluding implementation considerations.

The first part consists of a recommender systems field review including specifics regarding their application in the Technology-Enhanced Learning (TEL) domain. The review was conducted with the main goal to understand the current state of the field and its promising directions of evolution, as well as become familiar with problems usually encountered in their practical implementations.

The second part is a project-oriented design of a recommender system for a website with educational materials so as to enrich it with the functionality of generating recommendations for users based on their past behaviour. The exercise is mostly successful and its results are promising though they could have been better for two main reasons: a) there is no much activity (per user) on the site yet, b) available data is prone to randomness and its collection was not designed to enable a modern recommender system. In order to remedy that, a number of enhancement requests were raised with the system owner, further work on improving quality of capturing user-item interaction-centric data to evolve the recommender system model is suggested.

Chapter 1

Introduction

Actuality

The world experiences unprecedented pressure of information due the era of Internet stimulates more people to participate in generating information, thus, more people are generally expected to manage with larger volume of information, or, alternatively producers of information are expected to become more precise at presenting relevant information to those who are genuinely interested in that.

One of methods to deal with it efficiently is the concept of content personalization i.e. to show to a web site's visitor content that meets that visitor's specific needs. The concept is utilized by probably all major Internet sites and, definitely, by such iconic sites like Amazon, Netflix, Google etc. It is considered the best practice to present different goods and services to a visitor (possibly, a customer) in that 'wiser' way. Commonly used technique for achieving it is to design and to include so-called *recommender system* in inner (back-end) parts of web site.

Online education is also one of those applied fields which strive to deliver personalized content to an user. It helps a user to become aware of potentially interested items for them with less efforts. The current work is motivated by the same market-based guiding principle.

Specifically, the work is devoted to designing and building a recommender system for an online education site users thereof are mostly entrepreneurs from small and medium-sized businesses (SMBs). The relational number and maturity of entrepreneurs in Russia is on significantly lower level than in the developed countries, it is estimated at 5-10% vs. up to 40% in Europe and the USA [79; 80]. One of reasons for that is believed to be lack of relevant

education in Russia. The site with entrepreneurial-specific materials is aimed to improve the situation.

Users register on the site via different channels, log on and browse through materials on it. It is required to support their interest and navigation with personalized recommendations as in the content.

Objectives

The objective is to design and implement a recommender system including the technology level. It aims to improves the site's usefulness for its users and must use information of past user behaviour.

Hypothesis

It's possible to derive personalized user recommendations regarding items based on their previous behaviour on the site captured in the form of data.

Objects

The primary objects are users, items and relationships between the former two.

Tasks

It is required to analyze what methods are available, what methods are supported by available data and design a data-driven recommender system on both the technology and algorithms levels.

Methods

There are used memory-based and model-based methods from the field of recommender systems, namely, the user-based collaborative filtering and the matrix factorization.

Data

The data is user navigation history in the form of web events (calls to a web server), and some basic information of users and items.

Report Structure

The work consists of introduction, five chapters, conclusion and two appendices. The complete work consists of 67 pages including 13 figures and 8 tables. The reference list contains more than 75 items. The source code used is published at the link https://github.com/ddonetskov/masna, also coded as a QR code on the right. Erratum and updates to this work, would they deem necessary, are also published at the link.

Chapter 2

Field Review

2.1 Main Objective of Recommender Systems

Recommender systems (RS) is an important applied field of Data Science and use many of techniques from Computer Sciences, Data Mining, Machine Learning and Decision Science. They are very popular as both research and application areas, and can be found nowadays in different applied domains including e-commerce, entertainment, social networks, online education, and others. The practical significance of recommender systems was highlighted in numerous articles [43] and even worldwide contests with large sum prizes [10; 12].

Traditional terminology refers to the entity to which a recommendation is given as the *user*, and an entity being recommended as the *item*.

There are conferences devoted to the field, one of the largest is The *ACM Recommender Systems conference (RecSys)*[1]. It is a well-established one since 2007 and its participants regularly contribute new results in the field.

As the name suggests the main objective of RS is to provide recommendations in a systematic way, and to quote the ACM RecSys website what a recommendation is

> Recommendation is a particular form of information filtering, that exploits past behaviors and user similarities to generate a list of information items that is personally tailored to an end-user's preferences.

[1] https://recsys.acm.org/

In fact, that is largely the Netflix competition which attracted many talented researchers throughout the world and motivated them to suggest many new algorithms to tackle with the question. Assuringly, following that competition there was established the conference and many other businesses were inspired to build their own recommender systems to increase their profits on the basis of better meeting mutual interests between them and their users.

There are well-regarded textbooks completely devoted to the topic [5; 6; 60; 61; 63].

One can think of at least two reasons for the necessity and the growing popularity of recommender systems.

The first one is the evolving Internet era which has allowed a lot of people to express their preferences towards different things on many sites in Internet, therefore, information about their preferences, previously difficult to collect in the pre-Internet era, has become largely available and its volume is exponentially growing ever since. Until recently, it was required to run through expensive procedures to obtain that kind of data, e.g. asking people explicitly by calling them, interviewing them.

The second one is that a common user usually do not have resources these days to browse through all possible materials, offers etc potentially fulfilling their requirements but they still want to deal with items more or less meeting their interests. The phenomena of too much information is also called *information overload*. The ability of business to make this task easier for an end user is a concurrency instrument especially if they are given in a clear and concise manner to let user make an informative decision.

Users usually do not object that their interests are regressed and computed in a controlled manner as they tend to benefit of it. In fact, recommender systems is an excellent example of the 'win-win' case. Designers of recommender systems try to go further and open a dialogue with users by providing explanations why the latter receive certain recommendations and allowing users react to those explanations (which is another source of information to improve the accuracy of recommender systems).

Given the above, researchers and practitioners back in the 1990s started to justify and engineer values out of all that data became available for users. The widely used name for building a framework around relevant data to estimate user preferences and express them in the form of recommendations

is *recommender system*.

I'd like also to illustrate the importance of a good recommender system on the example of academical publications. I hope a reader may excuse this digression for the sake of clarity. Thus, Larsen and Bormann from Scientometrics estimated there is a substantial increase in the total number of academic journals, researchers and, consequently, research papers: the annual increasing rate of the number of published papers is estimated from 3-5% [35; 42] to 8-9% [13]. Researchers seem to no longer be able to comprehensively monitor publications in their areas of interest but have to rely on various search techniques so the quality of thereof are critical for academic advancements by researcher workforce in an economical and practical way. On the other hand, there are doubts that this proliferation of papers represent real growth of knowledge. As far back as 1965, Price noted a now familiar observation [23]:

> I am tempted to conclude that a very large fraction of the alleged 35,000 journals now current must be reckoned as merely a distant background noise, and as very far from central or strategic in any of the knitted strips from which the cloth of science is woven.

Therefore, researchers have to cope with both the information overload and the controversially quality degradation of that information. A specialized recommender system would help them here to find *more* (value) with *less* (efforts).

The same principle applies to other areas where many people have to deal with many items. So, the main purpose of recommender systems in broad terms is to help users to cope with the *information overload* challenge which most of people, from news readers and on-line shoppers to professionals and researchers, experience more and more every new year, by providing recommendations on what items they might like most of in the context of their interests.

2.2 Diverse World of Recommender Systems

Recommender systems rely on an assumption that users behaviours are not changed over time at least not over relatively a short period of time (an interpretation ot it depends on an application) i.e. users are likely to have

preferences in the future similar to those they have in the past. It is this assumption which allows extrapolating user's history into their future and predict items which they might benefit of. This assumption is a strong one and one can question its validity but for the purposes of a recommender system it is considered correct enough in practice because many studies around that assumption showed recommendations provide results better than just random prediction.

There are so many considerations and factors that assumption triggers when trying to put it under a quantitative analysis that the field of recommender systems has become considerable broad over the years of its development. The main source of diversity is probably in dealing with

- various types of user preferences as the Internet space of human activities is getting more dimensions, and user preferences can be expressed either *explicitly* or *implicitly*,
- various types of items: from utilitarian objects to books, movies, music, tourist recommendations, social connections, research papers, services up to anything which can be shown on a web site

Some recommender systems go further and try to impress users with what they did not originally know but they would really like or need it [28], that also adds the variability to the set of methods.

There might be different classifications of recommender systems because of that variability in terms of algorithms, methods of evaluations, domains and context applied but traditionally they are grouped as content-based, collaborative filtering and knowledge-based. Though they all are devoted to meeting the main objective (inferring user interests based on existing information), they differ how they treat users, items, past user interests and future user requirements.

There are three basic types of recommender systems: the *collaborative filtering* ones focus on ratings for user-item interactions, the*content-based* ones mostly rely on attributes of users and items (though they may account for ratings), the *knowledge-based* recommender systems rely on explicitly specified user requirements and constraints. Sometimes, these methods are combined to create a *hybrid* recommender system. Real implementations of a recommender system may target different goals therefore several classifications have been

used in the past to overview recommender systems. One of interesting attempts were made in [58] and is shown in Figure 2.1.

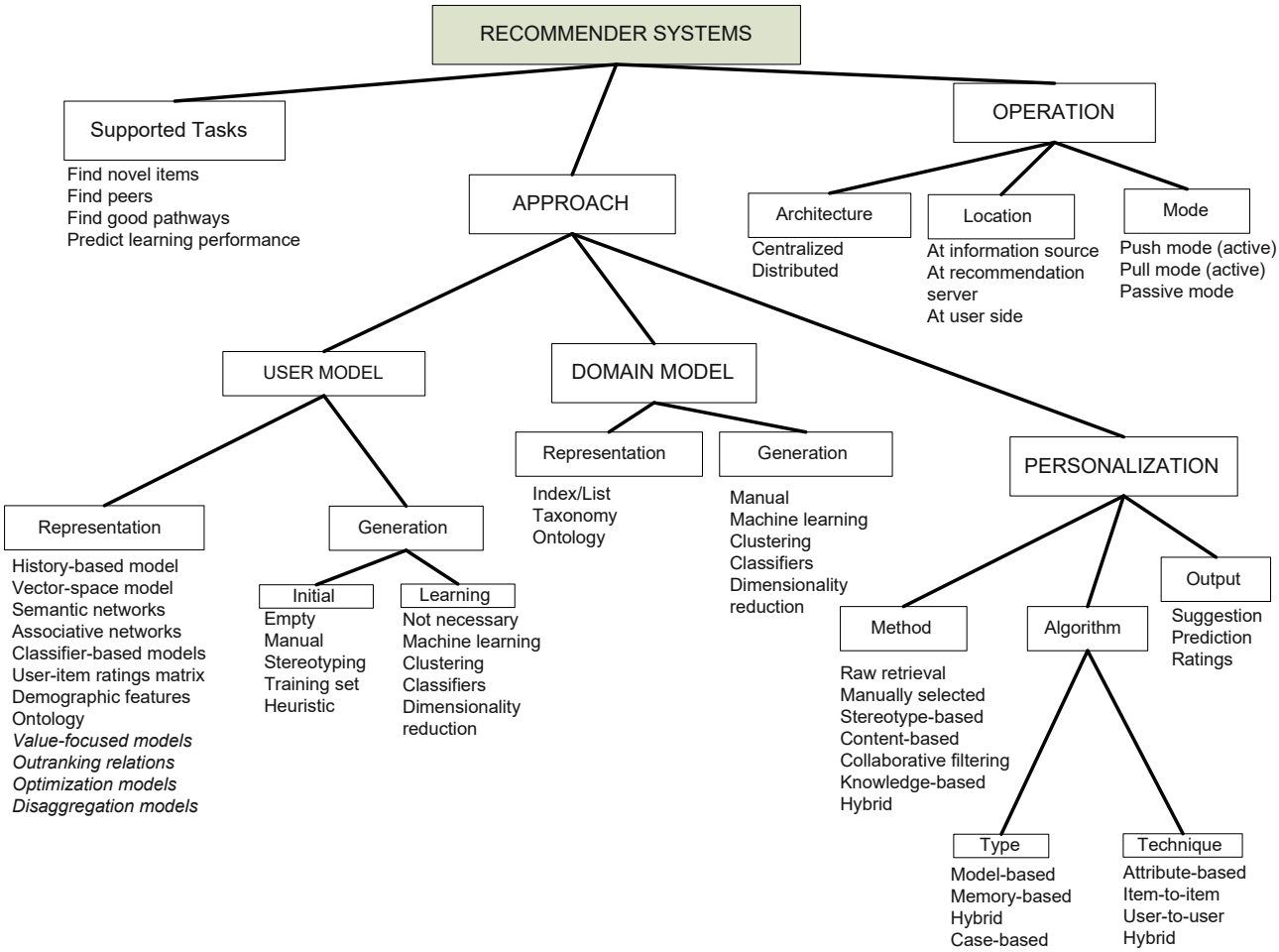

Figure 2.1: Classification framework for recommender systems

There are different sources of data for recommender systems. Loosely, they all can be divided into *explicit* feedback (e.g. explicit ratings of items) and *implicit* feedback (like browsing and buying behaviour). There are different means to let users leave explicit feedback for items. The most common way is to let them rate items according to some *scale*. Naturally, recommender systems try to take into account other useful information about users like demographics [41; 53], temporal data, location data, social data etc according to specifics of context, which is currently effective, to infer or adjust interests. This type of recommender systems is often referred as context-based stressing that they use contextual information. They are probably the best at approximating the real world around user because they can produce the most

accurate recommendations.

2.3 Principles

The very basic principle underlying recommender systems is that for a given set of *users* $U = u$ and set of *items* $I = i$, there exist a quantified *relations* $R = r$ between them, and some of those relations are significant from the goals' point of view $G = g$:

$$\forall i \in I, \forall u \in U : \exists r \in R : r \; is \; of \; interest \; to \; u \; given \; G. \tag{2.1}$$

So, the principle implies R needs to be known, its missing values should be recovered (predicted), it invites using Statistics and Machine Learning methods to solve it as a prediction task. For example, Amazon generates recommendations on its site[2] by interpreting those associations as 'these items are probably interesting to you as they are similarly favoured by users like you' e.g. they infer correlations between users and items under that principle which is a statistical task.

So, the task is generalized as the prediction one, and for that reason it is assumed that the training data is available. In some cases, the task can be simplified to just ranking the ratings R instead of getting their values.

Recommender systems are practical things by their nature, and their development are historically driven by the industry, they are created and developed mostly by practitioners from the fields of Computer Sciences and Machine Learning. Unsurprisingly, they heavily rely on methods from those fields, most of recommender systems seem to be implemented with methods and techniques from those fields. Good overview of Data Mining methods used for the purpose is provided in [22] and shown in Figure 2.2 in association with different stages of data transformation. Again, the set of methods is as such that it supports the view the task of recommender systems is a generalization of the classification and the regression.

There are many articles on using statistical methods in the field but they do not look to be systematically described yet except in [4].

[2] https://www.amazon.com/

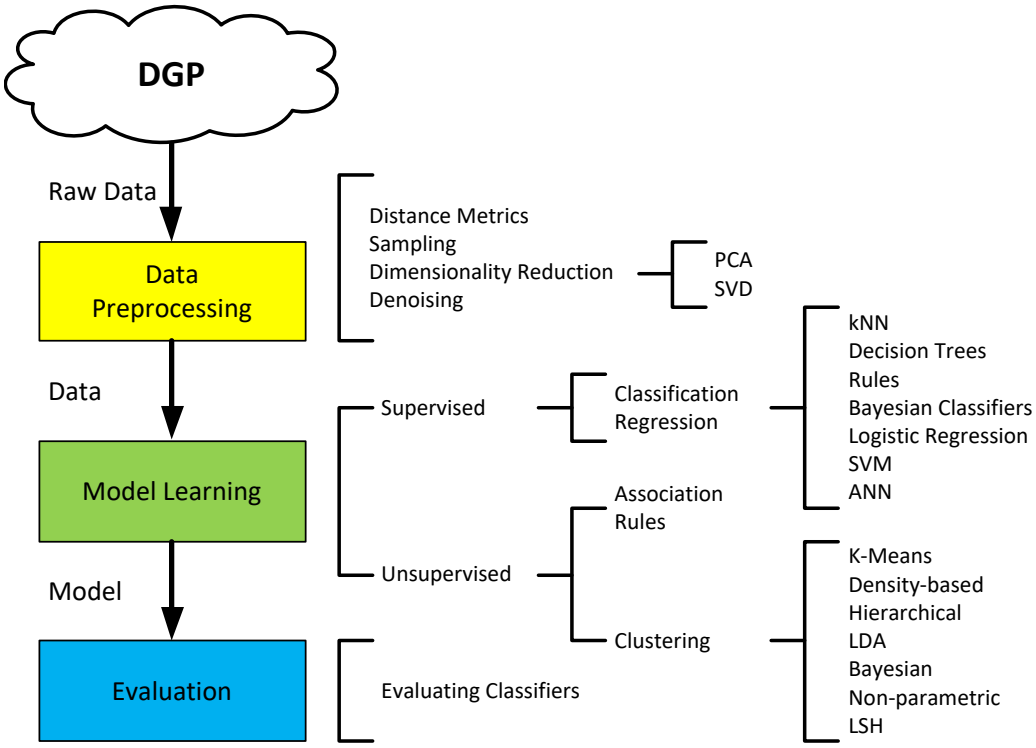

Figure 2.2: Data Mining methods used in recommender systems

2.4 Ratings

Ratings are values of R. They can be explicitly stated by users or derived from any available data which may reflect user preferences (implicit feedback). Explicit ratings are hard to obtain as most of users do not seem to like leave them. Ratings can be defined with different scales. Most known scales are continuous, interval-based, ordinal, binary, unary ones [34].

The rating matrix R have two important properties:

1. The *sparsity* of it is usually high because it's very unusual that most of users have contracted with most of items. It creates a challenge for computational effectiveness of algorithms.

2. The distribution of ratings among items frequently has the *right long tail* which is a consequence of that only a small portion of items are rated frequently. Sometimes, it is called the *popularity bias*. It was shown in [76] that some of the most accurate algorithms have strong popularity bias, which leads to recommending mostly popular items and that is not

always desirable in practice. Indeed, some analysis suggests that items from the tail (rarer ones) are more interesting for users [8], it improves the diversity, and therefore a good recommender system should be able to promote such items [19; 52] whereas most of recommender systems fail with that [21].

Ratings, as previously stated, are not always available in an explicit form because it requires efforts from users and usually raises a question of motivation of users to leave explicit ratingss. In fact, they are mostly unavailable in that form. So, it is required to algorithmically derive them from any kind of helpful data which is usually data about clicks, views, purchases etc.

As the number of Internet users grows and the data collection techniques are evolved, it seems that the ratio of explicit to implicit feedback is decreasing, thus, the motivation to use the implicit part of feedback is increasing especially given it becomes technologically easier than in the previous years. This view supported with an observations that there are more and more papers on how to make use of the implicit feedback. Example of CF-based recommender system using implicit feedback is shown in [33].

There are some distinguishable properties of the implicit ratings

- No fixed neutral point. Implicit feedback may carry information of both negative and positive attitude of an user towards an item but it's not clear how to separate them.
- No obvious preference. Numerical values derived from implicit feedback not necessarily reflects user's preference. Therefore, it is required to do that numerical derivation with respect to some confidence.
- Noise. Implicit feedback is more noise because it may include much more factors beyond just user's opinion. This disadvantage is compensated to some degree as implicit data is easy to acquire in large quantities and it does not require any efforts on the user's side.

Hence, working with the implicit ratings is more difficult. Certainly, both types of ratings are not mutually exclusive and can be combined to achieve the best results.

There is another notable challenge around estimating user preferences. In case of two persons sharing same computer and/or same user account, it is near impossible to distinguish those persons.

There are also physological aspects as in true user preferences. There are studies that have shown smae users can rate same items differently for no apparent reasons.

2.5 Effectiveness

In order to truly maintain value of a recommender system for its users, the former needs to be evaluated in terms of its effectiveness. It is another broad topic in the field. The evaluation depends on methods used in a recommender system though in overall it shares similarity with that of classification and regression modelling which is due to the fact that both the collaborative filtering and the content-based methods can be considered as a generalization of classification and regression modelling. However, there are usually several evaluation metrics as a single criterion cannot capture all aspects of a recommender system's effectiveness. Excellent overview of evaluation methods can be found in [32]. They can broadly be divided into *online* and *offline* ones.

In an online system, reaction of real users is measured in association with presented recommendations. The users may be actively recruited for the evaluation or they may not be aware of it. In the first case, users are directly asked to reveal their preferences by interacting with a recommender system; though it's difficult to ensure users represent the population, and that they are not biased by the knowledge their reaction is measured therefore results obtained in such the evaluation design cannot be fully trusted. An example of such evaluation is discussed in [37; 45]. In the second case, users are not aware of the evaluation, they use a recommender system in the natural course of affairs. In this design, users are randomly organized into two or more samples to conduct the evaluation of different algorithms. Typical aggregative metric used here is the *conversion rate* which reflects the frequency with which users select recommended items. This method is also referred as A/B testing, it is frequently used in commercial recommender systems and very similar to those found in clinical trials. Its drawback is that it requires large number of users which may not be feasible at the startup phase of system. Discussion on conducting this type of evaluation can be found in [20].

In offline testing, the evaluation is designed around a historical data such as ratings. The main advantage of this approach is that the available data

being ”a frozen history of system” can be used for benchmarking of different methods without having to involve into it any real users. A famous example of such evaluation design is the Netflix Prize [12]. Offline methods are the most used ones based on the literature reviews [1; 57; 62]. The main disadvantage of this approach is that the history may not reflect future state of things: user preferences may change in the future and the historical data may not contain information about that.

Effectiveness of recommender systems at first sight may appear as measured by *relevance* of recommendations. Which is not far from the truth but there are other important metrics [see 6, page 4] to be taken into consideration: *novelty* - revealing something new an user has not seen yet [18]; *serendipity* (”lucky discovery”) - revealing something truly unexpected, surprising; *diversity* - revealing wide range of items not just items though of high relevance but of limited scope. There might be more specific goals depending on context of a recommender system as they are usually established and evolved by business which are driven by business context-specific goals. All these metrics collectively can be considered as *accuracy*.

So, in a broad sense it’s a complex metric as in the effectiveness.

In a more narrow sense, the prediction accuracy, especially, in the offline testing is usually measures with MSE or RMSE with errors being prediction errors if they are on the interval or ratio scales. If predictions are ranks i.e. they are on the ordinal scale, rank correlation coefficients, ROC are used instead.

2.6 Collaborative Filtering Approach

In the collaborative filtering approach users are seen as a community which can collectively put items for its members in order in respect to their preferences.

Here, the concept of distance or similarity has found its application, in the class of neighbourhood-based algorithms. They rely on the principle that similar users tend to share similar behaviour patterns in terms of item rating so that similar items get similar ratings.

The concept of *distance* plays significant role in everyday life, it’s an universal concept to measure closeness (similarity) of two objects in some space. There are probably hundreds of various measures of distance from

very broad range of applications [25]. In the world of recommender systems, three distance metrics are popular, they are the cosine similarity, the Pearson correlation coefficient and the Mean Squared Difference (MSD)similarity .

In applying that concept, it's possible to depart from either *user* or *item* which leads to two different types of neighbourhood-based algorithms:

- similarity is between users, recommendation for an item i for a particular user u is based on that particular user's neighbours' weighted ratings against i,
- similarity between items, recommendation for an item i for a particular user u is based on weighted ratings for items which are neighbours of i.

The first way is thought to provide more balanced recommendations as it involves opinions of other users, the second way is very user-centric and may lack the important properties of effectiveness such as diversity and serendipity.

The model-based collaborative filtering approach try to employ the concept of a model to meet the recommendation goals, building a model involves different machine learning methods for there are many to consider [49], though being applied directly they provide suboptimal results, they are prone to the effect known as overfitting, which, in the context of recommender system, means that a model will be good at restoring recommendations it has been trained for but will perform poor at generating new recommendations. This is due to the nature of extreme sparsity of the rating matrix.

One of famous competitions in the field, run by Netflix [12] stimulated many variants of machine learning methods adopted to the task [10].

A remarkable method which gained a lot of researchers' attention and was significantly developed specifically for the sparsity issue is the matrix factorization method [7; 39]. This method works by trying to factorise the rating matrix R into two lower dimensional matrices U and V, so that $R = U \times V$. The dimensions of U, V are much smaller than R and they loosely can be thought as "all items by some taste dimensions" and "all users by some taste dimensions" versus "all users by all items". These "taste" dimensions are called latent or hidden features. Working the R as a factorized composition makes calculating recommendations much more computationally effective and provides some ability to reason about those "taste" dimensions.

Matrix-factorization approaches can be more accurate and scalable over the

neighbourhood-based collaborative filtering one, they involve more information into building latent factors and it may help to uncover more information for a specific user. Over the years, matrix factorization has become a popular technique due to its accuracy and, importantly, scalability.

Both the types heavily relies on the user-item rating matrix R. The task of generating recommendations can be re-formulated then as the prediction task of filling in the missing entries in R, the filled-in values are essentially recommendations. The specifics is that there is no distinct separation of variables into independent and dependent ones, all them can be treated as both types, and missing values in R is expected.

2.7 Ensembling Approaches

So, it can be seen there are different clearly distinguishable approaches to solving the same task. Naturally enough, researchers explored into possibilities to ensemble, merge different models into one, a hybrid model. Approaches used are quite similar to those in other applied fields of Machine Learning i.e. results from different models are summarized in a way meaningful for their consumers. Specific implementations were systematically reviewed in [15; 16] and recently in [77].

2.8 Common Problems

The field is still evolving and there are open common problems in it which should not be ignored in implementations of recommender systems to mitigate the risk of getting inaccurate results.

An important assumption for recommender systems is that there is strong user identification to have ability to track an individual user's interests. But in many Internet-facing applications, this assumption may not hold especially if there is no strong user authentication built-in. One individual may interact with items presented by such an application by means of different devices (PC, laptop, tablet, smartphone etc), and an application is likely to consider such an individual as separate persons.

In case of collaborative filtering methods, initial set of ratings is often not enough for predicting ratings with adequate accuracy, sparsity of the user-item

matrix is too high. It is the cold-start problem.

As recommender systems are used to essentially attract customers, some malicious agents on the market may try to impute incorrect information into recommendation systems of other agents to disrupt effective work of the formers. For example, by sending specific ratings to promote or to demote certain items. This inappropriate practice may lead to *untrustworthy* recommendations, and therefore requires recommender systems to be resistant to such attacks. In other words, to be stable and robust in the presence of such attacks. Various techniques for that are discussed in [44; 54; 66; 68; 71; 72].

Information gathered for and inferred by recommender systems can be highly sensitive as in revealing interests of users such as political opinions, social connections, personal preferences. Special care needs to be taken to avoid non-authorized access to that information. It was shown in a study a person can be identified from data on their behaviour.

It's advisable to take those problems into account when designing a recommender system and find appropriate methods to mitigate corresponding risks.

2.9 Recommender Systems for Education

Recommender systems, initially well-received and established in e-commerce, have found their ways in online education platforms as well, they are used to recommend to users "what to learn next". Which is deemed to be very practical as users of such platforms are usually not sure themselves about that.

Recommendations for technology enhanced learning scenarios tend to differ from those in other areas because of different goals, the former should be guided by education objectives, not by only preferences of user [64].

Technology-enhancement learning (TEL) is an application domain that generally addresses using technologies to support teaching and learning activities. But that was only since the year of 2010 when researchers from the TEL community jointed their efforts with ones from the field of recommender systems [see 59, page vii]. Several literature reviews [46; 73; 78] suggests increasing popularity of the subject among researchers. Today, there are tens or

more of so-called massive open online course platforms (MOOC[3]), the digital learning content is increasingly produced and published on them.

TEL designers try to improve learning experience of their users by utilizing methods from the field of recommender systems. Recommender systems for educational purposes have their own specifics. In general, it's more difficult to generate right recommendations for a learner than for a buyer. Items in the education domain are fundamentally interdependent and sequential. An item or a set thereof may suit a student at one time but may be completely useless earlier or later.

Over the years researchers employed different ways to build efficient TEL recommender systems. A state-of-the-art review [51] over 82 reported implementation demonstrate good diversity in techniques. It appears that various modification of collaborative filtering and content-based methods were popular in the beginning of the 2004-2014 period with gradual shift of researching efforts to context-based methods with using graph algorithms. Evaluation criteria for TEL recommender systems differ from traditional recommender system criteria and include specific learning related ones such as effectiveness and efficiency of the learning process.

Gartner in their analytical report [17] classify Adaptive Learning and Big Data as factors going to mature in Online Educations over the preceding years (from 2017) Figure 2.3. Recommender Systems perfectly fall under the category of Adaptive Learning and, loosely speaking, Artificial Intelligence Education Applications.

2.10 Design Considerations

Recommender systems are practical things by their nature, and their development are historically driven by the industry, they are created and developed mostly by practitioners. However, there is limited information on how real life recommender systems are designed, implemented and operated. Possibles reasons might be that they are owned privately and their owners do not want to share much details of them in fear to lose competency, practitioners do not tend to publish articles as much as researchers.

[3] https://en.wikipedia.org/wiki/Massive_open_online_course

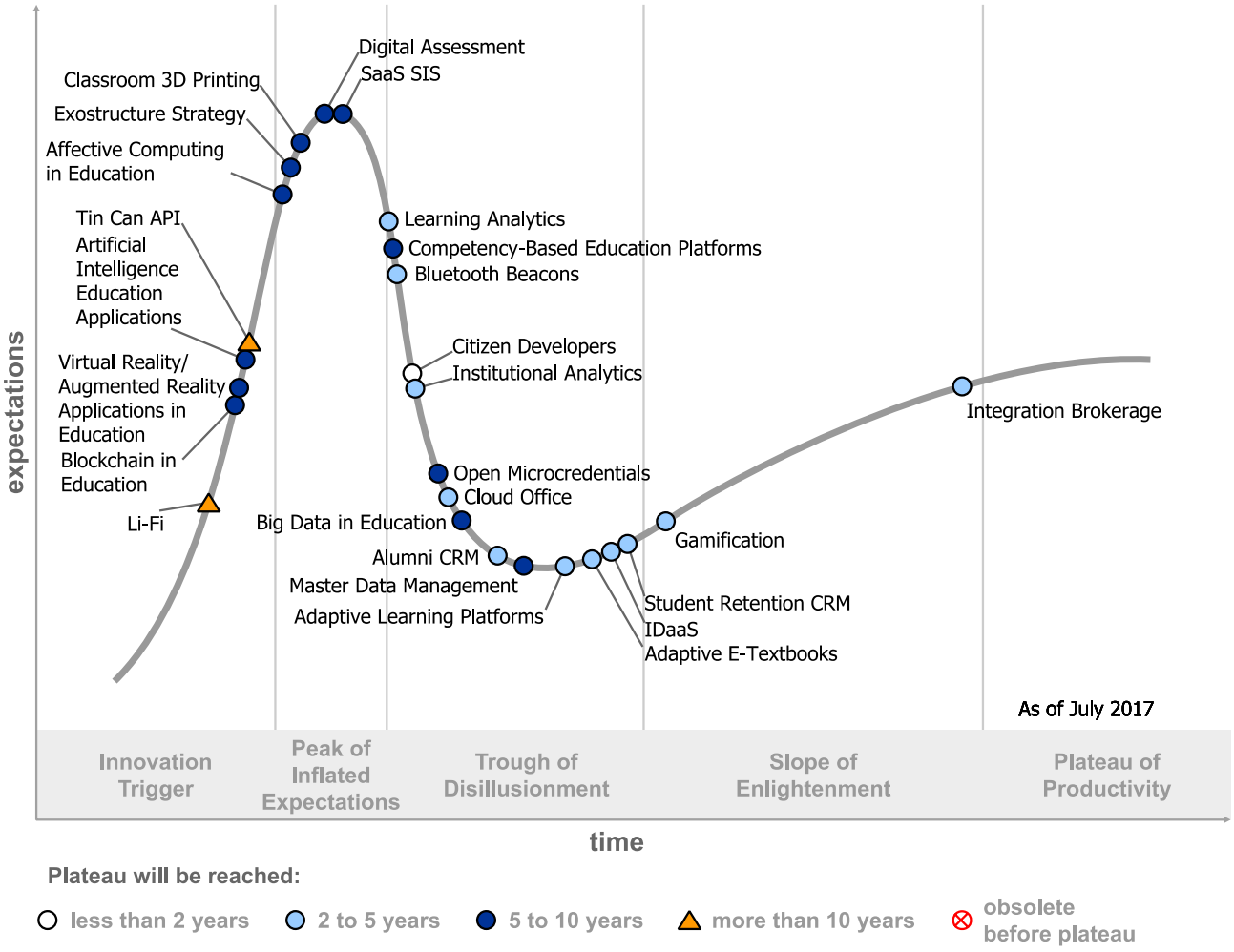

Figure 2.3: Gartner Hype Cycle for Education, 2017

Algorithms used are heavily computational in nature, and they are generally required to handle high-frequency, sometimes high-dimensional categorical data, which leads to significant engineering efforts to make a commercial recommender system be responsive and scalable according to user demand [65].

There are some information though on implementations [24].

Based on the above, recommender system should operate according to business goals and be able to support business-centric metrics. Seemingly, a modern recommender system to meet typical set of business requirements needs to be a hybrid and to harmonically co-exist with other IT systems aimed to maintain user experience.

The author's conclusion if placement of recommender system in a data-driven organization is shown in Figure 2.4.

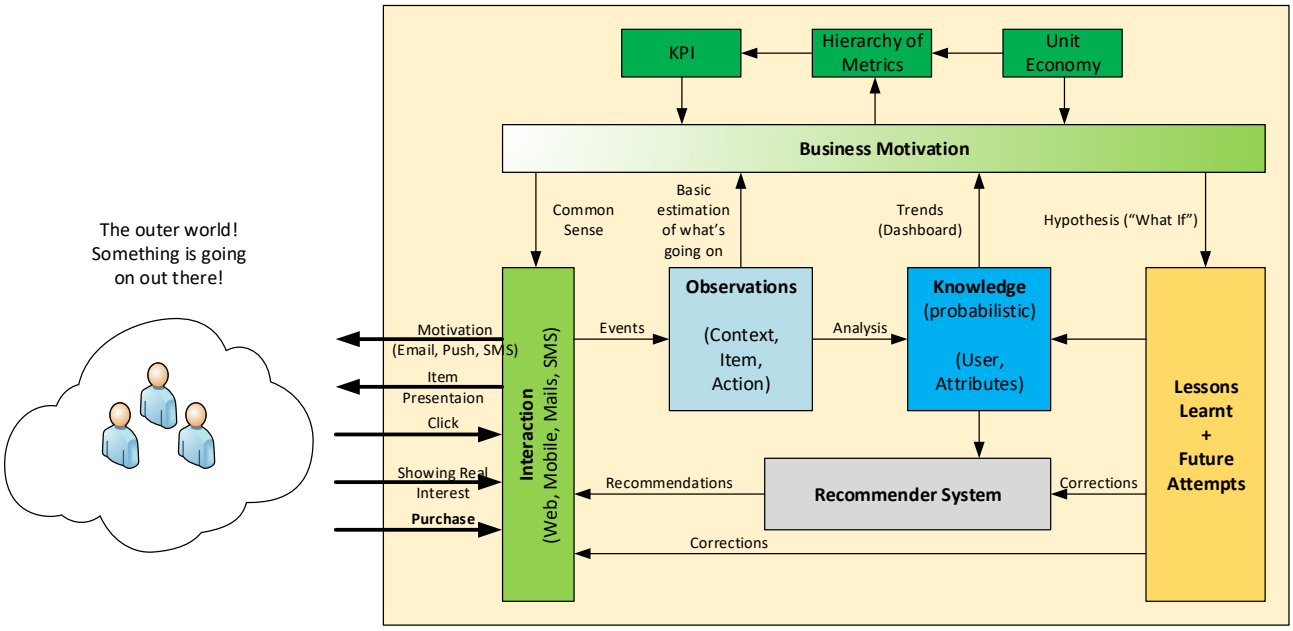

Figure 2.4: Placement of recommender systems within a data-driven organization

2.11 Technology Considerations

Implementation of a recommender system involves wide range of methods and techniques of handling and analysing data. Data is generated and captured at quite diverse set of computing and networking layers around users like web servers, user devices, exported from social networks, advertisement systems, web tracking systems. All that data are pushed and pulled in different streams, different formats with different delays respecting original occurrence of events in the data generation process. Therefore, it's critical to preprocess data and organize it into a consistent stream of events before passing it to a recommender system. Here, well-known ETL methods are used.

Most of recommender systems reportedly process incoming data in batches with a typical time gap of hours between receiving data of what happened and generating recommendations. That is getting obsolete now. New generation of recommender systems are built to generate recommendations in the nearly real time, and industries rely on the so-called 'Big Data' technological components in here [2; 3; 9; 74].

Technology aspects of recommender systems implementation is poorly covered by researches, even worse than the design questions. The main reason

is probably the same, it is a very applied question that most of researches are not interested. Therefore, the author has relied more on industry-oriented practice than on research papers in considering the technology aspect. Proper implementation of industry-grade recommender system raises many questions e.g. how to extract and transform necessary data, how to keep it in a persistent store, how to process it in a timely manner, how to automate processes. Answers and solutions to that should support frequent changes in the data flow (up to structural changes).

Some basic considerations with examples were shown by Toby Segaran by [67] however it does not account for the Big Data three to four 'V' factors[4]: Volume, Velocity, Variety and Veracity. Sometimes there is designated the fifth 'V' factor: Value[5].

A popular agreement for choosing technology components for the implementation is to use those from the well-regarded Apache Hadoop technology stack [31; 74], they are popular, proven by practice. The stack is quite diversed and there are challenges to select only necessary set of components out of it. Usually, it boils down to several significant questions:

- What is the balance between having to support batch processing vs. online processing?
- What are services (other technology components) that need to work with the Hadoop components?
- How will the cluster be maintained?

There are challenges between matching batch-oriented components and online-oriented processing components but there are recent additions to the technology stack which aim to compromise between these two contradicting worlds at least in terms of the storage [56], namely, Kudu[6] is advertised as one of such solutions:

> A new addition to the open source Apache Hadoop ecosystem, Apache Kudu completes Hadoop's storage layer to enable fast analytics on fast data.

[4] https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data
[5] https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/
[6] https://kudu.apache.org

For optimizing the maintenance costs, it is a common practice to go away from manual composition and integration of the Hadoop stack components but choose already integrated and tested set of them, called a distributions. The popular ones are published and supported by Cloudera[7] and HortonWorks[8]. Also, it is common to build technology infrastructure in a cloud, using either IaaS or SaaS services, this implementation model is going to be stronger according to Gartner Research [47]. AWS[9] by Amazon and Azure by Microsoft[10] are probably the market leader here. Most popular components are listed in Table 2.1 with promising ones marked in bold.

Table 2.1: Big Data technology components

Layer/sPhase	Components
Ingest/Propagate	Apache Flume, **Apache Kafka**, Apache Sqoop.
Describe/Develop	Apache Crunch, Apache HCatalog, Apache Hive, Apache Pig, Cascading, **Cloudera Hue**, DataFu, Dataquise, IBM Jaql
Compute, Search	Apache Blur, Apache Giraph, Apache Hama, Apache Lucene, Apache MapReduce, Apache Solr, **Cloudera Impala**, HStreaming, SQLstream, Storm, **Spark**
Persist (File System)	Apache HDFS, IBM GPFS, MapR File System
Persist (Serialization)	Apache Avro, Apache Parquet, RCFile, SequenceFile, Text, Trevni
Persist (DBMS)	Apache Accumulo, Apache Cassandra, Apache HBase, **Apache Kudu**
Analytics, ML	Apache Drill, Apache Mahout, Datameer, RHadoop, **Spark ML**

Capturing, storing and processing data requires data schema, thus, data modelling. There are two opposite camps here: from 'schema-less' solution

[7] https://www.cloudera.com/documentation/enterprise/5-16-x.html
[8] https://hortonworks.com/products/data-platforms/hdp/
[9] https://aws.amazon.com/
[10] https://azure.microsoft.com/en-us/

(NoSQL[11]) to fully normalized forms [29], solutions in the middle are traditional DWH schema modelling e.g. the star schema suggested by Kimball [36].

2.12 Future Development

The future development of recommender systems was outlined in [38]. One of directions named is having a research infrastructure as part of recommender system. It is because offline tests are not so valuable as real life tests like A/B testings therefore reseachers seeking to improve effectiveness of a recommender system should be able to test different scenarios online [30].

The success of applying deep learning to many prediction and classification tasks motivates researches to using it in recommender systems [75], it's also seen as one of promising directions esp. it allows using more features about items.

[11] https://en.wikipedia.org/wiki/NoSQL

Chapter 3

Methodology

Given the project objective, it is foreseen to be a multidisciplinary IT project involving expertise from different levels. As with most of data-driven projects, the the cross-industry standard process for data mining (CRISP-DM[1]) methodology, as explained in [55] is largely applied here to arrive to a first version of recommender systems. It is very likely it will evolve i.e. it is not an one-off modelling therefore the methodology should support building such a solution it can be developed further if required.

In the context, it can be broken into several large steps:

- Gather and consider business requirements, put them into details and clearness with experts of involved domains.
- Study the system architecture to gain understanding how it works in the grand scheme of things.
- Conduct EDA to understand what data is available, how it can be used to the purpose. How the critical entities as *user*, *item* and *rating* are presented.
- Make an architectural decision how recommender system as a new system component will be integrated with the rest.
- Investigate into the technology aspect, decide on what technology components will be used for the implementation. It is critical to allow project's results be usable and adequate for further developments.
- Using the available test as a 'train' one, decide on the type of recommender system and prototype it with attention to such principal factors as the cold start, its computational complexity and performance requirements.

[1] http://crisp-dm.eu/

- Implement the first version of solution and collect first

The methodology is quite traditional for IT projects with specifics for its modelling part. Certainly, some steps may require revisiting them as more understanding is gathered.

Chapter 4

System Architecture

4.1 Requirements

There is a site with educational materials, its users are entrepreneurs from SMB. They register on the site (possibly, via different channels), log on and browse materials (items) on it. The items are blogs (texts) worth 5 and more minutes of reading and videos usually from 10 to 60 minutes long. It was required to created a recommender system for it.

As any other commercial project, the current one was started with formulating major requirements, see Table 4.1. Strictly speaking they are not all about the recommender system but the list includes them anyway as they which should be taken into considerations in association with relevant data analysis.

That was agreed with the system owner that a metric to evaluate performance of recommender system form business point of view is positive changes in DAU/MAU under the condition there are no other influential factors, i.e. adequate recommendations should be associated with an increase of DAU/MAU. Another metric is the coverage of catalog (%).

4.2 Data Flow

In order to design a data-driven solution and successfully couple it with the rest of system, a good starting point would be to understand data flow in that system i.e. what are origins and transfers of in a system. The author has investigated into this and gained understanding of the data flow as shown

Table 4.1: Functional and Technical Requirements

№	Requirement
1	Recommender system must technologically be compatible and integrated with the other IT components that support the web site.
2	Recommender system must work autonomously, without requiring human attention for its normal work.
3	Implementation should be future-proof in terms of performance i.e. it should have capability to run near real-time and to scale horizontally in the future if required.
4	It should be possible to temporarily switch off recommendations to conduct A/B testing.
5	History of user activity should be organized as time-dimensional series of events for ad-hoc analysis.
6	Event may have a marketing tag attached (on the front-end) to it. The system should allow storing it.
7	User activity should be accessible with a SQL92-complaint client for its ad-hoc analysis.
8	Technology stack should be a matured product with a technical support.
0	It must comply with Federal Law of 27 July 2006 N 152-FZ on personal data.

Figure 4.1 in the notation of ArchiMate[1].

There are several data sources of different types which should certainly be considered as data sources for the recommender systems, namely, an RDBMS that stores information of users and items, web server logs that keeps information of hits. There are other sources like CDN web logs, YM hits and statistics about email distribution. They are not immediately required for enabling first versions of recommender system but can be useful for later iterations, that requires additional investigation.

[1] https://www.opengroup.org/archimate-forum

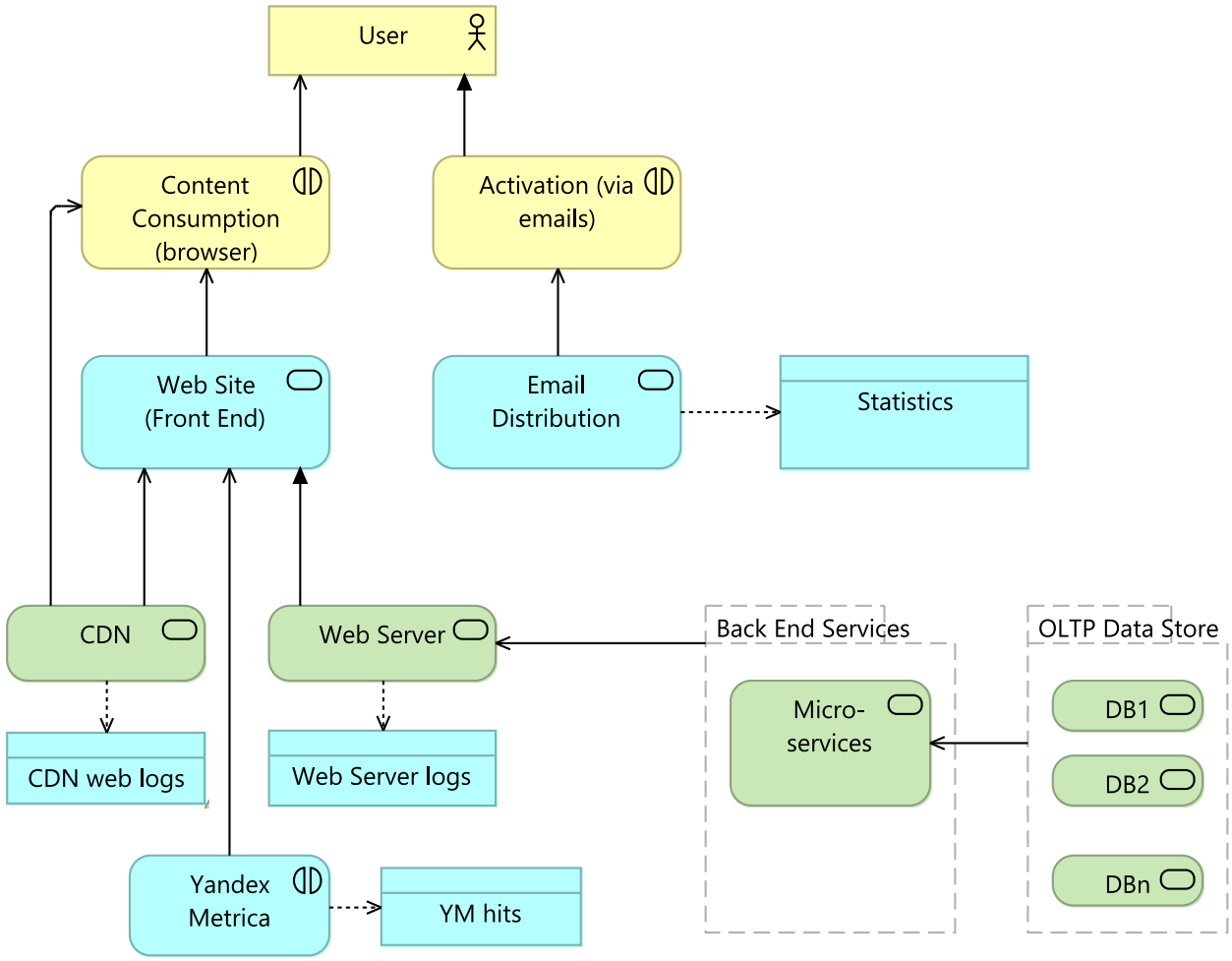

Figure 4.1: Principal Data Flow

The web site front-end is developed with JavaScript, to deliver content to an user's browser there are calls to back-end services (microservices, developed in Java). There were discussions with developers on how recommendations as new data type entities can be integrated with the platform and it was agreed they should be stored in a database accessible by relevant microservices so as to the front-end can fetch recommended items via them when required on the UI level i.e. in a browser when an user browse relevant pages.

There were considerations about factoring email distribution in the recommender system so as to it would be possible to automatically advertise recommended items via email but subsequent investigation of that data source ruled it out because it is an external system without good measures to integrate with it.

4.3 Data Schema

The selected data sources are summarized in Table 4.2 in no particular order.

Table 4.2: Data Schema

Name	Technology	Source Description
User	RDBMS	Database with tables, they keep user identifiers and their profiles.
Item	RDBMS	Multiple databases (one per each type of item) with metadata information about items including their identifiers.
Rating	RDBMS	Database with explicit ratings on the five star scale.
Favourite	RDBMS	Database with lists of favourite items as chosen by users.
Web logs	ElasticSearch	Records of web calls to microservices, they consist of tens fields including timestamp, IP address of origin, type of call, destination resource address, identifier of an user in case of their correct authentication. There are many types of records in the source, some of them represent 'clicks' of users forming so-called 'clickstream'. This is a valuable and the only one source for information on implicit feedback.
Web events	Kafka/JSON	Records of events raised on the front-end as configured by its developers and web site analysts. The records contain some information about user interactions, namely, 'keep alive' signals.

Chapter 5

Evaluation of Available Data

There are two key entities involved in the data generation process: users U and items I. The former interact with the latter, and that creates the third entity type: interactions. From the project's point of view an interaction is either explicit rating or a click of an user on an item.

5.1 Primary Data

The system was launched in February, 2018 and there are records of interests available from this date. Time tracking of user activity was launched much later, and relevant data is available since October, 2018. See Figure 5.1 for the number of records. There are peaks, they are mostly due to some one-off events. For example, the peak of new items is February, 2018 is due to launching the system with initial number of items, the similar peak in April, 2019 is due to launch of a new product consisting of large number of videos. The peak for user growth in December, 2018 was generated by a massive advertising campaign. The hit numbers naturally follows user activity to some extent, however they also contain a lot of calls between microservices happening in the background, changes in that part of hit numbers can be driven by changes in code of microservices i.e. not associated with user activities.

An interim conclusion here is that there are millions of web call records, they are the single source of information of the implicit feedback but that information is buried within, need to be extracted, associated with users and items, thus, converted into data of type a recommender system can work with. As information about user activity reportedly contains the time tracking component only since October, 2018, there is a decision required how it can be combined with the rest of data about hits (basically, with just 'clicks').

Figure 5.1: Raw data: number of records

5.1.1 Preprocessing

The initial data preparation was quite an effort, and did cost up to probably 50% of the total time for the current work. But one can't rely pass this stage for a data-driven solution, besides, it's a valuable experience as it allows understanding the data flow and the data schema much better.

Namely, the data preprocessing stage required to develop a library of routines to fetch data from the data sources and combine it all together. The mostly notable experience was with decoding web log entries into meaningful hits so as to call it implicit feedback data. The reduction was as such as that only approx. 1% of records turned out to be of interest, the rest carrying same information from the project's point of view. This is because of the system architecture - a refresh of web page initiates tens of calls to microservices, direct as well as between themselves. That led to raising a concern with the system owner that the recommender system would require its own source of data about user behaviour, and it was agreed to create a system of interaction-centric events sent over Kafka whenever an interaction happens. This approach is still in the development.

The data was additionally cleaned from test activities and the left web log records were matched with users and items so as to create a rating matrix.

Finally, there were approx. 400k users, 2600 items and 6 mln. hit records for an initial analysis.

5.2 Users

There are users (registered and authenticated persons) and visitors (non-registered or unauthenticated persons) on the site. The *user* entity's attributes are listed in Table 5.1.

The possibility of matching a unauthenticated person correctly with an user account or even with a set of user accounts is a complex issue. It is difficult to identify a person in Internet correctly, cases when one person can interact with same online resources being unauthenticated as well as authenticated, or when multiple users shares same device are not rare. Historically, web developers eager to distinguish users so as to track their activities separately relied on straightforward techniques like using the source IP address and/or

Table 5.1: User Attributes

Name	Type	Description
Identifier	Mandatory	System unique identifier (integer)
Email	Mandatory	User email
Creation date	Mandatory	User account creation date
Registration channel	Mandatory	Channel used during the registration (it's used for marketing purposes)
Name	Optional	Full name
Birth date	Optional	Birth date

storing certain values in cookies. These techniques are less reliable these days than they were used years ago as the Internet network architecture is getting far from being 'plain' and users are more aware of cookies and use measures to avoid being tracked.

The current state of the art in this field consists in running series of hardware- and software-dependent procedures in a client's browser so as to get results quite unique for that particular browser on that particular machine, so-called fingerprints; in analysing typing behaviour, mouse movements, scrolling behaviour etc; basically, in analysing series of events raised on an user devices due to user activities or triggered by code itself to compute uniqueness of its computing environment. Successful examples of these online tracking approaches were originally shown in [48] and later in [27; 70].

This kind of tracking identifiers are not available in the data. Moreover, using them raises ethical concerns therefore for the data analysis I relied only the user numerical identifier to distinguish users among themselves.

The DaData[1] data validation service was used to estimate the sex of an user by their full name (for those users who decided to leave it), the distribution for a random sample of 1000 users was 63% of men and 37% of women, these figures correlate with the Russian Federation Federal State Statistics Service (Rosstat[2]) [80].

[1] https://dadata.ru/
[2] http://www.gks.ru/

5.2.1 Distribution of First Time Visit Delay

The distribution of first time visit delay since creation of an user, in days, is shown in Figure 5.2. According to it users usually start working within several days with the site after having confirmed their registration. That's interesting there are still cases when users may make their first visit only after several days. Subsequent discussion with the developers suggested there were issues with correct sending user authentication header to the web server. That needs to be investigated additionally. For the time being, we can conclude there is a risk user activity information not properly captured for all users and the affected users may not receive their personalized recommendations rather sooner than later.

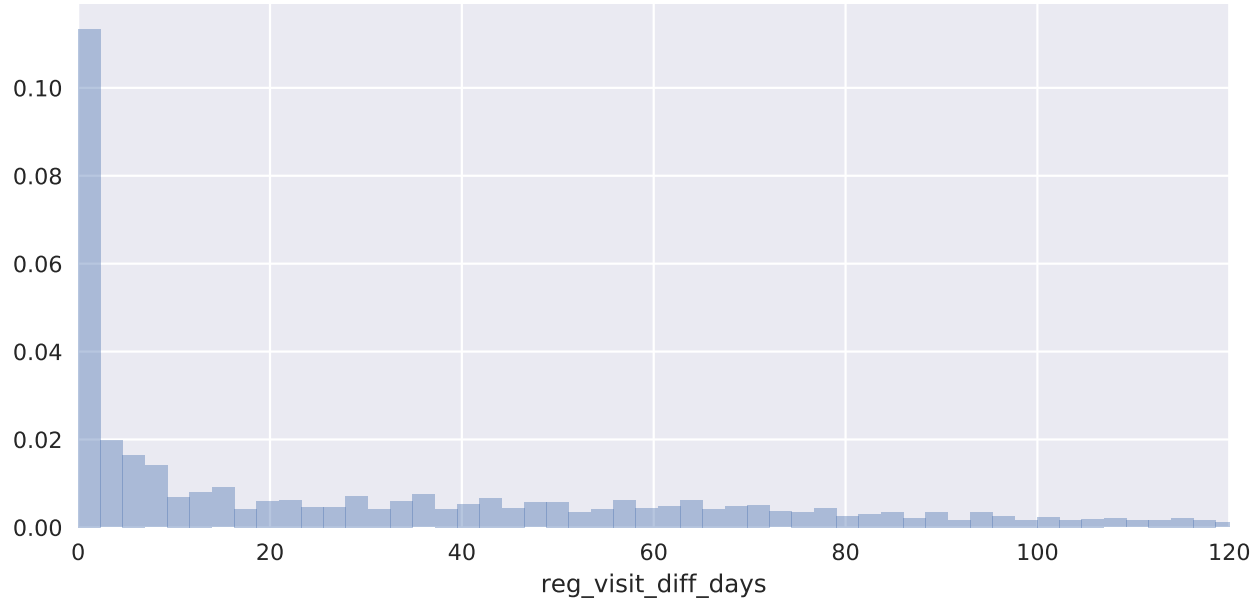

Figure 5.2: Distribution of First Time Visit Delay

Sadly, there are web records for activities of only 15.5% of registered users. Most of registered users do not seem to use the site or are not properly tracked. The former reasons seem to prevail due to specific registration procedures (by partners).

5.2.2 Distribution of Visit Days to User Account Age

For users with visits, the distribution of ratio defined as the number of visit days for an user account to the age of it (in days) was calculated to understand

how frequently visiting users tend to work with the site. The ratio is defined as

$$r_{visits\_to\_age} = \frac{\#\{days\ user\ visited\}}{\#\{days\ user\ account\ exists\}}. \tag{5.1}$$

The distribution of $r_{visits\_to\_age}$ is shown in Figure 5.3. Its mean value is 0.026 with the 0.25/0.50/0.75-quantiles as 0.004, 0.008 and 0.022 accordingly. So, half of users visited the site <1% of days within their lifetime on it. That's considerably low.

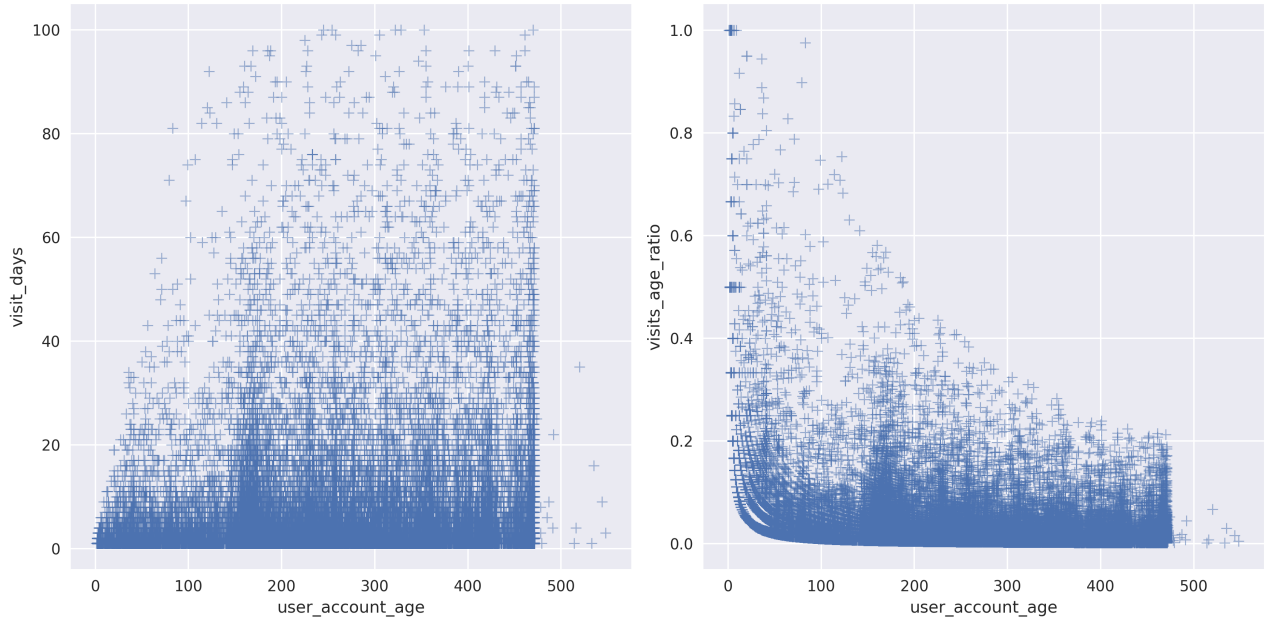

Figure 5.3: Distribution of Visit Days to User Account Age (absolute and relative)

5.3 Items

There are items of several types. For the first version it was agreed that the recommender system should not take them into account because it is not quite clear whether this system of types stay long. Probably, all items will be reorganized into just two main types: textual (blogs) and videos. All items are organized into twelve *categories* $\{c_i : i = 1..k, k = 12\}$ which are aimed to reflect different aspects of an entrepreneur's interest for educational materials.

The *user* entity's attributes are listed in Table 5.2. The distribution of items amount their types and categories is shown in Listing 5.1.

Table 5.2: Item Attributes

Name	Type	Description
Identifier	Mandatory	Integer triple uniquely identifying an item within the system
Media type	Mandatory	Item type: blod, course, event, lesson, lifehack, trajectory, video
Creation date	Mandatory	Creation date
Category	Optional	There are twelve categories each representing different stages in the life cycle of an entrepreneur, they together are meant to cover the whole life cycle, its optional though it is presented for 92% of materials.

Listing 5.1: Item types across categories

item_type_code	blog	course	event	lesson	lifehack	trajectory	video	All
category_code								
_not_assigned	0	0	14	0	283	0	10	307
advertising	29	29	0	286	0	0	20	364
crisis	4	7	0	36	0	0	13	60
development	10	15	0	171	0	0	10	206
finance	22	17	0	105	0	0	25	169
hr	25	20	0	233	0	0	22	300
inspiration	34	1	0	3	0	0	36	74
law	37	18	0	97	0	0	33	185
management	20	14	0	103	0	0	14	151
pe	17	14	0	127	0	0	13	171
risks	17	13	0	133	0	0	14	177
staff	30	12	0	113	0	0	25	180
start	17	22	0	236	0	34	20	329
All	262	182	14	1643	283	34	255	2673

5.4 Ratings

5.4.1 Explicit

There are explicit ratings set on the five star scale and, as discussed above. Users may also mark items as their favourites. The data is small though, thus,

0,57% of users rated 20.1% of ratings and 0.51% of users flagged 25.8% of items.

The distribution of explicit ratings is shown in Figure 5.4, its shape suggests that this specific user community tend to leave positive ratings more frequently that negative ones.

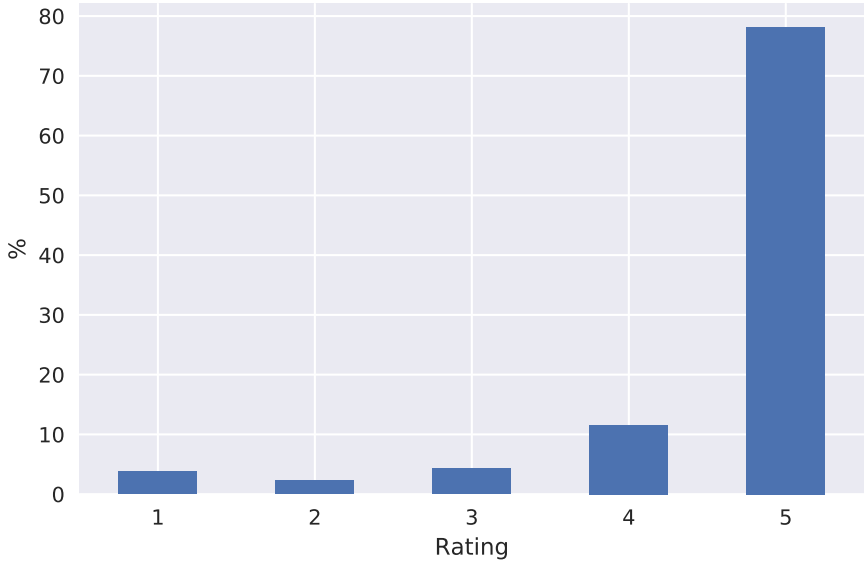

Figure 5.4: Distribution of ratings

That'd be interested to check the rating distribution for fraudulence according to Benford's Law [11; 40] but I have not been able to find a research supporting it can be applied to the five star scale ratings.

5.4.2 Implicit

The implicit feedback can presumably be found in the web log entries (an example of such entry is shown in Listing 5.2) and that's true to some extent. It is possible to extract that an user clicked on an item along with the click's timestamp.

Listing 5.2: web entry log

```
nginx: {
"@timestamp": "2019-05-18T14:44:22+03:00",
"remote_addr": "128.124.204.28",
"remote_user": "-",
"http_host": "uni-tracking.dasreda.ru",
"bytes_sent": 266,
```

```
"body_bytes_sent": 0,
"response_code": 200,
"request_time": 0.012,
"request_length": 2470,
"request_uri": "/v2/track",
"request_method": "POST",
"http_referrer": "https://****.**/learn/video/67",
"http_user_agent": "Mozilla/5.0 (Linux; U; Android 7.1.2; ru-ru; Redmi 4X
   ↪  Build/N2G47H) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0
   ↪ Chrome/61.0.3163.128 Mobile Safari/537.36 XiaoMi/MiuiBrowser
   ↪ /10.7.2-g",
"http_origin": "https://****.**",
"http_user_id": "431683",
"upstream_addr": "10.217.143.29:80",
"upstream_response_time": "0.010",
"upstream_response_length": "0",
"upstream_bytes_received": "178",
"http_x_forwarded_for": "-",
"http_ds_session_id": "ea347443-94b2-48e5-ab0a-a4cbb6b6cabe",
"http_source": "ui" }
```


There are two types of the implicit feedback. The first one is basically a click of an user on an item. The act of such clicking can be perceived as a positive rating for that item but it's not always true as an user might just open an item and navigate away from it not liking it. The second type is based on so-called heartbeat (the front-end sends a message every n seconds through the web server) so as to track that an user still keeps a certain item open. It allows getting the duration of time an user spends on a certain item, though it is prone to errors as well e.g. an user can open a page and leave it open for a long time.

The distribution of heartbeat timing for authenticated users is shown in Figure 5.5. It can be seen that according to it most of users tend to spend only several minutes on the site, there is a peak after 7000s because all values larger than 7200s where capped at this value.

5.4.3 Catalogue Coverage

The catalogue coverage according to the web log entries is 54.6%. That's a bit surprisingly low number given the site has been working for more than one year.

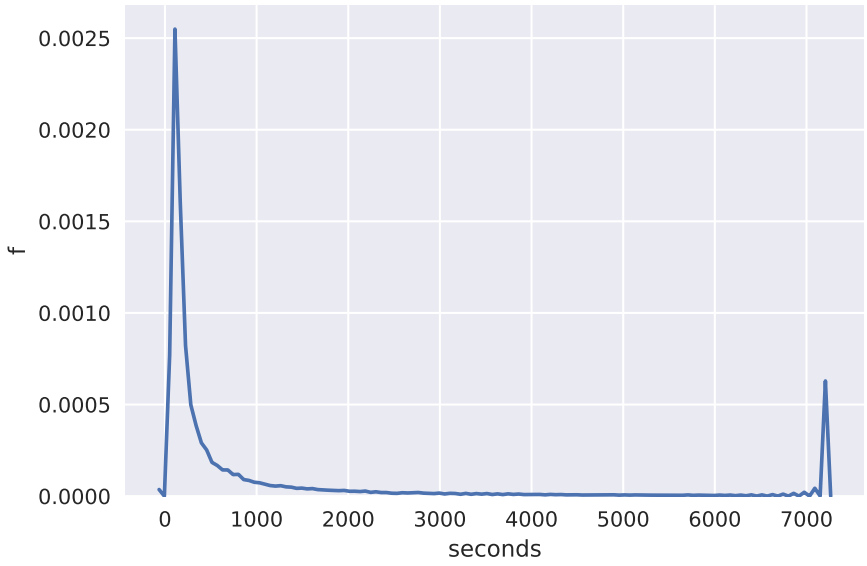

Figure 5.5: Distribution of heartbeat timing

5.4.4 User Interests

Naturally, users have different interests towards items and that'd be interested to how they are distributed. In order to 'measure' it, there is required to define a metric of interest. Here, the site navigation is such that users can easily pick up items of desirable media type and category, therefore one of those metrics can be distribution of user clicks or time spent across two properties.

This is left as an open question as out of the project's scope.

5.5 Conclusion

Users are not very active on the site, most of them do not even visit the site after their registration at a partner, and the visiting part of users opens it on less than 1% of days comparing to the 'age' of their user account. The catalogue coverage is also far from being complete being around 55%.

Items do not have rich metadata about them, at least, in a state ready for its usage in the recommender system therefore the latter should be able to work on the basis of only identifier of an item.

There are explicit ratings and favourites left by some users but their number is low comparing to the total user base (even to the visiting user base) and to the catalogue size. Therefore, it's important for recommender system to be

able to use the implicit part of feedback in order to compute recommendations for as many users as possible.

It's possible to cluster users based on the distribution of their interests towards categories. The practical usage case of it might be using the clusters for targetted marketing campaign as the categories have quite narrow practical meaning.

The EDA results were discussed with the system owner, it was decided to prototype a recommender system based on data available since October, 2018 as it did not make much sense using information of user activities older more than half a year. There was a valid argument that that historical data has some value as it can help to motivate some user to start using the web site again so the decision was corrected that for those users the number of hits can be converted into data compatible with the recommender system so as to generate recommendation for the no returning users if required.

Also, a number of enhancement requests were raised with system administrators and developers to improve the data collection in the system.

Chapter 6

Recommender System: Architecture

At this point, there should be some understanding of data in overall in the context of objective so as to start architectural (design) making on the recommender system and technology components for its implementation.

6.1 Recommender System Structure

The number of users is relevantly large (several hundred thousands), along with the requirement for near real-time (in subsequent versions) processing that suggests using a model-based recommender system. A memory-based recommender system can be unreasonably slow under the conditions due to the necessity to recalculate the distance matrix.

The explicit ratings optionally need to be merged with the implicit ones, the feedback data is quite noisy. Given that there will be required some filtering (possibly, configurable) before running data into a recommender system.

Based on the above, the author suggests the structure of recommender system as shown in Figure 6.1.

Raw data collected from different sources are transformed into explicit and implicit ratings, they are potentially merged together. Filters can be used to specify what part of data are filtered out e.g. if editors of the site do not want an item to be influenced by algorithmic recommendations. The initial set of filters developed during the design is listed in Table 6.1, they may also be called rules.

Model is a Statistics/Machine Learning model which can optionally be parametrised therefore there is the optimization algorithm to regulate model performance through those parameters, it is a standard approach of finding an optimum solution for cost function f over its parameters Θ:

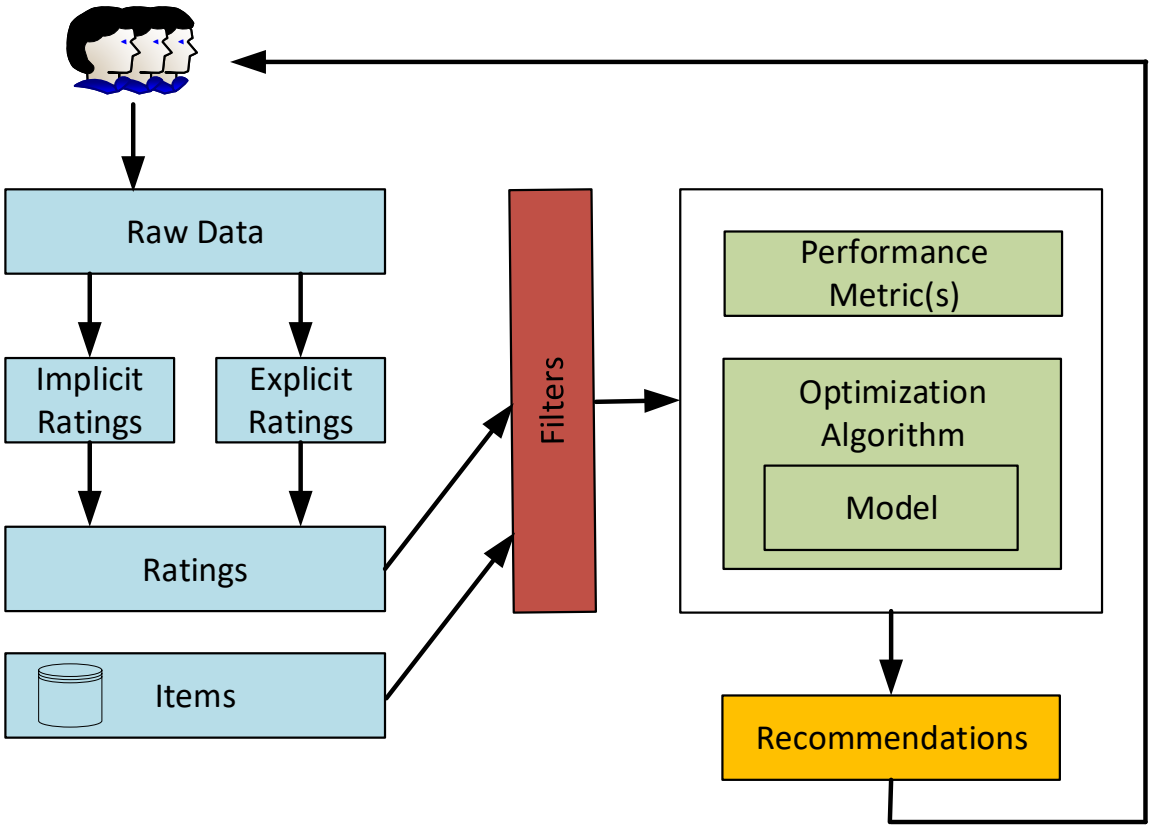

Figure 6.1: Design of Recommender System

$$\underset{\Theta}{\operatorname{argmin}} f(\Theta) = \{\Theta \mid f(x) = \min_{\Theta'} f(\Theta')\} \tag{6.1}$$

Instances of models are compared in terms of their performance by using different metrics. The most popular metrics for accuracy are the coefficient of variation (CV), mean absolute percentage error (MAPE), and root mean square error (RMSE).

$$CV = \frac{1}{\bar{y}} \times \sqrt{\frac{1}{n}\sum_i (\hat{y}_i - y_i)^2} \times 100, \% \tag{6.2}$$

$$MAPE = \frac{1}{n}\sum_i \left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100, \% \tag{6.3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_i (\hat{y}_i - y_i)^2} \tag{6.4}$$

Table 6.1: Filters

Filter	Type	Rule
f_{ut}	User	user account is not a test one
f_{ue}	User	user account does not belong to an employee
f_{uv}	User	user account is associated with less than 10 visits a day
f_{in}	Item	item is not a published one
f_{im}	Item	item is not a marketing one
f_{in}	Item	item is not currently in a heavy marketing campaign

For the inital phase of project, MAE looks adequate, preferable over RMSE as the latter is more affected by large error values, and it is suspected there will be outliers due to the issues with data quality.

6.2 Technical Architecture

As seen from the above, the recommender system requires a technology infrastructure to run on it. Data are originated from several sources, there are millions of records to process and they are not persistent in the system (that is they are usually removed relatively short period of time) so it is required to combine all that data on one single data management platform.

Getting the data management platform will also allow covering the requirement that there should be data access layer for business analysts. Certainly, there will be required to design a data schema populated through ETL and data marts jobs to let them work with data on the level of business entities rather than on the level of technical records.

There were considerations and discussions as to selecting technology components. Finally, Cloudera Hadoop distribution was chosen, installed on a cluster and configured by the author for data processing. The more complete list of tools selected for the project is listed in Appendix B, the main criteria in choosing them was their popularity for data-oriented projects as well as them being Open Source[1] for the Open Source Definition) and GPL-compatible[2] in

[1] see https://opensource.org/

[2] https://en.wikipedia.org/wiki/GNU_General_Public_License

terms of their licencing i.e. essentially free for use.

The proposed technical architecture is shown in Figure 6.2.

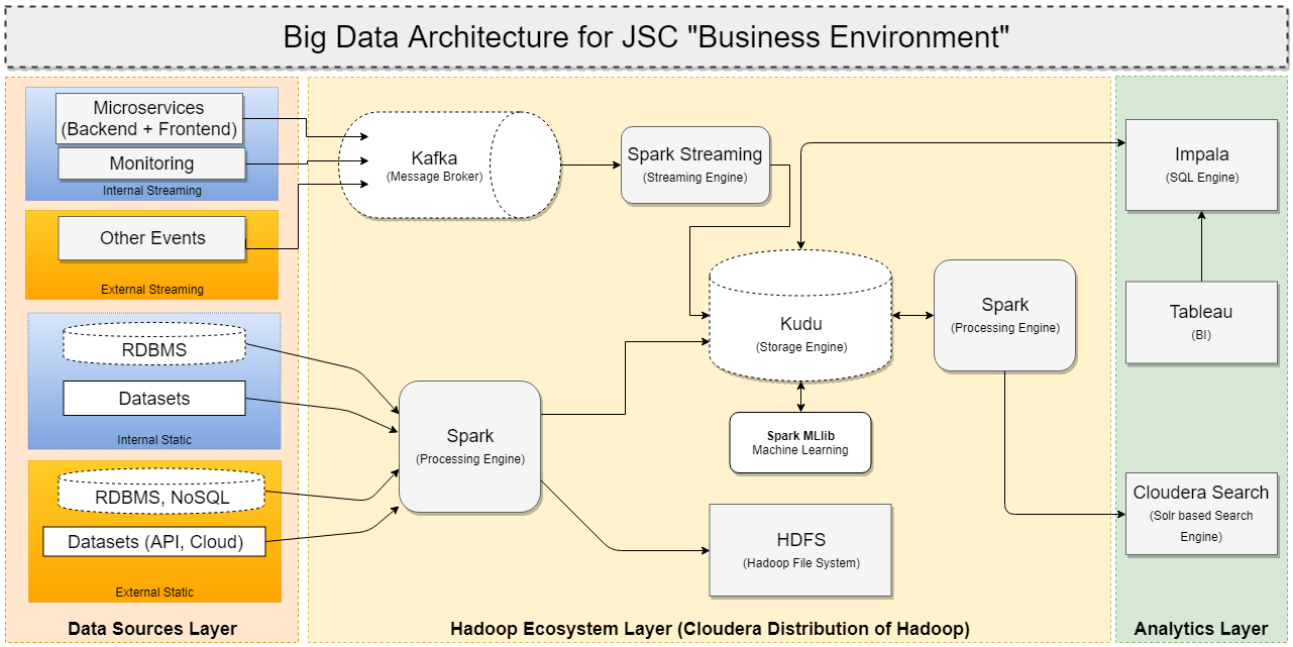

Figure 6.2: Technical Architecture

Chapter 7

Recommender System: Model

It's possible now to reason what type of recommender system should be built. Based on the review in chapter 2 and the EDA in chapter 5 there are suggested the following initial requirements to its design

- it should have ability to deal with the cold start,
- there is the explicit feedback data though it's rare, it should be used if possible,
- it should use the implicit feedback as the primary source of information about user interests,
- it should support sparse data,
- there is currently no strict requirement as to the computing difficult of model but, strategically, it should support near real-time computation of recommendations for an user,
- model's implementation should be able to scale horizontally.

The nature of available data on user-item interactions in the system is such that it needs to rely on the rating matrix concept to build a recommender system. The context-based and the knowledge-based approaches cannot be justified enough for to the project's needs at least in their puristic interpretations for two reasons: a) users do not provide any metadata information about their true interests so there is no information to match against items, b) items are not described detailed enough to be matched against users. So, it is needed to select an approach from the neighborhood-based collaborative filtering concept.

It is worth noting that the site navigation is as such that an user at every given time is meant to work with items within one *category* so recommendations for items should allow selecting them by categories to let the front-end process

them. This happens almost automatically because every item is associated with a category. Though it might be important in some parts of the model to take the category factor into account.

As there are many new users visiting the site first time and there are many already registered users which do not have any history of visits (but they may return) it needs to think of the cold start issue i.e. what would be means to provide recommendations to those users we know nothing about.

It is required to rely on the implicit feedback otherwise the rating matrix R is so sparse ($<0.3\%$) that it'd be hardly possible to generate valuable recommendations. For that reason, the implicit feedback which is essentially a number of clicks or the duration of time spent on an item page needs to converted into other numerical values so as the class of standard distance-based algorithms designed for the explicit feedback would apply. However, they are known to be slow and seemingly being phased out by more modern algorithms which support the implicit feedback by design, namely, by latent factor models.

The view is also supported by the performance requirements. Model-based algorithms are also beneficial in that sense they have parameters to regulate how large memory and cpu they may require. Most popular algorithms are supported by Spark Machine Learning Library[1] (Spark MLib) which support several node parallel computations and that can cover that requirement of horizontal scalability.

The chosen model for the recommendation system is a latent factor model.

7.1 Cold Start

For resolving the cold start issue a simple approach is suggested. For each item visited by users, the total number of visits is computed and subsequently normalized within a category on the five star scale. This allows ranking items according to their popularity *within* a category which is important for being compatible with the web site navigation schema.

To account for the decay of collective memory, the number of visits per items are weighted according to the decaying curve over an interval of one year. This interval was selected during discussion with business and mostly

[1] https://spark.apache.org/mllib/

justified by assumption that one year old materials are no longer of interest to most of users (on the condition they were not popular during that year).

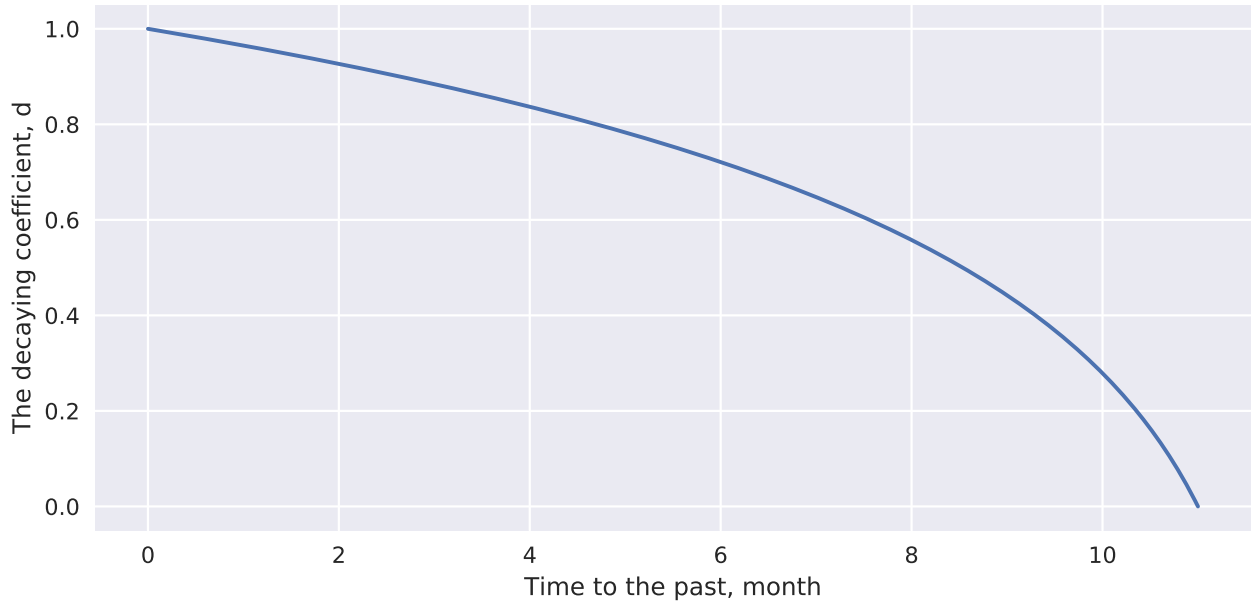

Figure 7.1: The ageing (decay) function

The weighting decay function is visualized in Figure 7.1, it is defined as

$$f(x) = \frac{\log(n - x)}{\log(n)}. \tag{7.1}$$

when n - the length of period (365 days) and x is a day which needs to weigh.

It was designed by the author but later it was found its shape is supporting by a recent work on decay of collective memory [69].

The result of the procedure is the matrix $G_{c,i}$ where for each category c there is a vector of items with their ratings and providing recommendations to unauthorized user is essentially selecting top n from G given c. These kind of recommendations can be called mass recommendations.

7.2 Rating Matrix

The explicit ratings are on the five star scale and that looks reasonable to compute recommendations on the same scale. Items tagged as favourites by

users are converted to the five star rating to help with decreasing the rating matrix sparsity.

The explicit rating matrix R sparsity is 0.0023, still it can possible be used.

7.3 Latent Factors Model

Latent factor models is probably the best available approach at the moment if little known of users and items except ratings, these models are thought to be state-of-the-art in recommender systems, they are based on matrix factorization techniques [39] and became famous following the Netflix prize contest. They are also attractive conceptually, they rely on an assumption there are some strong correlations within R which can be extracted as *factors* of much lower dimension. These factors are lately used to reconstruct a rating matrix of original dimension. This concept is probably supported by the real life as users of the site may be a large but quite consistent ecosystem influenced by same set of real-life factors.

This class of models reduce dimension of data which is beneficial in terms of infrastructure cost, and they potentially supports interpretation of its latent factors which can be useful for marketing purposes.

The core idea is matrix factorization. In terms of collaborative filtering, R is treated as a matrix of rank k and is factorized as the following product of two matrices of much lower dimensions:

$$R_{m\times n} \approx U_{m\times k} V^T_{n\times k} \tag{7.2}$$

Columns of U (or V) are perceived as latent components, and rows of U (or V) - as a latent factor. They loosely can be interpreted as ”interest dimension” of users.

There are multiple algorithms how to find U, V, the most popular ones in the field of recommender systems are Stochastic Gradient Descent (SGD) [14] and Alternating Least Squares (ALS) [33]. Historically, SVD was popular but it does not support regularization, and is known to be prone to overfitting where SGD and ALS are optimization algorithms with constrains and the

cost function. ALS is shown to be fast in [26], implemented[2] in the popular SparkML library. There are numerous descriptions of ALS so let me skip describing it here. What is important is that it is capable of finding U and V.

The matrix factorization algorithm was adapted to deal with implicit data by Hu, Korenand and Volinsky [33]. They suggested a way to view the implicit *feedback* f_{ui} as a composition of *preference* p_{ui} and *confidences* c_{ui}.

In that terminology, the preference is defined as binary representation of feedback f_{ui}:

$$p_{ui} = \begin{cases} 0, & \text{if } f_{ui} = 0, \\ 1, & \text{if } f_{ui} > 0. \end{cases} \tag{7.3}$$

The confidence c_{ui} is something more numerical taken from the implicit feedback with respect to the regularization parameter α:

$$c_{ui} = 1 + \alpha f_{ui} \tag{7.4}$$

The parameter α regulates

The feedback f_{ui} is naturally proportional to assumed interest of an user. Here, I suggest using a discretization function d for getting f_{ui} from the time duration of user keeping item page open, let's denote it as t_{ui}. The discretization function should take into account that small values of time duration are more on the negative side of rating than on the positive ones. Based on the

$$f_{ui} = \begin{cases} 0, & \text{if } t_{ui}\text{not specified}, \\ 1, & \text{if } 0 \leq t_{ui} \leq t_1, \\ 2, & \text{if } t_1 < t_{ui} \leq t_2, \\ 3, & \text{if } t_2 < t_{ui} \leq t_3, \\ 4, & \text{if } t_3 < t_{ui} \leq t_4, \\ 5, & \text{if } t_4 < t_{ui} < t_{max}. \end{cases} \tag{7.5}$$

[2] https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#module-pyspark.ml.recommendation

where t_i are the discretization bin boundaries, their exact values can be optimized over time, probably, even to respect of an user u and type of an item i.

The explicit ratings r_{ui} can be handled within the same function $d(t)$ directly as they are on the same scale e.g. $d(t)$ can take the value of r_{ui} and ignore t_{ui} completely or their can be a weighting scheme which makes sense if an user forgets updating their ratings but their true preferences are changing.

There is built a cost function on *preference* p_{ui} and *confidences* c_{ui} which is minimized by the ALS algorithm, thus, providing U, V.

The expected rating $\hat{r}_{ui}$ then is

$$\hat{r}_{ui} = U_i V^T. \tag{7.6}$$

7.4 Accuracy

The model was tested offline i.e. on the existing data using the cross-validation technique, the accuracy measured by MAE was within the interval 0.8-1.2. Further improvements are associated with the increase of user activity and more accurate tracking of their interactions.

Conclusion

The main objective was to start to design and integrate a recommender system for a real-life web site with educational materials. During the course of activities the field was studied and a complete end-to-end solution developed. I have used techniques from Statistics, Data Mining and Machine Learning to investigate into nature and structure of available data, there have been close collaboration with system administrations and developers for both the front-end and back-end components that support the web site.

The project has been treated as a multidisciplinary one, by the nature of its goal, demanding both technical (Computer Science) and analytical skills. That led to activities on several layers: from the infrastructure one, up to the business-centric one, namely:

- on the infrastructure level, used the Coursera Hadoop distribution, gained experience with Big Data technologies.
- on the storage level, designed database schema according to capabilities of Impala and Kudu.
- on the integration level, learnt how to work with external data sources (ElasticSearch, Solr),
- on the data processing level, improved my experience with Python data-centric libraries,
- on the application level, designed and developed the hybrid recommender system, created data layer for business analysts with SQL access to it.
- on the business level, improved my communication skills during a number of business meetings.

The recommender system certainly stands in the centre of project. Its design allows working with implicit ratings which is more complex than in the case of most traditional recommender systems as it does not expect users will necessarily provide ratings, that is one of promising directions of modern rec-

ommender systems in the field. This is especially important for this particular web site as user activity is weak on i, there is very limited number of explicit ratings and the recommender system should help to improve user experience with the site. hopefully to increase its popularity.

During the project, I was inspired with many ideas, some of them are directly related to it, the others are not but I'd like to mention some of them as they are currently finding their way into a roadmap for further development of the web site:

- improve data capturing methods, for that formulate a system of events which can describe all kind of user-item interactions, use them on the front-end and get away from the web log records in favour of events,
- provide an explanation to users why they are given certain recommendations with ability for them to react to those explanations,
- extract and analyse information of how different intervals of video are treated by users, finding less popular and more popular intervals, it may help editors to decide on quality of those videos,
- convert audio tracks of videos to texts (scripts) to have textual components for all items and add the context-based component to the recommender system (using NLP techniques),
- integrate with the site's search subsystem (built on Solr), use statistics on search queries as another source of information for the recommender system,
- formalize user activity as 'travelling' on some graph with nodes representing items, use the network theory then to find business meaningful patterns, probably, with the help of motifs and graphlets, those findings may help to improve the recommender quality significantly,
- tag recommendations with reference to conditions upon which they were derived, that creates possibility to conduct real-life A/B tests i.e. it can be considered as an research infrastructure in its basic form,
- integrate the recommender system with email marketing campaigns (conditioned on availability of relevant email distribution service).
- use cluster analysis on the rating data as an instrument for finding groups of users for better marketing campaigns.

The notable lessons learnt are

- Enabling automatic circulation of data in an IT system so as to build a data-driven solution is a non-trivial task, it is complicated by the factor of so-called data drift when meaning of data tends to be changed over time as an eco-system generating and maintaining it changes following external requirements.
- Frequently, available data is not rich and/or reliable enough to answer questions especially complex ones like "Find a particular user of group which is likely to stop using the site because of that new change" or "What are motivating factors for users?". Business does not always understand technical and methodological difficulties associated with finding objective answers to these questions. Working on that requires ensuring of data quality and adequate understanding of specifics of business which usually means close communications with two opposite worlds: Business and Technology.
- Working on a data-centric component raises many side questions. It might be difficult to keep a project on track, to not get distracted by trying to answer those questions.

To conclude, I have studied the field, new to me, on my own, by using my previous experience in Computer Science, Statistics and Machine Learning to design and develop a complete solution required for a real-life project, which is being implemented for its production usage.

Notation and Abbreviations

I	Set of items
U	Set of users
R	The rating matrix
r	rating
i,j	items
u,v	users
r_{ui}	The rating of the user u for the item i

EDA	Exploratory Data Analysis
MSE	Mean Squared Error
RS	Recommender System
SMB(s)	Small and medium-sized businesses

Bibliography

1. A Literature Review and Classification of Recommender Systems Research / D. H. Park [et al.] // Expert Systems with Applications. — 2012. — Sept. 1. — Vol. 39, no. 11. — P. 10059–10072. — URL: http://www.sciencedirect.com/science/article/pii/S0957417412002825.
2. A Survey of Machine Learning for Big Data Processing / J. Qiu [et al.] // 16876180. — 2016.
3. A Survey of Open Source Tools for Machine Learning with Big Data in the Hadoop Ecosystem / S. Landset [et al.] // Journal of Big Data. — 2015.
4. *Agarwal D. K.*, *Chen B.-C.* Statistical Methods for Recommender Systems. — New York, NY : Cambridge University Press, 2016. — 284 p.
5. *Aggarwal C. C.* An Introduction to Recommender Systems // Recommender Systems. — 2016.
6. *Aggarwal C. C.* Recommender Systems: The Textbook. — First edition. — Cham : Springer, 2016. — 498 p.
7. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems / Xin Luo [et al.] // IEEE Transactions on Industrial Informatics. — 2014. — May. — Vol. 10, no. 2. — P. 1273–1284. — URL: http://ieeexplore.ieee.org/document/6748996/.
8. *Anderson C.* The Long Tail: Why the Future of Business Is Selling Less of More. — Rev. and updated ed. — New York : Hyperion, 2008. — 267 p.
9. *Athmaja S.*, *Hanumanthappa M.*, *Kavitha V.* A Survey of Machine Learning Algorithms for Big Data Analytics // 2017 International Conference on Innovations in Information, Embedded and Communication Systems

(ICIIECS) (2017 4th International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)). — Coimbatore : IEEE, 03/2017. — P. 1–4. — URL: http://ieeexplore.ieee.org/document/8276028/.

10. *Bell R. M.*, *Koren Y.* Lessons from the Netflix Prize Challenge // ACM SIGKDD Explorations Newsletter. — 2007.

11. Benford's Law: Theory and Applications / ed. by S. J. Miller. — 1st ed. — Princeton University Press, 05/26/2015. — URL: http://princeton.universitypressscholarship.com/view/10.23943/princeton/9780691147611.001.0001/upso-9780691147611.

12. *Bennett J.*, *Lanning S.* The Netflix Prize // Proceedings of the KDD Cup Workshop 2007. — New York : ACM, 08/2007. — P. 3–6. — URL: http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf.

13. *Bornmann L.*, *Mutz R.* Growth Rates of Modern Science: A Bibliometric Analysis Based on the Number of Publications and Cited References: Growth Rates of Modern Science: A Bibliometric Analysis Based on the Number of Publications and Cited References // Journal of the Association for Information Science and Technology. — 2015. — Nov. — Vol. 66, no. 11. — P. 2215–2222. — URL: http://doi.wiley.com/10.1002/asi.23329.

14. *Bottou L.* On-Line Learning and Stochastic Approximations // On-Line Learning in Neural Networks / ed. by D. Saad. — Cambridge : Cambridge University Press, 1999. — P. 9–42. — URL: https://www.cambridge.org/core/product/identifier/CBO9780511569920A009/type/book_part.

15. *Burke R.* Hybrid Recommender Systems: Survey and Experiments // User Modeling and User-Adapted Interaction. — 2002. — Nov. — Vol. 12, no. 4. — P. 331–370. — URL: https://doi.org/10.1023/A:1021240730564.

16. *Burke R.* Hybrid Web Recommender Systems // The Adaptive Web. Vol. 4321 / ed. by P. Brusilovsky, A. Kobsa, W. Nejdl. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. — P. 377–408. — URL: http://link.springer.com/10.1007/978-3-540-72079-9_12.

17. *Calhoun Williams K. J.* Hype Cycle for Education. — 2017. — URL: https://www.gartner.com/en/documents/3769145.

18. *Castells P.*, *Hurley N. J.*, *Vargas S.* Novelty and Diversity in Recommender Systems // Recommender Systems Handbook, Second Edition. — 2015.

19. Challenging the Long Tail Recommendation / H. Yin [et al.] // Proceedings of the VLDB Endowment. — 2012. — May 1. — Vol. 5, no. 9. — P. 896–907. — URL: http://dl.acm.org/citation.cfm?doid=2311906.2311916.

20. Controlled Experiments on the Web: Survey and Practical Guide / R. Kohavi [et al.] // Data Mining and Knowledge Discovery. — 2009.

21. *Cremonesi P.*, *Koren Y.*, *Turrin R.* Performance of Recommender Algorithms on Top-n Recommendation Tasks // Proceedings of the Fourth ACM Conference on Recommender Systems - RecSys '10 (The Fourth ACM Conference). — Barcelona, Spain : ACM Press, 2010. — P. 39. — URL: http://portal.acm.org/citation.cfm?doid=1864708.1864721.

22. Data Mining Methods for Recommender Systems / X. Amatriain [et al.] // Recommender Systems Handbook. — 2011.

23. *De Solla Price D. J.* Networks of Scientific Papers // Science. — 1965. — July 30. — Vol. 149, no. 3683. — P. 510–515. — URL: http://www.sciencemag.org/cgi/doi/10.1126/science.149.3683.510.

24. Design of a Recommender System Based on Customer Preferences: A Comparison Between Two Approaches / I. Jomaa [et al.] // Volume 2: Applied Fluid Mechanics; Electromechanical Systems and Mechatronics; Advanced Energy Systems; Thermal Engineering; Human Factors and Cognitive Engineering (ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis). — Nantes, France : ASME, 07/02/2012. — P. 827. — URL: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?doi=10.1115/ESDA2012-82771.

25. *Deza M. M.*, *Deza E.* Encyclopedia of Distances. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2016. — URL: http://link.springer.com/10.1007/978-3-662-52844-0.

26. Efficient and Portable ALS Matrix Factorization for Recommender Systems / J. Chen [et al.] // 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)). — Orlando / Buena Vista, FL, USA : IEEE, 05/2017. — P. 409–418. — URL: http://ieeexplore.ieee.org/document/7965075/.

27. *Englehardt S.*, *Narayanan A.* Online Tracking: A 1-Million-Site Measurement and Analysis // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16 (The 2016 ACM SIGSAC Conference). — Vienna, Austria : ACM Press, 2016. — P. 1388–1401. — URL: http://dl.acm.org/citation.cfm?doid=2976749.2978313.

28. *Gedikli F.*, *Jannach D.*, *Ge M.* How Should i Explain? A Comparison of Different Explanation Types for Recommender Systems // International Journal of Human Computer Studies. — 2014.

29. *Golov N.*, *Rönnbäck L.* Big Data Normalization for Massively Parallel Processing Databases // Computer Standards & Interfaces. — 2017. — Nov. — Vol. 54. — P. 86–93. — URL: https://linkinghub.elsevier.com/retrieve/pii/S0920548917300363.

30. *Gomez-Uribe C. A.*, *Hunt N.* The Netflix Recommender System: Algorithms, Business Value, and Innovation // ACM Transactions on Management Information Systems. — 2015. — Dec. 28. — Vol. 6, no. 4. — P. 1–19. — URL: http://dl.acm.org/citation.cfm?doid=2869770.2843948.

31. Guide to Big Data Applications. Vol. 26 / ed. by S. Srinivasan. — Cham : Springer International Publishing, 2018. — (Studies in Big Data). — URL: http://link.springer.com/10.1007/978-3-319-53817-4.

32. *Gunawardana A.*, *Shani G.* Evaluating Recommender Systems // Recommender Systems Handbook, Second Edition. — 2015.

33. *Hu Y.*, *Koren Y.*, *Volinsky C.* Collaborative Filtering for Implicit Feedback Datasets // 2008 Eighth IEEE International Conference on Data Mining (2008 Eighth IEEE International Conference on Data Mining (ICDM)). — Pisa, Italy : IEEE, 12/2008. — P. 263–272. — URL: http://ieeexplore.ieee.org/document/4781121/.

34. Is Seeing Believing?: How Recommender System Interfaces Affect Users' Opinions / D. Cosley [et al.] // Proceedings of the Conference on Human Factors in Computing Systems - CHI '03 (The Conference). — Ft. Lauderdale, Florida, USA : ACM Press, 2003. — P. 585. — URL: http://portal.acm.org/citation.cfm?doid=642611.642713.

35. *Johnson R.*, *Watkinson A.*, *Mabe M.* The STM Report: An Overview of Scientific and Scholarly Publishing. — 2018. — URL: https://www.stm-assoc.org/2018_10_04_STM_Report_2018.pdf.

36. *Kimball R.*, *Ross M.* The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. — 3rd edition. — Indianapolis, Ind : Wiley, 2013. — 564 p. — URL: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/books/data-warehouse-dw-toolkit/ ; OCLC: 840431951.

37. *Knijnenburg B. P.*, *Willemsen M. C.* Evaluating Recommender Systems with User Experiments // Recommender Systems Handbook, Second Edition. — 2015.

38. *Konstan J. A.*, *Riedl J.* Recommender Systems: From Algorithms to User Experience // User Modeling and User-Adapted Interaction. — 2012. — Apr. — Vol. 22, no. 1/2. — P. 101–123. — URL: http://link.springer.com/10.1007/s11257-011-9112-x.

39. *Koren Y.*, *Bell R.*, *Volinsky C.* Matrix Factorization Techniques for Recommender Systems // Computer. — 2009. — eprint: 17255001.

40. *Kossovsky A. E.* Benford's Law: Theory, the General Law of Relative Quantities, and Forensic Fraud Detection Applications. — WORLD SCIENTIFIC, 10/2014. — URL: https://www.worldscientific.com/worldscibooks/10.1142/9089.

41. *Krulwich B.* Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data // AI Magazine. — 1997.

42. *Larsen P. O.*, *von Ins M.* The Rate of Growth in Scientific Publication and the Decline in Coverage Provided by Science Citation Index // Scientometrics. — 2010. — Sept. — Vol. 84, no. 3. — P. 575–603. — URL: http://link.springer.com/10.1007/s11192-010-0202-z.

43. *Linden G.*, *Smith B.*, *York J.* Amazon.Com Recommendations: Item-to-Item Collaborative Filtering // IEEE Internet Computing. — 2003.

44. *Mehta B.*, *Hofmann T.*, *Fankhauser P.* Lies and Propaganda: Detecting Spam Users in Collaborative Filtering // Proceedings of the 12th international conference on Intelligent user interfaces. — 2007.

45. *Middleton S. E.*, *de Roure D.*, *Shadbolt N. R.* Ontology-Based Recommender Systems // Handbook on Ontologies. — 2009.

46. MOOCs: A Systematic Study of the Published Literature 2008- 2012 / T. R. Liyanagunawardena [et al.] // International Review of Research in Open and Distance Learning. — 2013. — eprint: 12089095.

47. *Morgan G.* Market Guide for Higher Education Learning Management Systems. — 10/19/2017. — URL: https://www.gartner.com/en/documents/3817664.

48. *Mowery K.*, *Shacham H.* Pixel Perfect: Fingerprinting Canvas in HTML5 // Proceedings of W2SP 2012 / ed. by M. Fredrikson. — IEEE Computer Society. 05/2012.

49. *Murphy K. P.* Machine Learning: A Probabilistic Perspective. — Cambridge, MA : MIT Press, 2012. — 1067 p. — (Adaptive Computation and Machine Learning Series).

50. Orange: Data Mining Toolbox in Python / J. Deḿsar [et al.] // Journal of Machine Learning Research. — 2013.

51. Panorama of Recommender Systems to Support Learning / H. Drachsler [et al.] // Recommender Systems Handbook. — 2015.

52. *Park Y.-J.*, *Tuzhilin A.* The Long Tail of Recommender Systems and How to Leverage It // Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08 (The 2008 ACM Conference). — Lausanne, Switzerland : ACM Press, 2008. — P. 11. — URL: http://portal.acm.org/citation.cfm?doid=1454008.1454012.

53. *Pazzani M. J.* A Framework for Collaborative, Content-Based and Demographic Filtering // Artificial Intelligence Review. — 1999.

54. Privacy Aspects of Recommender Systems / A. Friedman [et al.] // Recommender Systems Handbook, Second Edition. — 2015.

55. *Provost F.*, *Fawcett T.* Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking. — 1st. — O'Reilly Media, Inc, 2013.

56. *Quinto B.* Next-Generation Big Data. — Berkeley, CA : Apress, 2018. — URL: http://link.springer.com/10.1007/978-1-4842-3147-0.

57. Recommender System Application Developments: A Survey / J. Lu [et al.] // Decision Support Systems. — 2015.

58. Recommender Systems for Learning / ed. by N. Manouselis. — New York : Springer, 2013. — 76 p. — (Springer Briefs in Electrical and Computer Engineering). — OCLC: 793571937.

59. Recommender Systems for Technology Enhanced Learning: Research Trends and Applications / N. Manouselis [et al.]. — 2014. — OCLC: 867615360.

60. Recommender Systems Handbook / ed. by F. Ricci [et al.]. — New York : Springer, 2011. — 842 p. — OCLC: 699734744.

61. Recommender Systems Handbook / ed. by F. Ricci, L. Rokach, B. Shapira. — Second edition. — New York Heidelberg Dordrecht London : Springer, 2015. — 1003 p. — OCLC: 935904837.

62. Recommender Systems Survey / J. Bobadilla [et al.] // Knowledge-Based Systems. — 2013. — eprint: 25246403.

63. Recommender Systems: An Introduction / D. Jannach [et al.]. — Cambridge : Cambridge University Press, 2010. — URL: http://ebooks.cambridge.org/ref/id/CBO9780511763113.

64. *Santos O. C.*, *Boticario J. G.* Modeling Recommendations for the Educational Domain // Procedia Computer Science. — 2010. — Vol. 1, no. 2. — P. 2793–2800. — URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050910003170.

65. Scalable Collaborative Filtering Approaches for Large Recommender Systems / G. Takács [et al.] // Journal of Machine Learning Research. — 2009. — eprint: 47676567.

66. *Schaffer J.*, *Höllerer T.*, *O'Donovan J.* Hypothetical Recommendation: A Study of Interactive Profile Manipulation Behavior for Recommender Systems // FLAIRS Conference. — 2015.

67. *Segaran T.* Programming Collective Intelligence: Building Smart Web 2.0 Applications. — 1st ed. — Beijing ; Sebastapol [CA] : O'Reilly, 2007. — 334 p. — OCLC: ocn166886837.

68. Shilling Attacks against Recommender Systems: A Comprehensive Survey / I. Gunes [et al.] // Artificial Intelligence Review. — 2012. — Vol. 42, no. 4. — P. 767–799.

69. The Universal Decay of Collective Memory and Attention / C. Candia [et al.] // Nature Human Behaviour. — 2019. — Jan. — Vol. 3, no. 1. — P. 82–91. — URL: http://www.nature.com/articles/s41562-018-0474-5.

70. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild / G. Acar [et al.] // Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14 (The 2014 ACM SIGSAC Conference). — Scottsdale, Arizona, USA : ACM Press, 2014. — P. 674–689. — URL: http://dl.acm.org/citation.cfm?doid=2660267.2660347.

71. *Toledo R. Y.*, *Mota Y. C.*, *Martínez L.* Correcting Noisy Ratings in Collaborative Recommender Systems // Knowledge-Based Systems. — 2015.

72. Toward Trustworthy Recommender Systems : An Analysis of Attack Models and Algorithm Robustness / B. Mobasher [et al.] // ACM Transactions on Internet Technology. — 2007.

73. *Veletsianos G.*, *Shepherdson P.* A Systematic Analysis and Synthesis of the Empirical MOOC Literature Published in 2013-2015 // International Review of Research in Open and Distance Learning. — 2016.

74. *Verma J. P.*, *Patel B.*, *Patel A.* Big Data Analysis: Recommendation System with Hadoop Framework // Proceedings - 2015 IEEE International Conference on Computational Intelligence and Communication Technology, CICT 2015. — 2015.

75. *Wang H.*, *Wang N.*, *Yeung D.-Y.* Collaborative Deep Learning for Recommender Systems // Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15 (The 21th ACM SIGKDD International Conference). — Sydney, NSW, Australia : ACM Press, 2015. — P. 1235–1244. — URL: http://dl.acm.org/citation.cfm?doid=2783258.2783273.

76. What Recommenders Recommend: An Analysis of Recommendation Biases and Possible Countermeasures / D. Jannach [et al.] // User Modeling and User-Adapted Interaction. — 2015. — Dec. — Vol. 25, no. 5. — P. 427–491. — URL: http://link.springer.com/10.1007/s11257-015-9165-3.

77. *Woźniak M.*, *Graña M.*, *Corchado E.* A Survey of Multiple Classifier Systems as Hybrid Systems // Information Fusion. — 2014. — Mar. — Vol. 16. — P. 3–17. — URL: https://linkinghub.elsevier.com/retrieve/pii/S156625351300047X.

78. *Zhu M.*, *Sari A.*, *Lee M. M.* A Systematic Review of Research Methods and Topics of the Empirical MOOC Literature (2014–2016) // Internet and Higher Education. — 2018. — eprint: 15974422.

79. *Верховская О. Р.* Глобальный Мониторинг Предпринимательства. Россия 2016/2017. — 2017.

80. *Кевеш А.* Малое и Среднее Предпринимательство в России 2017 (Статистический Сборник). — Росстат, 2017.

List of Figures

List of Tables

Appendix A

Data Model

The data schema of keeping records from various sources was designed according to the star schema principles.

Appendix B

Tools Used

Table B.1: Tools used

Type	Name
Architectual modelling	Archi, Microsoft Visio
Modelling	Python 3 with statsmodels, scikit-learn
Vizualization	Orange [50], Python 3 with matplotlib, seaborn
Processing	Python 3 with numpy, pandas
Integration	Kafka
Storage	PostgreSQL (keeps original data and interim results), Impala, Kudu
Infrastructre	Linux, Cloudera Enterprise 5.16 (the documentation)