Information Retrieval & Recommender Systems

Alejandro Bellogín and Alan Said

Abstract This chapter provides a brief introduction to two of the most common applications of data science methods in e-commerce: information retrieval and recommender systems. First, a brief overview of the systems is presented followed by details on some of the most commonly applied models used for these systems and how these systems are evaluated. The chapter ends with an overview of some of the application areas in which information retrieval and recommender systems are typically developed.

1 Introduction

Information retrieval is the process of retrieving information relevant to a query from an information source, e.g. a book from a library based on a title, or a relevant search result based on a query posted to a web search engine. Recommender systems, closely related to information retrieval systems, however work *without* a query. Instead, the recommender system attempts to identify the most relevant piece of information solely based on an implicitly expressed information need and intent – i.e., the user profile.

In the Information Retrieval (IR) community, information retrieval often means *text retrieval* by default, either intentionally or unintentionally. This might be due to historical reasons [39] or simply because text retrieval has been the most predominant information retrieval application. But, nonetheless, there exist many other forms of information retrieval applications. A typical example is collaborative filtering, which aims at finding information items a target user is likely to like by taking into account other users' preferences or tastes. Unlike text retrieval, collaborative

Alejandro Bellogín
Iniversidad Autńoma de Madrid, Spain e-mail: alejandro.bellogin@uam.es

Alan Said
University of Skövde, Sweden e-mail: alansaid@acm.org

filtering (CF) does not necessarily need textual descriptions of information items and user needs. It makes personalized recommendations by aggregating the opinions and preferences of previous users. Originally, the idea of collaborative filtering was derived from heuristics and coined by Goldberg and colleagues when developing an automatic filtering system for electronic mail [17]. Due to the uniqueness of the problem, it has been modeled and studied differently since then, mainly drawing from the preference prediction and machine learning view point [6].

In this chapter, we introduce the main concepts and models from Information Retrieval and Recommender Systems, paying special attention to one problem where there are several connections between these two areas: evaluation. We also present some of the most important applications for each community.

2 Models

2.1 Information Retrieval Models

Text retrieval techniques have been widely used in many different areas such as web retrieval [2], image and video retrieval [40], or content-based recommendation [41]. Among many techniques, one of the most commonly used are the term weighting functions, from which several extensions and probabilistic adaptations have been proposed throughout the years; the most representative methods are reviewed in the next sections.

2.1.1 Classical retrieval models: Boolean, TFIDF, vector models

The Boolean retrieval model was used in the first search engines and is still in use today [13]. Documents are retrieved if they exactly match the query terms, however, this does not generate a ranking, since this model assumes that all documents in the retrieved set are equivalent in terms of relevance. Nonetheless, this simple model has some advantages: it is very predictable and easy to explain to users; from an implementation point of view, it is also very efficient because documents can be directly removed from the retrieved set if they do not include any query features.

The major drawback of this approach, however, is the lack of a proper ranking algorithm. Because of this, the vector space model (VSM) was proposed [37], where documents and queries are assumed to be part of the term space. In such space, term weighting functions and similarities between documents can be defined. Usually, the terms are weighted according to their frequency in the document (TF, or term frequency) or normalized with respect to the number of documents where they appear (IDF, or inverse document frequency). Cosine similarity is typically used to compute the similarity between the document and query vectors, generated based on a specific term weighting function. This is used because, even though there is no

explicit definition of relevance in the VSM, there is an implicit assumption that relevant documents are located closer in the term space to the query – i.e., the distance between their corresponding vectors should be small.

2.1.2 Probabilistic models: probabilistic model, BM25, language models

Probabilistic models, in contrast with the previously described retrieval models, provide some levels of theoretical guarantees that the models will perform well, as long as the model assumptions are consistent with the observed data. The Probability Ranking Principle or PRP [34] states that ranking the documents by probability of relevance will maximize the precision at any given rank – assuming that the relevance of a document to a query is independent of other documents. Since the PRP does not tell us how to calculate or estimate such probability of relevance, each different probabilistic model proposes a different method for that which, at the end, could produce better or worse documents for a given query.

One of the first methods to calculate the probability of relevance using document terms was proposed in [35]. Other approaches estimate text statistics in the documents and queries and then build up the term weighting function through them [3]. The BM25 ranking algorithm extends the scoring function to include document and query term weights [33]:

$$\sum_{t_i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i} \quad (1)$$

where f_i is the frequency of term t_i in the document, qf_i is the frequency of the term t_i in the query, n_i (r_i) is the number of (relevant) documents that contain term t_i, N (R) is the number of (relevant) documents in the collection, and k_1, k_2, K are free parameters. While this extension is based on probabilistic arguments and experimental validation, it is not a formal model. Nonetheless, it performs very well in many different tasks.

More sophisticated approaches like the Relevance-Based Language Models (or Relevance Models for short, RM) are among the best-performing ranking techniques in text retrieval [22]. They were devised with the aim of explicitly introducing the concept of relevance, intrinsic to the probabilistic model, in statistical Language Models (LM) [30]. In common IR settings, the exact and complete set of relevant documents is generally unknown; relevance feedback techniques work with approximations to this set, which can be obtained by a wide variety of approaches, such as asking the user (explicit relevance feedback), or just taking the initial output of a well-performing IR system as a good guess (pseudo relevance feedback). Given a query and such an approximation to the set of relevant documents, RM selects good expansion terms from those present in the pseudo-relevant documents in order to formulate and run a better query.

2.1.3 Models for the Web: PageRank and Learning to rank

Ranking documents in the Web is substantially different from ranking a static collection, since other aspects besides the relevance to a query should be considered: spam should be avoided, pages with higher quality should be presented before (authorities), new content is created every day (hence, the models should be efficient so they are executed fast and able to manage huge amounts of data), and so on [5].

One of the most important signals available in the Web is the links between Web pages. The link structure of the Web allows to classify the pages into authorities, hubs, sinks, and normal pages by simply checking whether they are linked by many pages (authority), link to important pages (hubs), do not have further links (sinks), and the rest. PageRank and HITS algorithms exploit these ideas and compute a score for every page in the Web graph, by simulating a random surfer that navigates the Web [7, 20].

Another important signal that is typically exploited by search engines is the information about which documents are being clicked for each query. Once enough information is collected (click logs), standard machine learning algorithms can be used to learn a ranking based on some training data. The loss function to minimize is either the number of mistakes done by the algorithm or the (negative) average precision, see [12, 24] for a more comprehensive overview of these techniques.

2.2 Recommender System Models

Recommender Systems (RS) are ubiquitously used on the World Wide Web and are increasingly being put to use in more application scenarios. Even though recommender systems have grown to prominence in the *online* world, the techniques for many of today's systems were developed prior to the Internet breakthrough. Traditionally, recommender systems have been closely linked with information retrieval systems as they use similar models to identify relevant data. However, in recent years, with the rapid developments of recommender systems, the two fields have come to grow more separate.

In general, recommender systems approaches are categorized as *collaborative filtering*-based, *content*-based, or as *hybrid* models which blend both collaborative and content information into one unified approach.

2.2.1 Collaborative Filtering Recommender Systems

Collaborative Filtering (CF) was initially developed as a means of filtering out relevant news posts on Usenet [32]. The intuition behind the initial *user-based* CF algorithm is that people with similar preferences to each other have not all seen the same items. Thus, by looking at the items that users similar to oneself have seen, we can find items which should be of interest to us.

The traditional user-based collaborative filtering method uses the k-nearest neighbor algorithm in order to find the most suitable recommendations by predicting rating scores that will be given to items by users, i.e.

$$\tilde{r}_{ui} \propto \sum_{v \in N_k(u)} \text{sim}(u,v) r(v,i) \tag{2}$$

where r_{ui} refers to the predicted rating that user u will assign to item i, $N_k(u)$ is the set of the k most similar users (nearest neighbors) to user u, and $sim(u,v)$ the similarity between user u and neighbor v. The similarity method $sim(u,v)$ can be chosen freely. Common methods are *cosine similarity* or *Pearson correlation*.

Cosine similarity measures the cosine angle between two users where each user is represented by a vector of their ratings, i.e.

$$cos(u,v) = \frac{u \cdot v}{||u||||v||} = \frac{\sum_{i=1}^{|I|} u_i v_i}{\sqrt{\sum_i {u_i}^2} \sqrt{\sum_i {v_i}^2}} \tag{3}$$

where u and v are the two user respectively, and u_i and v_i the ratings to item i. Similarly, the Pearson correlation measures the linear correlation between two vectors, where each vector represents the ratings of a user, i.e.

$$P(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{4}$$

where u and v are two, I_{uv} the intersection of user u's and v's rated items, and r_{ui} and r_{vi} the rating given to item i by user u and user v respectively.

Analog to user-based collaborative filtering, *item-based* collaborative filtering instead looks at similarities between items. Instead of identifying similarities between users, this method attempts to find the most similar items to those the candidate user has already rated (or otherwise interacted with).

User-based and item-based collaborative filtering belong to the *memory-based* family of recommendation approaches. As the name implies, memory-based approaches keep the entire database of user-item interactions in memory in order to find the most appropriate recommendations. *Model-based* recommendation approaches instead create a recommendation model from the available data, when providing recommendations to users, recommendations can be accessed from the precomputed model. There is a wide variety of memory-based recommendation models, some of the most commonly ones fall into the category known as *matrix factorization*, e.g. Bayesian Personalized Ranking [31], SVD++ [21]. Somewhat related, *probabilistic topic modeling* approaches like Latent Dirichlet Annotation [4] also fall in the model-based category.

2.2.2 Content-based Recommender Systems

Content-based recommender systems (CB) analyze the items the user liked in the past and recommend items having similar features. The main process of making recommendations using a content-based approach consists in matching users preferences and interests obtained in the users' record, with the attributes of the items [25]. Since CB systems recommend items analyzing the user profile, they aim to maximize the similarity between the user profile and the content of the item, that is, $sim(UserProfile(u), Content(i))$, where $Content(i)$ will be the item profile (attributes that characterize item i).

As CB recommenders tend to use articles represented by text, the content is normally represented by keywords using simple retrieval models such as the ones presented before, the most representative example being the Vector Space Model, where cosine similarity with weights computed using either TF or TFIDF are the most popular techniques.

Nonetheless, there are other techniques for CB recommendation. One of the most important techniques is Bayesian classifiers. These approaches estimate the posterior probability $P(c \mid d)$ of a document belonging to a specific class c based on the prior probability of the class $P(c)$, the probability of observing the document in class c (i.e., $P(d \mid c)$) and the probability of observing the document d denoted as $P(d)$ [16]. The estimation of $P(d \mid c)$ is complicated, thus it is common to use the naïve Bayes classifier, as shown in [27] for book recommendation and in [28] for classifying unrated web pages. With the naïve Bayes classifier, the document is replaced by a vector of keywords over the system vocabulary. Each component of the vector indicates whether that keyword appeared in the document or not. If we work with binary values, we are using a multivariate Bernoulli approach and if we count how many times the word appeared in the document, we are making use of multinomial naïve Bayes [16].

2.2.3 Hybrid Recommender Systems

Hybrid recommendation takes advantage of the techniques from two or more recommender systems to achieve a higher performance while limiting the potential drawbacks that each system may obtain separately. Although there are hybrids that combine implementations of recommendations of the same type (for example, two content-based techniques), the most interesting ones are those that are able to work with different recommenders. There are different strategies for hybrid recommendation which [9] summarized in the seven methods shown in Table 1.

Table 1 Hybrid recommendation methods from [9].

Hybrid technique	Description
Weighted	Each recommender obtains a score for each candidate item and these scores are combined using a linear formula.
Switching	It switches between recommenders depending on the situation.
Mixed	Each recommender makes its own recommendations and the final output is a combination of them.
Feature combination	Features derived from various sources are combined and sent to the recommendation scheme.
Feature augmentation	Similar to feature combination but instead of deriving, the recommenders augment (compute) new features and send them to the final recommendation scheme.
Cascade	They normally use a weak and a strong recommender. The weak recommender is only used when breaking ties in the ranking.
Meta-Level	A recommender produces a model, which is the input for the second recommender. Similar to feature augmentation but the second recommender does not work with raw data, only with the model provided by the first recommender.

3 Evaluation

The evaluation of IR and RS is seemingly similar, but has stark conceptual differences. IR evaluation measures how well a system is able to retrieve relevant results to a certain query, whereas evaluation of RS looks into how relevant each recommended item is to the user it gets recommended to. Evaluation of recommender systems has, to a large extent, been based on information retrieval concepts such as precision, recall, F-measure, etc. These measures represent some form of quality of the system, e.g. the higher the precision and/or recall value is, the more accurate the system is. However, even though information retrieval systems and recommender systems are similar both in their use and implementation, there is a distinct contextual difference; whereas retrieving a known (but sought for) item is positive, recommending a known item has far lower utility [19].

There exist a few key concepts in evaluation of both information retrieval and recommender systems that are as relevant to both. One of these, and very likely the most important aspect of both information retrieval and recommendation results, is the *relevance* of the information presented to the end user. Any piece of information retrieved or recommended to the user should exhibit a certain relevance to the users information need. In general, evaluation of both type of systems focuses on measuring the relevance (through some metric) of the items presented to the user.

In order to measure the relevance of items, we need to know the *ground truth*, i.e. we need to know which items are relevant to which users. It is using this ground truth that we can gauge how well a recommender or retrieval system performs. The ground truth is specified by a dataset, for recommendation purposes, this often consists of historical interactions between items and users, for information retrieval it is often found in a *document collection* which consists of e.g. a set of documents, search queries used on the set of documents, and relevance assessments which tell how relevant a document is to each query.

3.1 Metrics

Recommendation qualities are commonly expressed through a number of metrics and methods. The choice of these is often based on the type of dataset used in the system, the use case, expected outcome, etc. Arguably the most common metrics in both recommender systems and information retrieval is the precision (P) and recall (R) pair [19]. These metrics are usually applied in offline training/test scenarios, where algorithms are trained using a portion of the available data and then evaluated by comparing predictions to a withheld portion of the data, i.e. *true positive* recommendation.

Precision is the fraction of relevant retrieved documents. In a recommender system evaluation setting it corresponds to the true positive fraction of recommended items. Recall is the fraction of all relevant items which are retrieved. The formula for calculating precision is shown in Equation 5 while recall is shown in Equation 6. In both equations, *relevant* refers to the complete set of relevant items, and *retrieved* refers to the complete set of retrieved items.

$$P = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|} \tag{5}$$

$$R = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|} \tag{6}$$

Commonly, precision is expressed as precision at k where k is the length of the list of recommended items, e.g. $P@1 = 1$ would indicate that one item was recommended, and the item was deemed to be a true positive recommendation, $P@2 = 0.5$ would indicate that two items were recommended and one them was deemed a true positive, etc.

Variants of precision used for recommender evaluation include *Average Precision* (AP) and *Mean Average Precision* (MAP). Both these metrics are used when more than one item is recommended. They extend the precision metric by taking into consideration the position of true positive recommendations in a list of recommended items, i.e. the position k in a list of n recommended items in Equation 7. $rel(k;q)$ is a binary classifier taking the value 1 if the item at position k is relevant for query q and 0 otherwise. Mean Average Precision additionally averages the scores

at each query (or user, in recommendation), as shown in Equation 8.

$$AP(q) = \frac{\sum_{k=1}^{n} (P@k(q) \times rel(k;q))}{|\{relevant\}|} \tag{7}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{8}$$

Other common metrics, used in the context of rating prediction, are the Root-Mean-Square Error (RMSE) and normalized Discounted Cumulative Gain (nDCG). In contrast to precision-based metrics, RMSE attempts to estimate the recommendation algorithm's rating prediction error, e.g. by comparing predicted ratings to actual ratings. The lower the error, the better the algorithm performs. RMSE is calculated as shown in Equation 9, where X, Y are two rating vectors (e.g. predicted item ratings vs. actual item ratings), where each position in the vector corresponds to the rating of a specific movie and n the size of the intersection of nonzero elements in both vectors.

$$RMSE(X,Y) = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}} \tag{9}$$

nDCG and DCG on the other hand measures the usefulness (relevance) of a document based on its position in the list of recommended items. In a rating scenario, this corresponds to how high the predicted ratings of the top-k items are, the formula is shown in Equation 10 where the *gain* (the predicted rating) of each item i for each user u in a list of J items is represented by g_{ui_j}. nDCG is the DCG over the true DCG, i.e. *ideal* DCG (IDCG) – the actual ratings, as shown in Equation 11.

$$DCG = \frac{1}{N} \sum_{u=1}^{N} \sum_{j=1}^{J} \frac{g_{ui_j}}{\max(1, \log_2 j)} \tag{10}$$

$$nDCG = \frac{DCG}{IDCG} \tag{11}$$

3.2 Collections

As briefly noted in the beginning of this section, document and data collections are a necessity in order to be able to evaluate information retrieval and recommender systems. Even though there are similarities between the collections used for information retrieval and for recommender systems, there are clear differences. Below, we present document collections and historical interaction datasets used for search and recommendation respectively.

3.2.1 Document Collections

To ensure fair experimental comparisons and repeatable experimental settings, the data used in such experiments should be fixed. In information retrieval, test collections are assembled consisting of documents, queries, and relevance judgements (for some query-document pairs). Differently to other areas such as Machine Learning, in IR, the queries and the relevance judgments are gathered specifically for a particular search task, at the same time as the documents [13].

In this context, creating relevance judgments require a considerable investment of manual effort. When collections were very small, most of the collections could be evaluated for relevance, hence, the relevance judgments for each query were relatively exhaustive. In larger collections, a technique called *pooling* is used, where the top k results from different rankings are merged into a pool, duplicates are removed, and these documents are then presented to the assessors or judges to produce the relevance judgments. This technique is based on two assumptions: most of the documents in the pool are relevant (because they were retrieved high in the ranking by different methods) and those documents not in the pool can be considered not relevant. Both assumptions have been verified to be accurate for some specific IR tasks [5], however, whenever a new retrieval algorithm (that did not contribute to the pool) is evaluated in this way, the effectiveness of this method could be underestimated if it mostly retrieves documents not in the original pool.

Following this evaluation paradigm, information retrieval researchers have built many test collections, ranging from thousands to millions of documents. The Text Retrieval Conferences (TREC) have been the main events around which IR experiments have been designed, usually involving the creation of queries and relevance judgments under a specific track or task (ad hoc, filtering, high precision, diversity, etc.). Other important document collections have been generated in the context of INEX (Initiative for the Evaluation of XML Retrieval), the NTCIR project, and CLEF (Workshop on Cross-Language IR and Evaluation), oriented respectively to XML retrieval, Japanese and cross-lingual retrieval, and different multilingual tasks.

3.2.2 Interaction Datasets

When evaluating recommender systems, traditionally, the underlying data that is used contains user-item interactions or relations, these relations can be e.g. a rating given to an item by a user, a record of a user purchasing a product, a record of a user clicking on an item in a list of recommended items, the number of times a user has played a specific song, or how many minutes of a video a specific user played, etc. This type of data is commonly referred to as a user-item matrix U where the rows and columns correspond to users and items respectively. Each cell in the matrix holds the interaction information between the user and the item, i.e. in a movie recommendation scenario each cell will contain either the rating the user has given to the movie or be empty if the user has not rated the movie, in an e-commerce

interaction matrix each cell will contain e.g. whether or not the user has purchased the item.

Table 2 shows an example of a user-item matrix where each cell contains either a rating given to an item by a user, or is empty in the cases where the user has no interaction history with the item – these cases represent the target user-item pairs to be predicted by our recommender system. Note an important difference with respect to how the document collections are built as described before: whereas the relevance of a document with respect to a query can be assessed by any judge (since relevance is usually assumed to be an inherent property of the document, and, hence, it is to some extent universal), in recommendation this assessment is personal, and cannot be inferred by a different user. This leads to a much sparse scenario, where typical values for the density of this matrix (ratio between the number of known cells with respect to the total number of cells) is below 6% [18].

	u_1	u_2	u_3	u_4	u_5
i_1	1	3			5
i_2	1			3	5
i_3	2	2			4
i_4	3	4			1
i_5		4	1	5	
i_6		3		5	

Table 2 A basic example of a user-item matrix where each cell in the matrix corresponds to the rating given by the users to items.

4 Applications

Information retrieval and recommender systems have many application domains and are frequently becoming increasingly more intertwined. Below follows an overview of some of the traditional and common application areas of both types of systems.

4.1 Information retrieval

In the Information Retrieval community, information retrieval often means text retrieval by default, either intentionally or unintentionally. This might be due to historical reasons [39] or simply because text retrieval has been the most predominant information retrieval application. But, nonetheless, there exist many other forms of information retrieval applications. In this section we briefly review some of the most important or well-known applications that fall under the umbrella of Information Retrieval [5].

Web search is today the most important application of IR and its techniques [5]. According to [5], there are at least five issues that have impacted how IR methods, technologies, and processes have adapted to the Web: a) characteristics of the document collection (large, unstructured, connected through hyperlinks, distributed, it has to be collected or *crawled*), b) size and volume of submitted queries, c) very large document collection that makes relevance prediction much harder (also because there are many noisy documents), d) new search tasks and user needs have emerged, together with structured data in both queries and documents, e) the increasingly pervasive presence of spam on the Web due to new incentives such as advertising and electronic commerce content.

Relevance feedback is the most popular query reformulation strategy. In a relevance feedback cycle, the user is presented with a list of the retrieved documents which are later marked as relevant or not relevant. The main idea consists of selecting important terms from the selected documents and enhance or decrease the importance of these terms in a new query formulation, together with expanding the query with new terms. When using a vector space model, this technique is simply defined as a term reweighting strategy, where different formulations exist to calculate the modified query depending on assumptions on the user behavior regarding the feedback cycle [5, 36].

Before the Web, the design of search tools was targeted to help people write good queries, implying the query language being adopted was usually complex. However, nowadays search engines are used not only to find information but to achieve other goals [5]. The first categorization of queries was done in [8] into three classes: informational, navigational (finding a Web site for browsing), and transactional (interactive tasks such as buying goods or downloading a file). This taxonomy was later refined; moreover, now the focus is on automatically predicting the **query intent**, so that different query attributes are analyzed and selected to derive which of them may be linked to each intent. However, ambiguous queries present a scenario where the user intent is harder to be predicted correctly, as expected. In this context, several proposals have been studied to measure the query difficulty, closely related to its ambiguity. More specifically, the *clarity score* measures how closely related are the documents returned for a query with respect to a particular document collection, which aims to measure the ambiguity of a query towards a collection [11].

Text classification corresponds to a broad problem – mostly addressed by Machine Learning researchers – where a collection of documents is assigned one (or more) class/label out of a predefined number of classes/labels. A particular, and one of the most important, variant of this problem is the topic classification task, where each class describes a topic referred to in the documents. There have been proposed several algorithms to address this problem, either for the multi-label or the single-label scenario – the latter is acknowledged to be harder, because it is also necessary to decide which class is the best one to a given document. Regarding the algorithms, both supervised (decision trees, nearest neighbors, naive Bayes, SVMs) and unsupervised (clustering, direct match) techniques can be applied, although in general supervised algorithms achieve better results, at the expense of requiring available training data.

Text compression allows to represent the text in fewer bits or bytes. Compression methods create a reduced representation, that can be later used to reconstruct the original text, by identifying and using patterns that exist in the text. These techniques help reducing costs associated with space requirements, input/output overhead, and communication delays, since compressed text requires less storage space, takes less time to be transmitted over a communication link, and takes less time to search directly the compressed text. All these advantages come at the expense of more time needed to code and decode the text. One of the most important challenges that appear when text compression techniques are applied in IR systems is the necessity of this type of systems to access text randomly, since most compression methods need to decode the entire text to find a match.

Rank fusion or rank aggregation is needed when you want to combine various result lists from different sources into one list, with no knowledge neither of the process followed by each source to produce those lists nor the data that has been used or the score rank for each element. Examples where rank fusion takes place include, for instance, metasearch, personalized retrieval (combine personalized results with query-based results), multi-criteria retrieval, etc. [38]

Enterprise search refers to the application of information retrieval technologies to information finding within organizations; in particular, of digital documents owned by the organization, such as their external Web site, the company intranet, and any other electronic text (email, database records, reports, shared documents, etc.). According to [5], a far-from-complete list of search-supported tasks that can be found in an enterprise is the following: approving an employee travel request, responding to calls in a call center, responding in the course of a dispute, writing a proposal, obtaining and defending patents, selling to an existing customer, expertise finding, and operating an E-commerce site.

Indexing is the process of creating the data structures needed to enable fast searching. Index creation must be efficient, both in terms of time and space, where usually a tradeoff must be met. Furthermore, indexes must also be able to be efficiently updated when new documents are found. Inverted files are the most common – and best choice for most applications – form of index used by search engines. They contain a list for every index term of the documents that contain such index term. Other techniques not so popular nowadays are suffix arrays and signature files.

Web crawling allows to automatically download Web pages, by means of programs usually called web *crawlers*, *spiders*, or *bots*. The order in which the URLs are traversed in the Web (by following the hyperlinks available in the pages) is important. In general, it is advised to do it using a breadth first strategy, since this is linked to finding *good* sites (i.e., those with higher Pagerank values) sooner in the process; however, depending on the final application and goal of the crawling, other preferences might be considered [23]. One of the main challenges on this topic is how to process efficiently as many pages as possible while preserving resource policies (collectively referred as crawler etiquette, e.g., comply with the Robot Exclusion Protocol) and ethical considerations.

XML (structured) retrieval supports querying and manipulating XML data by using languages that describe the hierarchical structure of XML data instead of

simpler models, enough to work with relational databases and unstructured documents [13]. Hence, the focus of these methods is on exploiting the document structure, thanks to the definition of a complex query language, XQuery. It is worth mentioning that the INEX[1] project has allowed to study the extent to which structure is useful in queries, by defining several tasks and building test collections for further evaluation, in a similar way as TREC did for Web search.

Multimedia retrieval is widely recognized as one of the most promising fields in the area of information management [5]. The most important characteristic of this type of systems is the variety of data it must be able to support, such as text, images (still and moving), graphs, and sound. For this reason, the corresponding data model, query language, and access and storage mechanisms should support objects with a very complex structure.

4.2 Recommender Systems

In the recommender systems community, recommendation was traditionally separated into two distinct use-cases, top-N recommendation and rating prediction. Top-n recommendation, as the name suggests focuses on generating a list of the N most relevant items to recommend to a specific user, whereas rating prediction focuses on predicting the rating a user will give to a certain item. The rating prediction task was very extensively researched in the past, this was in part due to the Netflix Prize[2] which focused on rating prediction and took place between 2006 and 2009. Today, various top-N aspects of recommendation are at the focus of the recommender systems research and practitioner communities.

Movie Recommendation was the primary application of RS systems previously. In part due to the Netflix Prize mentioned above, but also due to the availability of open research datasets such as Movielens [18] and, more recently, MovieTweetings [14]. Traditionally, movie recommendation focused on rating prediction but has gradually shifted to a top-N recommendation use case based on implicit data and models. The shift occurred as movie recommendation become increasingly more applied in streamed video use cases.

Music Recommendation is a common application scenario for recommendation. Similar to multimedia retrieval, a music recommender system must be able to support not only collaborative data (the user-item interactions) but also information such as timbre, tempo, genre, etc. making the data and recommendation models more complex. Music recommendation traditionally focuses on one of two tasks: recommendation of a single song or artist, where the recommendation is seen as a standalone action; and playlist recommendation where recommendations should meet requirements posed by the playlist that is being extended and the user who is listening.

[1] INitiative for the Evaluation of XML Retrieval, http://inex.mmci.uni-saarland.de

[2] http://www.netflixprize.com

User Recommendation has become very common with the rise of social media. Recommending acquaintances, colleagues, potential romantic partners, and similar applications differ from traditional recommendation of items in one very significant respect, namely reciprocity. When recommending a social connection to a user, the recommendation should be of value for both of the involved parties [29]. This requirement, although not unique for user recommendation, is a key factor of importance in this setting.

Context-aware Recommendation attempts to tailor recommendations for the user's current situation, or context, taking into consideration aspects such as the relevance variability of certain items in certain situations. Consider, for example, an e-commerce recommender system recommending clothing. When recommending clothing in summer, the recommendation could take into consideration the outdoor temperate, and similarly so in winter. The methods for using context in recommendation are generally considered to be either based on contextual filtering or contextual modeling [1], filtering referring to the concept of applying a filter to the list of recommended items to only show the items valid in the specific context, whereas modeling refers to when the recommendation algorithm takes the context into consideration when identifying items to recommend.

Explanations of Recommendations tell the user why a certain item is recommended. Recommendation explanation can be generated in various ways, commonly, the explanations are high-level enough to be understood by users who have no insight into how recommendation algorithms work. A simple example could be, e.g. telling the user that an item is recommended because "people similar to you liked this item". Similarity in this case could be based on any number of factors, commonly this can refer to e.g. the collaborative similarity presented in Section 2.2 of this chapter.

Recommender Systems in Education is the application of recommendations in order to enhance learning by supporting students through tailored educational materials and exercises according to their learning preferences, knowledge levels, goals, etc [26]. Recommendation models in this application space must adhere to learning-oriented requirements specified by teachers and educators, e.g. learning materials, expected attained knowledge, and course curricula among others.

Recommendations for Groups specifically attempt to tailor recommendation for a group of people at the same time, instead of the regular approach where personalized recommendations are delivered to a single user. Approaches for group recommendations take into consideration issues regarding the group satisfaction with a certain recommendation, e.g. whether it is, say, better to recommend very suitable items for the majority of the group at the cost of leaving certain group members unhappy (maximize happiness), or whether it is better to recommend items that the whole group will be comfortable with, albeit not specifically happy (minimize discontent) [15].

Cross-domain Recommendations capture the preferences users express about items in one domain, e.g. movies, and use these preferences to identify relevant recommendations in a separate domain, e.g. music. Given that items, interactions, and preferences can be expressed differently across different domains, cross-domain rec-

ommender systems attempt to leverage and transfer the knowledge in other domains and apply on the current one [10]. A well-known example of cross-domain recommendation is e-commerce, e.g. recommendations at Amazon or eBay are often sets of items from various domains, e.g. books, electronics, clothing, etc.

5 Summary

This chapter has presented a brief overview of Information Retrieval and Recommender Systems, two very common, if not the most common, applications of data science methods. While information retrieval and recommender systems share many similarities, there are certain differences that need to be addressed when planning, implementing, and deploying them.

Information retrieval and recommender systems are ubiquitously present on the World Wide Web supporting users to find, retrieve, and suggest information. Services ranging from search engines (Google), multimedia delivery (Netflix, Spotify), e-commerce (eBay, Amazon), and social networks (LinkedIn, Facebook, Twitter) have business models closely intertwined with how well their search and recommendation systems work. This very direct connection to commercial real-world applications makes research and development of information retrieval and recommender systems very tightly connected with industry.

References

1. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender systems handbook, pp. 191–226. Springer (2015)
2. Agichtein, E., Brill, E., Dumais, S.: Improving web search ranking by incorporating user behavior information. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06, pp. 19–26. ACM, New York, NY, USA (2006). DOI 10.1145/1148170.1148177. URL http://dx.doi.org/10.1145/1148170.1148177
3. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. **20**(4), 357–389 (2002). DOI 10.1145/582415.582416. URL http://dx.doi.org/10.1145/582415.582416
4. Amatriain, X., Pujol, J.M.: Data mining methods for recommender systems. In: F. Ricci, L. Rokach, B. Shapira (eds.) Recommender Systems Handbook, pp. 227–262. Springer (2015). DOI 10.1007/978-1-4899-7637-6_7. URL https://doi.org/10.1007/978-1-4899-76376_7
5. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval - the concepts and technology behind search, Second edition. Pearson Education Ltd., Harlow, England (2011). URL http://www.mir2ed.org/
6. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI'98, pp. 43–52. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998). URL http://dl.acm.org/citation.cfm?id=2074094.2074100

7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks **30**(1-7), 107–117 (1998). DOI 10.1016/S0169-7552(98)00110-X. URL https://doi.org/10.1016/S0169-7552(98)00110-X
8. Broder, A.Z.: A taxonomy of web search. SIGIR Forum **36**(2), 3–10 (2002). DOI 10.1145/792550.792552. URL http://doi.acm.org/10.1145/792550.792552
9. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction **12**(4), 331–370 (2002)
10. Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems. In: Recommender Systems Handbook, pp. 919–959. Springer (2015)
11. Carmel, D., Yom-Tov, E.: Estimating the Query Difficulty for Information Retrieval. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers (2010). DOI 10.2200/S00235ED1V01Y201004ICR015. URL https://doi.org/10.2200/S00235ED1V01Y201004ICR015
12. Chakrabarti, S.: Learning to rank in vector spaces and social networks. Internet Mathematics **4**(2), 267–298 (2007). DOI 10.1080/15427951.2007.10129291. URL https://doi.org/10.1080/15427951.2007.10129291
13. Croft, W.B., Metzler, D., Strohman, T.: Search Engines - Information Retrieval in Practice. Pearson Education (2009). URL http://www.search-engines-book.com/
14. Dooms, S., De Pessemier, T., Martens, L.: Movietweetings: a movie rating dataset collected from twitter. In: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013 (2013)
15. Felfernig, A., Boratto, L., Stettinger, M., Tkalčič, M.: Group recommender systems: An introduction. SPRINGERBRIEFS IN ELECTRICAL AND COMPUTER ENGINEERING (2018)
16. de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-aware content-based recommender systems. In: F. Ricci, L. Rokach, B. Shapira (eds.) Recommender Systems Handbook, pp. 119–159. Springer (2015). DOI 10.1007/978-1-4899-7637-6. URL https://doi.org/10.1007/978-1-4899-7637-6
17. Goldberg, D., Nichols, D.A., Oki, B.M., Terry, D.B.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992). DOI 10.1145/138859.138867. URL http://doi.acm.org/10.1145/138859.138867
18. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. **5**(4), 19:1–19:19 (2015). DOI 10.1145/2827872. URL http://doi.acm.org/10.1145/2827872
19. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004). DOI 10.1145/963770.963772. URL http://doi.acm.org/10.1145/963770.963772
20. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999). DOI 10.1145/324133.324140. URL http://doi.acm.org/10.1145/324133.324140
21. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pp. 426–434. ACM, New York, NY, USA (2008). DOI 10.1145/1401890.1401944. URL http://doi.acm.org/10.1145/1401890.1401944
22. Lavrenko, V., Croft, W.B.: Relevance-based language models. In: W.B. Croft, D.J. Harper, D.H. Kraft, J. Zobel (eds.) SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA, pp. 120–127. ACM (2001). DOI 10.1145/383952.383972. URL http://doi.acm.org/10.1145/383952.383972
23. Liu, B.: Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Second Edition. Data-Centric Systems and Applications. Springer (2011). DOI 10.1007/978-3-642-19460-3. URL https://doi.org/10.1007/978-3-642-19460-3
24. Liu, T.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval **3**(3), 225–331 (2009). DOI 10.1561/1500000016. URL https://doi.org/10.1561/1500000016

25. Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (eds.) Recommender Systems Handbook, pp. 73–105. Springer US, Boston, MA (2011)
26. Manouselis, N., Drachsler, H., Verbert, K., Santos, O.C.: Recommender Systems for Technology Enhanced Learning: Research Trends and Applications. Springer Publishing Company, Incorporated (2014)
27. Mooney, R.J., Bennett, P.N., Roy, L.: Book Recommending Using Text Categorization with Extracted Information. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), pp. 70–74. Madison, WI (1998)
28. Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine Learning **27**(3), 313–331 (1997)
29. Pizzato, L., Rej, T., Chung, T., Koprinska, I., Kay, J.: Recon: A reciprocal recommender for online dating. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, pp. 207–214. ACM, New York, NY, USA (2010). DOI 10.1145/1864708.1864747. URL http://doi.acm.org/10.1145/1864708.1864747
30. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, J. Zobel (eds.) SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia, pp. 275–281. ACM (1998). DOI 10.1145/290941.291008. URL http://doi.acm.org/10.1145/290941.291008
31. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. CoRR **abs/1205.2618** (2012). URL http://arxiv.org/abs/1205.2618
32. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW '94, pp. 175–186. ACM, New York, NY, USA (1994). DOI 10.1145/192844.192905. URL http://doi.acm.org/10.1145/192844.192905
33. Robertson, S.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends® in Information Retrieval **3**(4), 333–389 (2010). DOI 10.1561/1500000019. URL http://dx.doi.org/10.1561/1500000019
34. Robertson, S.E.: Readings in information retrieval. chap. The Probability Ranking Principle in IR, pp. 281–286. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997). URL http://dl.acm.org/citation.cfm?id=275537.275701
35. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms, pp. 143–160. Taylor Graham Publishing, London, UK, UK (1988)
36. Rocchio, J.J.: Relevance feedback in information retrieval. In: G. Salton (ed.) The Smart retrieval system - experiments in automatic document processing, pp. 313–323. Englewood Cliffs, NJ: Prentice-Hall (1971)
37. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975). DOI 10.1145/361219.361220. URL http://dx.doi.org/10.1145/361219.361220
38. Shaw, J.A., Fox, E.A.: Combination of multiple searches. In: TREC, vol. Special Publication 500-225, pp. 105–108. National Institute of Standards and Technology (NIST) (1994)
39. Singhal, A.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
40. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2, ICCV '03, pp. 1470–. IEEE Computer Society, Washington, DC, USA (2003). URL http://dl.acm.org/citation.cfm?id=946247.946751
41. Xu, S., Bao, S., Fei, B., Su, Z., Yu, Y.: Exploring folksonomy for personalized search. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08, pp. 155–162. ACM, New York, NY, USA (2008). DOI 10.1145/1390334.1390363. URL http://dx.doi.org/10.1145/1390334.1390363