— 1 —
Theory

1.1 Personalized Search

Personalized search means adapting the results of a search engine to each individual user. As we shall see, this field has a lot in common with user modeling and recommender system. In both situations, we wish to predict how relevant an item will be to each user. Before delving into the techniques of personalizing search results, we present the basics of *information retrieval* (IR).

Information Retrieval

Manning et al. (2008, p1) define IR as "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)".

How does this relate to recommender systems? There is an important distinction: The purpose of *recommending* is twofold: (1) show the user items similar to another item, and (2) allow discovery of relevant items the user did not know exist. The purpose of *search* is a bit different: allow the user to find the location of information he or she knows (or hopes) exists. In other words, the defining separator is often the knowledge of existence.

However, as we shall see in this chapter, the two fields employ a lot of the same methods and terminology. In the next chapter, we will show how these can work together.

Baeza-Yates and Ribeiro-Neto (1999, p23) presents a formal definition of an IR system: IR $= (Documents, Queries, Framework, ranking(q_i, d_i))$

As evident by the scope of IR literature, these elements can be just about anything that has to do with retrieving information. However, in what is often called *classic IR*, the documents contain free text with little to no describing structure, and the queries are short user-initiated descriptions of an *information need* (Baeza-Yates and Ribeiro-Neto, 1999, p19). Among other domains, this model describes web search engines, where the documents are web pages and queries are short sentences or a few keywords input by users.

The *Framework* in our quadruple refers to how documents are stored and retrieved. Basic

approaches to IR split each document into a set of terms (e.g. words), and create an inverted index (Manning et al., 2008, p22) that lists each document that each term appears in. There are numerous extensions to this framework, including:

- Positional information for phrase search (Manning et al., 2008, p39)
- Stop word removal (removing the most common terms) (Manning et al., 2008, p27)
- Stemming (reducing words to their root forms) (Manning et al., 2008, p32)
- Lemmatization (contextual inflection removal) (Manning et al., 2008, p32)
- Query reformulation (Baeza-Yates and Ribeiro-Neto, 1999, p117)

All these techniques help improve (among other things) the *recall* and *precision* of the retrieval engine. Recall and precision and relevance are well defined masures for evaluating the quality of a search engine (Manning et al., 2008, p5):

- A document is *relevant* if it satisfies the user's information need.
- *Recall* is the fraction of relevant documents retrieved by the system.
- *Precision* if the fraction of retrieved documents that are relevant.

There are many more measures, but these two succintly define what a search engine must to to be successfull: retrieve many relevant documents and few irrelevant documents. Failing this test is to neglect the main purpose of an IR: preventing information overload by allowing people efficient access to relevant parts of an otherwise overwhelming information repository.

To us, however, the most interesting part of any IR system is the *ranking function*. This function maps queries to documents by a scalar score, signifying how well a match each document is to a query. The relation to recommender systems should be self-evident, and indeed, IR systems use many of the same metrics to measure query/document similarity.

A common framework for storing and ranking documents is the previously mentioned vector space model (VSM). This model stores documents as term vectors. Each term represents a dimension, and documents are vectors in this term-space. When performing a query, the query terms are also represented as a vector in the same space. By computing the cosine similarity between the query and each document, we get a good estimate of how well a document matches a query (Baeza-Yates and Ribeiro-Neto, 1999, p29).

The next question is what to store at each (document, term) coordinate in the vector space (called the document-term weights). Storing simple 1 or 0 values representing whether

or not terms are present gives a model where a document's relevance is proportional to how many of the query terms it includes. However, this is not very precise. For example, by this definition, a document containing every conceivable query term would be the most relevant to any query. A better idea is to use something like the TF-IDF weighting scheme (Baeza-Yates and Ribeiro-Neto, 1999, p29):

$$w_{t,d} = tf_{t,d} \times idf_t = \frac{\text{freq}(t,d)}{\sum_{k \in d} freq(k,d)} \times \log \frac{N}{n_t}.$$

The final weight is computed by multiplying the term frequency score (TF) $tf_{t,d}$ with the inverse document frequency (IDF) idf_t. TF evaluates how well the term describes the document contents, while IDF punish terms that appear in many documents. $freq_{t,d}$ gives the frequency of a term in a document. N is the total number of documents, and n_t the number of documents in which t appears. The effect of the IDF factor is dampened by taking its log-value. Together, TF and IDF ranks documents higher by words that discriminate well within the document corpus, and ignores words that appear in so many documents that they have little to no predictive capacity.

While simple, TF-IDF has proven itself resilient when compared to more complex methods, and many more complex methods have been built on its foundations (e.g. BM25, one of the most successfull probabilistic weighting algorithms (Robertson, 2010)).

There are as many IR models as there are domains that need search, and even the basic vector space model can be constructed in a myriad of ways. There is also the simpler *boolean search model*, where queries are based on boolean algebra. Probabilistic models frame the similarity question as the probability that the document is relevant. Latent Semantic Indexing (LSI), the application of SVD to IR by performing dimensionality reduction of the term-space into concept-space is another approach. See Manning et al. (2008), Robertson (2010) or Baeza-Yates and Ribeiro-Neto (1999) for a more comprehensive introduction to models in IR.

Signals

Modern web search engines have long ago moved on from simple ranking metrics such as TF-IDF. While similar traditional metrics may form the foundation of modern search engines, a lot more thought go into the final results. Different types of rankings are combined to produce the final *search engine results page* (SERP), with each ranking

function often being referred to as a *signal*. Alternate names include *reranking* or *rescoring* functions.

Google, the company behind the popular online search engine, writes: "Today we use more than 200 signals, including PageRank, to order websites, and we update these algorithms on a weekly basis. For example, we offer personalized search results based on your web history and location."[1] Bing, another popular search engine, uses the same terminology: "We use over 1,000 different signals and features in our ranking algorithm."[2]

PageRank (Sergey and Lawrence, 1998, p4) is perhaps the most known of these additional signals. The algorithm forms a probability distribution over web pages, ranking their percieved authority or importance according to a simple iterative estimation. Each web site is given its rank based on how many pages that link to it. For each page that provides links, the score it contributes to the linked-to page is its own page rank weighted inversely proportional to the number of outbound links the page has. Another intuitive justification for a site's PageRank is the *random surfer model* (Sergey and Lawrence, 1998, p4). The probability that the random surfer visits a page is its PageRank. The "randomness" is introduced by a damping parameter d, which is the probability that a user will stop browsing and start at a new random page:

$$\mathrm{PageRank}(x) = \frac{1-d}{N} + d \sum_{y \in B_x} \frac{\mathrm{PageRank}(y)}{\mathrm{Links}(y)},$$

where B_x is the set of pages linking to page x, and $\mathrm{Links}(y)$ is the number of outbound links from page y. The first term distributes an equal pagerank score to all pages that have no outbound links, as N is the total number of pages.

Personalization

1.2 Aggregate Modeling

Current methods (non-personal)

Use cases (netflix, ...)

For personalized search (speculation)

(1) google.com/corporate/tech.html — accessed 11/04/2011
(2) bing.com/community/site_blogs/b/search/archive/2011/02/01/thoughts-on-search-quality.aspx — accessed 11/04/2011

Explain next chapter

References

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Manning, C., Raghavan, P., and Schutze, H. (2008). *Introduction to information retrieval*. Camebridge University Press.

Robertson, S. (2010). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Sergey, B. and Lawrence, P. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.