Information Retrieval: Concepts, Models, and Systems

Venkat N. Gudivada*,[1], Dhana L. Rao† and Amogh R. Gudivada*
*Department of Computer Science, East Carolina University, Greenville, NC, United States
†Department of Biology, East Carolina University, Greenville, NC, United States
[1]Corresponding author: e-mail: gudivadav15@ecu.edu

ABSTRACT
This chapter presents a tutorial introduction to modern information retrieval concepts, models, and systems. It begins with a reference architecture for the current Information Retrieval (IR) systems, which provides a backdrop for rest of the chapter. Text preprocessing is discussed using a mini Gutenberg corpus. Next, a categorization of IR models is presented followed by Boolean IR model description. Positional index is introduced, and execution of phrase and proximity queries is discussed. Various term weighting schemes are discussed next followed by descriptions of three IR models—Vector Space, Probabilistic, and Language models. Approaches to evaluating IR systems are presented. Relevance feedback techniques as a means to improving retrieval effectiveness are described. Various IR libraries, frameworks, and test collections are indicated. The chapter concludes by outlining facets of IR research and indicating additional reading.

Keywords: Information retrieval, IR models, BM25, Language model, TF-IDF, Probability of relevance framework, Vector space model, Boolean information retrieval, Divergence from randomness model, Relevance feedback

1 INTRODUCTION

We are generating unprecedented volumes of data in machine readable form. This data is of three types: structured, semi-structured, and unstructured. Structured data refers to that data which has predefined and clear semantics, and is amenable for storing and retrieving using traditional Relational Database Management Systems (RDBMS) (Codd, 1970). Structured Query Language (SQL) is the ANSI/ISO standard for inserting, retrieving, and modifying structured data stored in RDBMS (Chamberlin and Boyce, 1974). *Unstructured data* refers to data such as research papers, web pages, blog posts, email messages, twitter feeds, audio, and video. Information Retrieval

Handbook of Statistics. https://doi.org/10.1016/bs.host.2018.07.009

(IR) systems are used to store and retrieve unstructured (primarily textual) data. Manning et al. (2008, chap. 1, p. 1) 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)." IR systems are expected to find *relevant documents*, though the notion of relevance itself is not defined precisely (Saracevic, 2016). Furthermore, the semantics of unstructured data are context and user dependent, and extracting semantics is challenging. Extracting structured data from unstructured data is referred to as *information extraction* (Sarawagi, 2008).

Semi-structured data falls between structured and unstructured data in the sense that the data has some flexible structure to it. For example, Java Script Object Notation (JSON) and eXtensible Markup Language (XML) documents are semi-structured documents. During the last 5 years, numerous systems under the umbrella term NoSQL have been introduced to manage structured, semi-structured, and unstructured data (Gudivada et al., 2014, 2016). Search-Based Application (SBA) is a new model for applications that combine the strategies and structures of search engines and RDBMS to address performance-at-scale solutions to information management problems (Grefenstette and Wilber, 2010).

Traditional IR systems are focused on retrieving relevant documents to user queries from large text collections (Rijsbergen, 1979; Salton, 1968, 1983). Search techniques are at the core of IR systems. Search applications vary from domain-specific systems such as PubMed (a vertical search application) to desktop search (e.g., Spotlight search on Apple computers), systems that integrate search with analytics (e.g., Elasticsearch), and general-purpose search engines such as Bing and Google. An *IR model* is the primary component of an IR system. Fig. 1 depicts the structure of an IR model at an abstract level. The model provides two major functions: *indexing* and *search*. The *indexing* function provides a uniform and consistent internal representation for both documents and queries. Given the internal representation of a document and a query, the *search* function computes the relevance of the document to the query. This function is referred to as *retrieval/scoring* algorithm. Typically, the relevance is a numeric value and is referred to as *relevance status value* (RSV). An IR system computes the relevance of every document in the corpus to the query, and ranks them in decreasing order of their RSVs.

The retrieval/scoring algorithm is subject to heuristics/ constraints, and it varies from one IR model to another. For example, a *term frequency constraint* specifies that a document with more occurrences of a query term should be scored higher than a document with fewer occurrences of the query term. Also, the retrieval algorithm may be provided with additional information in the form of *relevance feedback* to enhance its scoring effectiveness. In one form of relevance feedback, the user explicitly indicates which of the retrieved documents is relevant. This information is used by the scoring

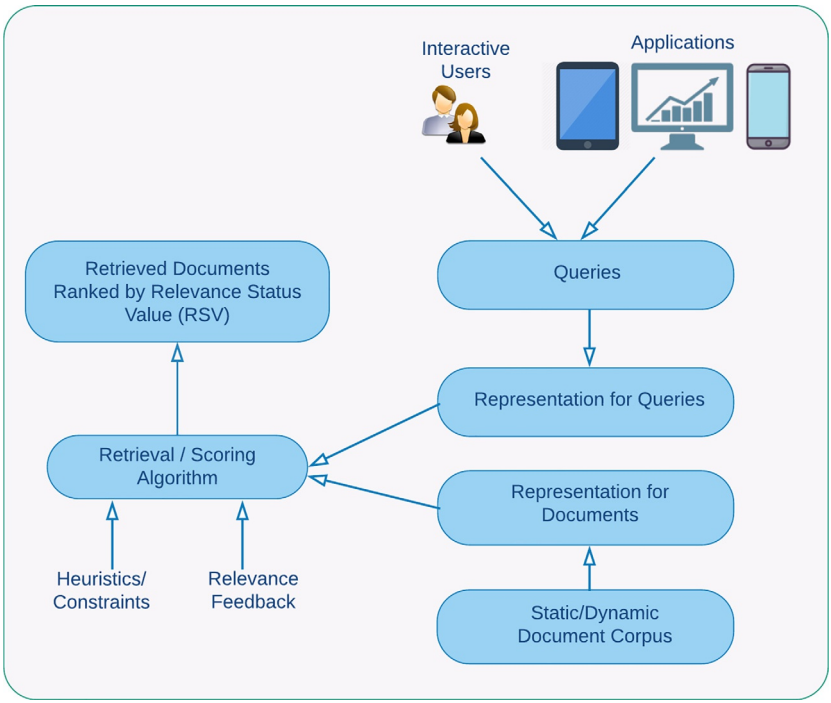

FIG. 1 An abstraction of an IR model.

algorithm to make changes to the user's initial query, and the revised query is re-executed. It is expected that the revised query will retrieve more relevant documents that meet the user's information need.

Unlike the SQL language used for querying RDBMS, IR system queries are typically short and free-form. Often IR system users do not use the right vocabulary to express their queries. This gives rise to mismatch between the user *information need* and the way the query is expressed. Furthermore, there is also uncertainty in the representation of documents, and the strategies used for matching user queries to documents. With the recent, rapid advances in Natural Language Processing (NLP), and Machine Learning (ML) in general and Deep Learning (DL) in particular (Barber, 2012; Murphy, 2012), there is a significant overlap between IR, NLP, and ML/DL (Li, 2014).

1.1 Chapter Organization

The goal of this chapter is to present a tutorial introduction to the fundamental IR concepts, retrieval models, system evaluation measures, and current systems. We limit the discussion to indexing and retrieving text documents. The chapter is organized as follows. In Section 2, a reference architecture for current IR systems is presented. This is intended to help the reader

understand how the various components of an IR system come together and lay the foundation for understanding the remainder of the chapter. Preprocessing operations on text data are discussed in Section 3. A simple corpus for describing IR concepts is presented in Section 4. Section 5 provides a taxonomy for categorizing IR models. The Boolean IR model, the simplest and also the most effective in certain contexts, is described in Section 6. Positional index and its use in processing phrase and proximity queries are discussed in Section 7. Assigning weights to terms that occur in documents to indicate their relative importance is presented in Section 8. In Sections 9–11, Vector space, probabilistic, and language model-based IR models are discussed. Measures for evaluating retrieval effectiveness of IR systems are discussed in Section 12. Improving retrieval effectiveness using relevance feedback is presented in Section 13. Section 14 presents software libraries and application frameworks for developing IR systems. Facets of IR research are listed in Section 15. Section 16 concludes the chapter by indicating additional reading.

2 A REFERENCE ARCHITECTURE FOR CURRENT IR SYSTEMS

A reference architecture for the current generation IR systems is shown in Fig. 2. The goal for this architecture is to provide a conceptual backdrop for the rest of the discussion in this chapter. In the figure, the circled numbers on the left are for exposition purpose only. We describe this architecture starting at the bottom layer and work our way upward. Layer ① denotes various sources of data for an IR system, which includes data in the computer file systems, application and end user-generated data, real-time event logs, and web crawlers. This data is ingested into an IR system using the Data Acquisition and Ingestion API, layer ②. Some of the ingested data is static and is used in read-only mode, whereas dynamic data may undergo updates. In both cases, data in its raw form is maintained along with derived data (e.g., inverted index) obtained through text preprocessing and other operations. The data stored in layer ③ encompasses structured, semi-structured, and unstructured types.

Layer ④ provides several functions. RDBMS and NoSQL systems are used to store and retrieve raw data, meta data, and other data generated from the raw data. This layer also provides preprocessing tools for text normalization, tokenization, stop word removal, stemming, and lemmatization. Lastly, this layer features functionality to weight document terms to indicate their relative importance. Layer ⑤ is the core of an IR system. It features one or more IR models—Boolean, Vector Space, Probabilistic, and Language Models, among others. An IR model specifies representation for both documents and queries and provides strategies for scoring relevance of documents to queries (Fig. 1). It also provides functionality for incorporating relevance and pseudo-relevance information to improve retrieval effectiveness. Lastly, this layer provides functions for personalization of search results, and machine learning algorithms for improving ranking of documents to queries.

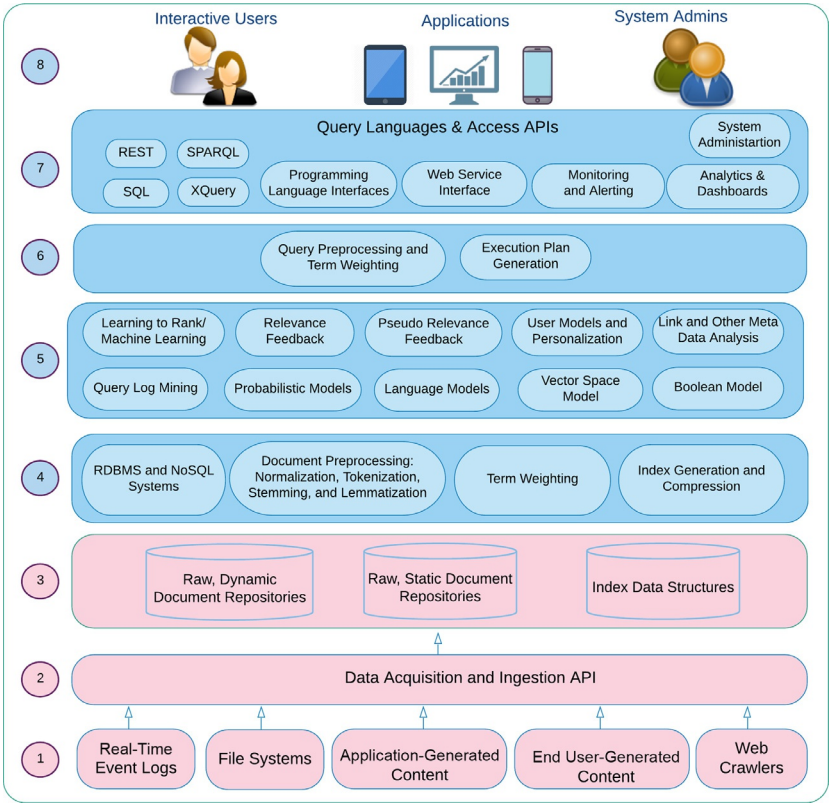

FIG. 2 A reference architecture for current IR systems.

Layer ⑥ parses IR queries, optionally determines weights for query terms, and produces a plan for efficiently executing queries. Layer ⑦ provides several Application Programming Interfaces (APIs) for three types of users—interactive end users, search applications, and system administrators. Layer ⑧ depicts client programs that consume IR system services.

3 DOCUMENT PREPROCESSING

We begin this section by defining terminology. *Indexing* is the process of creating both a *representation* for documents and associated *data structures* for storing and retrieving the representation. It is this representation that is used to determine the relevance of a document to a user query. For example, an inverted index is one such representation (discussed in Section 6).

3.1 Document Granularity

The first task of an IR system is to preprocess documents. In this context, we consider the following issues: document granularity, tokenization, and

normalization. Recall that the primary purpose of an IR system is to retrieve relevant documents to a user query from a large document collection. What exactly is a document? Is an entire book considered as one document? Or, is each chapter in a book a document? As another example, consider a chain of email messages. Is the entire chain one document or each email in the chain a separate document? The notion of document is important for almost all IR systems such as Elasticsearch (Gheorghe et al., 2015). Each document is assigned a unique identifier and forms the basic unit from a retrieval perspective. An exception to this is the Wumpus IR system (The University of Waterloo, 2018), which treats the entire document collection as one mega document, and each passage in the mega document is a unit of text for retrieval. For other documents such as journal articles, each document component—title, abstract, keywords, and rest of the article—is a separate unit for search. This enables selectively searching on the title, abstract, or the body of the article.

Precision and *recall* are the two parameters of retrieval effectiveness. Precision refers to how many of the retrieved documents are relevant to the user, whereas recall refers to what fraction of relevant documents in the collection are retrieved. *Indexing granularity* refers to the size/component of the document chosen for indexing. If the indexing granularity is high—for example, the entire book is considered as one document—may result in increased false positives (i.e., low precision). On the other hand, if the document granularity is too fine, it will entail low recall. This is because important passages will be missed as the passage vocabulary terms are split across mini documents.

3.2 Tokenization

A document is viewed as a sequence of bytes. ASCII encoding uses one byte per character and this suffices for the English language. Other encoding schemes such as Unicode UTF-8 uses one to four bytes to represent 1,112,064 distinct codes to accommodate characters in all written languages. A *token* corresponds to a sequence of bytes. For example, in "seven wonders of the world" there are five tokens and each is a different token type. On the other hand, in "to be, or not to be" there are seven tokens but only five distinct token types. The token types "to" and "be" each have two instances. Note that "," is a punctuation token. The whitespace-based demarcation of tokens works in languages such as English. Even in these languages, problems arise when phrases like "New York" and "North Carolina" are segmented. These phrases should be segmented as single tokens. Language-specific rules are effective in recognizing single-token phrases. In many other languages compound words abound and segmenting them into individual tokens is nontrivial. The text "Aliikusersuillammassuaanerartassagaluarpaalli" from Western Greenlandic language translates to "However, they will say that he is a great entertainer, but …." This is typical in polysynthetic languages where several *morphemes* (i.e., smallest units of meaning in a language) are strung together to make larger words.

Other tokenization issues include dealing with apostrophes (e.g., O'Brian, Carolina's), contractions (I'll), and hyphenated words (now-a-days). Acronyms also pose challenges (U.S.A/USA, Los Angles/LA, Louisiana/LA). *Token normalization* refers to canonicalizing tokens so that matches occur despite superficial differences (Manning et al., 2008). Token normalization is also referred to as *equivalence classing of tokens*. For example, various date forms such as 04/01/2018, 2018/04/01, and April 4th, 2018 denote the same date and therefore are members of the same equivalence class. Domain-specific mapping rules are used to specify equivalence classes. Alternatively, *relations* between unnormalized tokens are maintained using a hand-constructed list of synonyms. For example, synonyms of accelerate include advance, expedite, hasten, hurry, quicken, step up, and stimulate. Suppose that the term *accelerate* occurs in a document d_1. This term in unnormalized form is used to index d_1. Also, assume that a user issues a query which contains just the term *expedite*. An IR system may add additional terms in the synonyms list of *expedite* to the initial query, and the query is processed as a disjunction of synonyms. That is, the expanded query is: expedite *or* accelerate *or* advance *or* hasten *or* hurry *or* quicken *or* step up *or* stimulate. An alternative to query expansion is to index d_1 using the term *accelerate* and also all the terms in its synonym list—advance, expedite, hasten, hurry, quicken, step up, and stimulate.

3.3 Stemming and Lemmatization

Morphology is the study of words, their formation, and their relationship to other words in the same language. There are two types of morphology—inflectional and derivational. *Inflectional morphology* produces different forms of the same word rather than different words. Inflectional categories include number, tense, person, case, gender, among others. For example, leaves is produced from leaf, and both the original and new word belong to the same word category—nouns. In contrast, *derivational morphology* often involves the addition of derivational affixes, and affixation entails different categories for the new words. For example, the suffix "-ive" changes the word *select* to *selective*.

Other approaches to equivalence classing include stemming and lemmatization. *Stemming* is a crude heuristic process which collapses derivationally related words to their stem, base, or root form. There are two kinds of stemmers—algorithmic and dictionary. Algorithmic stemmers (e.g., Porter's) apply a set of rules to reduce a word to its stem form. In contrast, a dictionary stemmer looks up a dictionary to find the stem for a given word. Given the sentence "Other approaches to equivalence classing include stemming and lemmatization," Porter's algorithm reduces it to the following form: *Other approach to equival class includ stem and lemmat*. Lemmatization involves full morphological analysis of words to reduce inflectionally related and

sometimes derivationally related forms to their base form—lemma. The same sentence in the example above reduces to the following form through lemmatization: *Other approach to equivalence class include stemming and lemmatization*. Both stemming and lemmatization help to improve recall while hurting precision.

3.4 Stop Words, Accents, Case Folding, and Language Identification

Stop words are grammatical function words—for example, a, an, and, be, int, not, of, off, over, out, to, the, and under. Early IR system discarded stop words as they carry no content and exist only to meet grammatical requirements. If stop words are removed, phrases such as "to be, not to be, that is the question" do not get indexed correctly. Furthermore, for *phrase queries* (Section 6), presence of stop words contributes to better recall. Current IR systems including web search engines do not exclude stop words from indexing.

Accents and diacritics may be ignored in English text, but they can be quite significant for retrieval in other languages such as Spanish. *Case folding* is another normalization task. Beginning of the sentence words can be lowercased without retrieval implications. However, terms in the middle of a sentence should be left capitalized. Similar issues arise with acronyms. For applications such as web search engines, lowercasing everything is a pragmatic solution since users hardly use capitalization in their queries.

Writing systems of languages also pose problems for token extraction. In some writing systems, one reads from left to right, in others, right to left, and mixed (both left to right and right to left) yet in others. Though most documents on the web are written in English, of late, web documents written in other languages are becoming prevalent. In such cases, the first task is to identify the language of the document—Language Identification (LID). Short character sequences serve as distinctive signature patterns for the LID task. The LID problem has been solved with high degree of accuracy as a classification problem using supervised machine learning approaches. However, mixed-language documents, where a small fraction of words from another language are mixed, can create challenges to LID.

4 MINI GUTENBERG TEXT CORPUS

Though large IR datasets exist (see Section 14), we created a small text collection called *Mini Gutenberg* to illustrate select IR concepts. Project Gutenberg (2018) provides access to over 56,000 copyright-free or copyright-expired texts. We downloaded 18 books and created a *Mini Gutenberg* text collection. We lowercased all the words. Also, any extraneous preceding and trailing text from these documents was removed.

TABLE 1 Mini Gutenberg Text Collection

Doc ID	Document Name	Total Tokens	Token Types
d_1	austen-emma.txt	158,167	14,193
d_2	austen-persuasion.txt	83,308	11,481
d_3	austen-sense.txt	118,675	12,567
d_4	bible-kjv.txt	821,133	25,137
d_5	blake-poems.txt	6,845	3,021
d_6	bryant-stories.txt	45,988	7,669
d_7	burgess-busterbrown.txt	15,870	3,043
d_8	carroll-alice.txt	26,443	5,139
d_9	chesterton-ball.txt	79,543	15,739
d_{10}	chesterton-brown.txt	71,626	15,475
d_{11}	chesterton-thursday.txt	57,955	12,601
d_{12}	edgeworth-parents.txt	166,070	16,639
d_{13}	melville-moby-dick.txt	212,030	33,901
d_{14}	milton-paradise.txt	79,659	17,951
d_{15}	shakespeare-caesar.txt	21,245	5,687
d_{16}	shakespeare-hamlet.txt	29,605	9,399
d_{17}	shakespeare-macbeth.txt	17,741	6,895
d_{18}	whitman-leaves.txt	122,070	24,651
	Total	2,135,242	241,188

Table 1 shows the names of documents and other details about the *Mini Gutenberg* collection. The first word in the second column text is the document author and the second word/phrase is the name of the literary work. For example, carroll-alice.txt is *Alice's Adventures in Wonderland* (commonly referred to as *Alice in Wonderland*), which is an 1865 novel written by English author Charles Lutwidge Dodgson under the pseudonym Lewis Carroll. The third column shows the total number of tokens in the document, and the fourth column specifies the unique token types. As shown in the last row of the table, when summed document-wise, the collection has 2,135,342 tokens and 241,188 distinct token types. However, the collection has only 41,067 distinct token types at the *collection level*—across all the 18 documents.

These 41,067 tokens comprise the *vocabulary* of the collection. The vocabulary words/tokens are called *terms*.

If you try to reproduce these numbers, your numbers may be slightly different from the ones shown in Table 1. Furthermore, word counts also differ based on how contractions and apostrophes are handled and the definition of tokens used to isolate them from the byte sequence. We define a word as a sequences of characters from the set {a, …, z, A, …, Z} and words are delimited by white space.

4.1 Distribution of Characters

Knowledge of the distribution of letters in a language has several uses in addition to IR tasks. Such uses include decoding secret messages, solving word puzzle games, and designing keyboard layouts. Shown in Fig. 3 is the frequency of occurrence of characters in the Mini Gutenberg text collection. The vowel characters {a, e, i, o, u} occur most frequently.

4.2 Unigrams, Bigrams, and Trigrams

Unigrams are single words, *bigrams* are two-word sequences, and *trigrams* are three-word sequences. Likewise, an *n*-gram is a sequence of *n*-word sequences. Table 2 shows the top 10 frequently occurring unigrams, bigrams, and trigrams

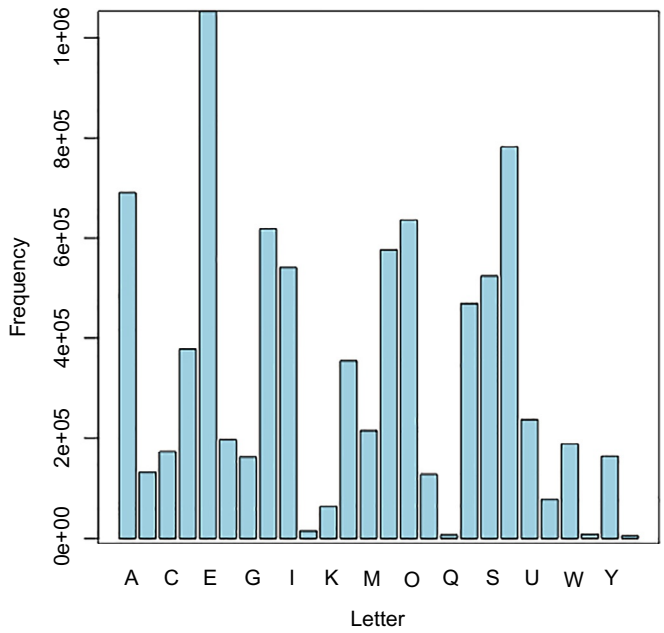

FIG. 3 Frequency of occurrence of characters in the Mini Gutenberg text collection.

TABLE 2 Top Ten Frequently Occurring Unigrams, Bigrams, and Trigrams in Mini Gutenberg Text Collection

Unigram	Frequency	Bigram	Frequency	Trigram	Frequency
the	133,603	of the	19,145	of the lord	1,780
and	95,444	in the	10,245	the son of	1,466
of	71,269	and the	8,880	the children of	1,368
to	48,060	the lord	7,081	out of the	1,202
a	33,962	to the	5,413	the house of	917
in	33,580	all the	3,600	saith the lord	854
i	30,300	and he	3,593	the lord and	818
that	28,811	to be	3,469	and i will	764
he	25,868	for the	3,049	and all the	697
it	22,304	it was	2,791	children of israel	647

in the Mini Gutenberg text collection. Note that the stop words dominate in bigrams and trigrams. The total number of unigrams, bigrams, and trigrams are 41,509, 580,644, and 1,416,000, respectively. The disk space for storing unigrams, bigrams, and trigrams is 541 KB, 8.5 MB, and 26.1 MB, respectively. Note that space requirements dramatically increase as we go from unigrams to trigrams.

4.3 Zip's Law

Zip's law is a commonly used model to describe the distribution of terms in a collection. To determine whether Zip's law holds for the *Mini Gutenberg* collection, we counted the number of occurrences of words/terms in the collection. Next, the words are listed in the decreasing order of their occurrence frequency. For a word w_i, let f_i be its frequency of occurrence and r_i be its rank position in the sorted list. For the 10th most frequently occurring word, for example, its rank is 10. Zipf's law states that for any given word, the product $f_i r_i$ is equal to some constant k. That is, $f_i r_i = k$. A *log–log* plot of *frequency of word occurrences* vs their ranks is shown in Fig. 4. A line with -1 slope corresponds to the Zip's function. We can conclude that the Zip's law holds reasonably well for the *Mini Gutenberg* collection.

Distribution of words revealed by Zip's law provides guidance for designing suitable data structures (term-document incidence matrix (Table 4, discussed in Section 6) vs positional inverted index (Table 8, discussed in Section 6)),

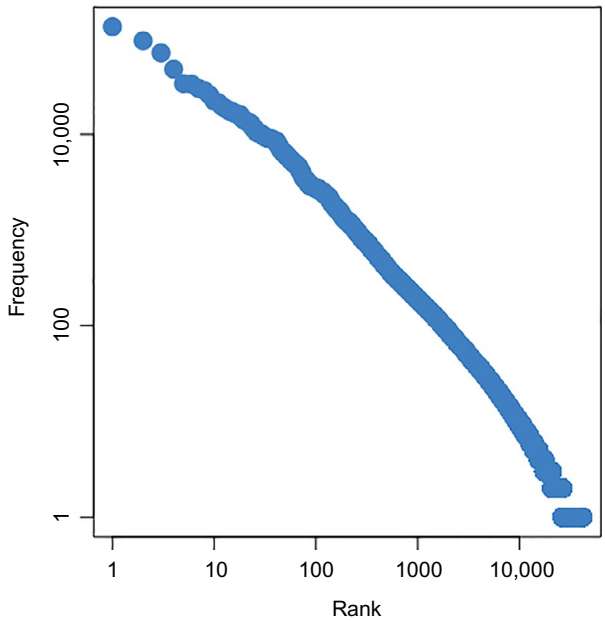

FIG. 4 Log–log plot of word frequencies and their ranks for Gutenberg corpus.

and offers heuristics for processing IR queries. Another key insight is that the collection has only a few frequent terms and many rare terms—a long right tail distribution. Frequent terms are less informative than rare terms. Term weighting schemes as well as retrieval function design should effectively address rare terms.

Advances in high-performance computing enable us to process enormous amounts of data using distributed architectures. MapReduce specifies an abstract framework and programming model for specifying computations in a cluster-based distributed computing environments. Lin and Dyer provide an exposition to MapReduce framework and its algorithm design (Lin and Dyer, 2010). They illustrate the application of MapReduce to text processing algorithms that are widely used in information retrieval, natural language processing, and machine learning domains.

5 A CATEGORIZATION OF IR MODELS

Table 3 shows a high-level categorization of IR models. In the sections that follow, we will discuss Boolean, Vector Space, Okapi BM25, and statistical language models in some detail.

Recall that terms are the foundational elements for representing both queries and documents. The order of the terms in documents and queries does

TABLE 3 A Categorization of IR Models

IR Model Class	Subclasses	Distinguishing Characteristics
Set-theoretic	Boolean	Represents documents by a set of *terms* (aka index terms). Each term is viewed as Boolean variable. A term's value in a document is true if the term is present in the document, false otherwise. Terms are not weighted. Queries are specified as arbitrary Boolean expressions, which are formed by linking terms with the logical connectives AND, OR, and NOT. Documents are not ranked (Salton, 1968)
	Fuzzy Set	Based on fuzzy set theory, which accommodates the notion of a *degree of membership* of terms in documents (Kraft and Colvin, 2017; Radecki, 1979). The degree of membership varies from 0 to 1. The logical connectives AND, OR, and NOT are suitably redefined to reflect the degree of membership. The model considers term correlations, but not the frequency of occurrence of terms in documents. Not a widely studied and used model
	Extended Boolean	Extends the Boolean model to enable partial matching and term weighting. It combines the characteristics of the Vector Space model with Boolean query formulation. Uses a generalized scalar product for computing document-query similarity. The well-known L_p norm defined for an n −dimensional vector is used for the scalar product. The interpretation of a query is altered by the value chosen for p. When $p = 1$, the model behaves like a Vector Space model; when $p = \infty$ and the query terms are all equally weighted, behaves like the fuzzy set model; when $p = \infty$ and the query terms are not weighted, behaves like the Boolean model
Algebraic	Vector Space	Both documents and queries are represented as vectors in an n-dimensional space, spanned by a set of orthonormal term vectors. The relevance of a document to a query is based on a similarity measure, which is the scalar product of the document and query vectors. It was the most widely used model until end of the last century
	Latent Semantic Indexing	Latent Semantic Analysis (LSA) is a technique in Natural Language Processing (NLP) for analyzing *distributional semantics*—relationships between a set of documents and the terms contained in them. LSA is based on the *distributional hypothesis*—words that are close in meaning will appear in similar documents. LSA estimates the pattern of word usage across documents. An enhanced term-document incidence matrix with term frequency counts is constructed, and the number of rows is reduced using Singular Value Decomposition (SVD) method. In the resulting low-dimensional space, each row is a term vector and the similarity between the terms is computed as the cosine of the angle between their term vectors. An IR model that uses this latent semantic structure is called latent semantic indexing (LSI). LSA assumes that terms and documents form a joint Gaussian model. However, this does not match the observed data. An alternative, Probabilistic Latent Semantic Analysis (PLSA) and its application to IR called the Probabilistic Latent Semantic Indexing (PLSI), has shown to perform better than LSA/LSI. An advantage of this technique is that queries can retrieve documents even when the query and documents have no common words

Continued

TABLE 3 A Categorization of IR Models—Cont'd

IR Model Class	Subclasses	Distinguishing Characteristics
	Generalized Vector Space	The Generalized Vector Space Model (GVSM) overcomes the pairwise orthogonality assumption of the Vector Space model by introducing term-to-term correlations (Wong et al., 1985). Term vectors are represented using smaller components called *minterms*, which are binary indicators of all patterns of occurrence of terms in documents. Minterms are represented as a vector, and a minterm vector represents one type of co-occurrence of terms in documents. For example, a minterm vector corresponding to terms t_1 and t_2 points to only documents in which t_1 and t_2 co-occur. For n terms, there will be 2^n minterm vectors. Pairwise orthogonal vectors associated with the minterms serve as the *basis vectors* for GVSM. The document-query similarity is calculated in the space of minterm vectors. The advantage of GVSM is that it considers correlations among terms. However, the computational cost is high for large document collections
	Neural Network	Neural network models for IR use Machine Learning (ML)-based approaches. They are an evolution of the traditional *learning-to-rank* models. While the learning-to-rank models employ ML algorithm using hand-crafted IR features, neural models learn representations of language directly from raw text. However, neural models require large-scale training data for their development. Neural approaches can be *shallow* or *deep* depending on the number of layers in the neural architecture. Some neural IR models base relevance on lexical matching only, while others are able to extract relevance from related terms as well use semantic matching. Current models seem to not perform well for queries that involve *rare* terms. Deep neural network models for IR is an active area of current IR research
Probabilistic	Classical	Probabilistic models are a family of models based on the *Probability Ranking Principle* (PRP). These models consider documents and queries as observations of random variables. They rank documents by estimating the probability of relevance for document-query pairs. The models differ based on the assumptions they make. The classical model makes *term independence assumption*—terms occur in documents independent of each other. Documents are ranked based on estimated probability of relevance of documents to query. Binary Independence Model (BIM) is a classical model which represents both documents and queries as binary vectors. Okapi BM25 models extends BIM by incorporating term frequency and document length normalization. Currently, the BM25 is a popular and widely used IR model

	Statistical Language Model	A statistical language model is a probability distribution over all word sequences in the language. The language model estimates the likelihood of all word sequences. In the language model approach to IR, there is a language model associated with each document. In one approach, the relevance of a document to a query is the probability that the query is most likely has been generated by the language model of the document. These probabilities are used to rank documents. Language model-based approaches to IR are popular and widely used (Hiemstra, 2001; Zhai, 2008)
		It is a generalization of Harter's 2-Poisson indexing model. The 2-Poisson model is based on the hypothesis that informative terms occur more frequently in a set of documents (referred to as *elite* set) than in the rest of the documents. For the terms which do not possess elite documents, their term frequencies follow a random distribution. The Divergence from Randomness (DFR) model is based on a simple idea: the more the divergence of a frequency of a term t in a document d from its *collection frequency*, the more the information carried by the term t in the document d. A DFR model is created by instantiating the three components of the model's framework: selecting a basic randomness model, applying the first normalization, and normalizing the term frequencies. Combination of these choices leads to different DFR models. Poisson model with Laplace after-effect and normalization 2 (aka PL2 model) is a well-known DFR model
	Probabilistic Inference	Probabilistic inference models are based on epistemological view of probability. Under this interpretation, probability is viewed as a degree of belief an individual places on the uncertainty of a particular situation. The model assumes that there exists an ideal *concept space* (aka domain of reference), and the elements of this space are *elementary concepts*. A proposition is a subset of the concept space. Documents, terms in the documents, and user queries are all represented as propositions, and a probability function p is defined on the concept space. For a document d, $p(d)$ is interpreted as the degree to which the concept space is covered by the knowledge contained in d. Likewise, for a given query q and document d, $p(q \cap d)$ quantifies the degree to which the knowledge common to both q and d is covered by the concept space. These probabilities are used to rank documents with respect to a query (Wong and Yao, 1995).
Axiomatic		Axiomatic approach provides a framework for developing new retrieval models. The approach is based on formally defined retrieval constraints at the term level. A retrieval model is found by searching in a space of candidate retrieval functions for one that satisfies a chosen set of retrieval constraints. Several new retrieval functions have been derived using the axiomatic approach. Experimental results show that the derived retrieval functions are more robust, and less sensitive to parameter settings than the existing retrieval functions with comparable optimal performance (Amigó et al., 2018; Fang et al., 2004)

not matter. This way of representing documents and queries is called the *bag of words* model. Most IR models use *bag of words* representation for documents and queries.

6 BOOLEAN IR MODEL

The Boolean model is one of the simplest and earliest IR models. First we describe a data structure called *term-document incidence matrix* (Table 4). Recall that the *Mini Gutenberg* collection has 18 documents and its vocabulary size is 41,067 (Section 4). In the *term-document incidence matrix*, each row corresponds to a term and each column corresponds to a document. Therefore, the size of the matrix is $41,067 \times 18$. A cell connects a term with a document—a value of 1 indicates that the term is present in the document, and 0 indicates the absence of the term. Each row in the matrix is a *term incidence vector* (or simply term vector). For example, the incidence vector of the term *serenity* is (1 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1). Each column corresponds to a *document vector*; 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 is the document vector of d_1. The terms in the matrix are maintained in sorted order for processing efficiency. Using this matrix, one can easily find all the documents that contain a given term, and all the terms that are present in a document.

In the Boolean model, a document is either relevant or nonrelevant to a query; there is no *degree of relevance*. The model is based on Boolean logic and classical set theory. Terms are viewed as Boolean variables—the value of a term is *true* or 1 with respect to a document if the term is present in the document; *false* or 0, otherwise. Just as documents are represented by document vectors, *Boolean expressions* are used to represent queries. *Boolean expressions* are written in either *conjunctive normal form* or *disjunctive normal form*. For example, the query expression $(t_1 \vee t_2 \vee t_3) \wedge (t_4 \vee t_5)$ is in *conjunctive normal form*—a conjunction (i.e., AND/$\wedge$) of disjunctions (i.e., OR/$\vee$). The expression $(t_1 \wedge t_2 \wedge t_3) \vee (t_4 \wedge t_5)$ is in *disjunctive normal form*. Terms may be negated in the expressions, for example, $\neg t_1$. Any Boolean function can be expressed using only two levels of logic and possible negation of terms. Therefore, conjunctive and disjunctive normal forms are two ways to express any Boolean function.

A *Boolean query* is any *valid Boolean expression* formed using vocabulary terms and the three Boolean operators: AND, OR, and NOT. The simplest Boolean query is just the term itself. For the term *serenity* as the query, the retrieved documents would be: d_1, d_9, d_{11}, d_{12}, d_{13}, d_{14}, d_{18}. These documents correspond to 1s in the serenity's incidence vector. Next, consider a more interesting query: retrieve documents that contain the terms *flirt* and *gait*—(flirt $\wedge$ gait). The corresponding incidence vectors are: (1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0) and (0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1). To process this query, we perform a bitwise AND on these two vectors, which produces the results vector: (0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0). The query returns

TABLE 4 Partial Term-Document Incidence Matrix for the Mini Gutenberg Text Collection

	Documents																	
Terms	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}
acorn	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
barbaric	0	0	1	1	0	0	0	0	1	1	1	1	1	1	0	1	0	1
beautiful	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
daughter	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1
document	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	0
flirt	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
gait	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1
handkerchief	0	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	1
hilarious	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
influence	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	1
majesty	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	1
melville	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
moby	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
ominous	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0	1
serenity	1	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	1
skeleton	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1
tremendous	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1
umbrage	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

the set of documents that correspond to the 1s in the results vector, which is $\{d_{13}\}$. For the OR operator, we perform a bitwise OR operation on the incidence vectors.

Consider a more complex query: retrieve documents that contain the terms *ominous* and *serenity*, but not *skeleton*. This query expressed as a Boolean expression is: *ominous* AND *serenity* AND (NOT *skeleton*), or equivalently *ominous* $\wedge$ *serenity* $\wedge$ ($\neg$ *skeleton*). The effect of applying the NOT operator on the *skeleton* vector is same as complementing its incidence vector. Lastly, we perform a bitwise AND of this vector with the two vectors corresponding to *ominous* and *serenity* vectors (Table 5). The documents retrieved for this query are those that are associated with incidence vector in the last row of Table 5, which are $\{d_{12}, d_{14}\}$.

Boolean IR model does not rank documents. A document is either relevant or nonrelevant and there is no provision for indicating a degree of relevance. Furthermore, it does not take into account the frequency of occurrence of terms in documents. The model is simple to implement. It is surprisingly effective in retrieving relevant documents, provided the end user knows the domain vocabulary and can write relatively complex Boolean queries. It is especially suitable for domains such as legal research, where achieving high recall is the overriding goal even at the cost of low precision. The commercially successful Thomson Reuters Westlaw system attests to the suitability and effectiveness of Boolean IR systems for niche domains.

A close look at Table 4 reveals that the matrix is very sparse. In reality, over 99.9% of cells will have zero values. An *inverted index* is a more efficient data structure for representing term-document relationships. Table 6 shows the partial inverted index for the *Mini Gutenberg* collection. The inverted index has two components: dictionary and postings. The dictionary is shown in the first column. It is an ordered list of vocabulary terms. In addition to the term name, the collection frequency of the term is also indicated. The collection frequency of a term is the number of times the term appears across all the documents in the collection. Paired with each term in the

TABLE 5 Processing a Complex Boolean Query

Boolean Operators Applied	Term	Incidence Vector
	skeleton	(0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1)
NOT	*skeleton*	(1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0)
	ominous	(0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1)
	serenity	(1 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1)
ominous AND *serenity* AND (NOT *skeleton*)		(0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0)

TABLE 6 Partial Inverted Index for the *Mini Gutenberg* Collection

Dictionary		Postings
acorn, 1	→	d_1
barbaric, 10	→	$d_3\ d_4\ d_9\ d_{10}\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{16}\ d_{18}$
beautiful, 17	→	$d_1\ d_2\ d_3\ d_4\ d_5\ d_6\ d_7\ d_8\ d_9\ d_{10}\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{15}\ d_{16}\ d_{18}$
daughter, 15	→	$d_1\ d_2\ d_3\ d_4\ d_5\ d_6\ d_8\ d_{10}\ d_{12}\ d_{13}\ d_{14}\ d_{15}\ d_{16}\ d_{17}\ d_{18}$
document, 5	→	$d_2\ d_9\ d_{10}\ d_{13}\ d_{16}$
flirt, 4	→	$d_1\ d_2\ d_{10}\ d_{13}$
gait, 6	→	$d_{10}\ d_{11}\ d_{13}\ d_{14}\ d_{15}\ d_{18}$
handkerchief, 7	→	$d_3\ d_4\ d_9\ d_{10}\ d_{12}\ d_{13}\ d_{18}$
hilarious, 3	→	$d_9\ d_{11}\ d_{13}$
influence, 11	→	$d_1\ d_2\ d_3\ d_4\ d_9\ d_{10}\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{18}$
majesty, 8	→	$d_4\ d_6\ d_8\ d_9\ d_{12}\ d_{13}\ d_{14}\ d_{18}$
melville, 1	→	d_{13}
moby, 1	→	d_{13}
ominous, 7	→	$d_9\ d_{10}\ d_{12}\ d_{13}\ d_{14}\ d_{16}\ d_{18}$
serenity, 7	→	$d_1\ d_9\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{18}$
skeleton, 3	→	$d_9\ d_{13}\ d_{18}$
tremendous, 7	→	$d_6\ d_9\ d_{10}\ d_{11}\ d_{12}\ d_{13}\ d_{18}$
umbrage, 1	→	d_{14}

dictionary is a postings list, which enumerates the documents in which the term occurs. Compared to the term-document incidence matrix, inverted index is an efficient representation from the perspective of both storage and processing requirements.

The number of documents in a typical IR system is in the order of millions and billions. In the case of web search engines, which are ultra large-scale IR systems, the number of documents is in the order of trillions. For today's standards, Reuters RCV1 collection is a small one (Lewis et al., 2004). It consists of about 800,000 Reuters news wires and the number of tokens exceeds 1 million. Creating an inverted index for enterprise IR systems and web search engines poses special challenges. The size of the intermediate files generated during the index construction process is orders of magnitude greater than the primary memory available on most computers. Therefore, *external sorting* algorithms and compression/decompression techniques are

critical to index construction and usage. The reader is referred to Witten et al. (1999) and Manning et al. (2008) for details.

7 POSITIONAL INDEX, PHRASE, AND PROXIMITY QUERIES

In this section, we discuss creation of a positional index and how it is used to process phrase and proximity queries. We use the five-document tiny *corpus* (documents $\{d_1, d_2, \ldots, d_5\}$) shown in Table 7 to illustrate positional index construction. We will use the index to process Boolean queries, *phrase* and *proximity queries*.

The tiny corpus tokens in sorted order are: a, address, also, analytics, and, approaches, are, as, assessment, being, between, central, concerned, considerations, context, dealing, deals, design, discussed, enhance, epistemological, epistemology, evaluating, feedback, how, in, incompleteness, interplay, is, issues, it, learning, measured, measurement, models, nature, next, of, other, pedagogy, play, prediction, providing, related, role, should, subjectivity, such, teaching, the, to, two, validation, what, whereas, with.

Shown in Table 8 is the positional inverted index for the tiny corpus. This is essentially an inverted index enhanced with positional information. Consider the first entry in the index: a, 1 $\langle d_1, 1 : [96] \rangle$. It indicates that the term *a* occurs in the corpus 1 time (i.e., collection frequency), and appears in document d_1 1 time at byte offset 96. Think of the byte offset as how many characters away is the first character in the term *a* from the beginning of the document. As another example, consider the entry: with, 3 $\langle d_2, 1 : [106]; d_4, 2 : [16, 89] \rangle$. It indicates that the term *with* appears in the corpus a total of 3 times, in document d_2 1 time at byte offset 106; in document d_4 2 times, first instance at byte offset 16 and the second instance at byte offset 89. Byte offset calculation assumes that the terms are separated by a space character. To keep this example conceptually simple, we did not perform stop word removal, stemming, or

TABLE 7 A Tiny Corpus for Positional Index Illustration

d_1	epistemological considerations such as what is being measured and how is it measured also play a central role
d_2	epistemological considerations should also address learning design and measurement approaches to dealing with subjectivity and incompleteness and validation of prediction models
d_3	two other related issues are pedagogy and assessment
d_4	pedagogy deals with the nature of learning and teaching, whereas assessment is concerned with evaluating learning and providing feedback to enhance learning
d_5	the interplay between epistemology assessment and pedagogy in the context of learning analytics is discussed next

TABLE 8 Positional Inverted Index for a Tiny Corpus

Dictionary	Postings With Positional Information
a, 1	$\langle d_1, 1 : [96] \rangle$
address, 1	$\langle d_2, 1 : [44] \rangle$
also, 2	$\langle d_1, 1 : [86]; d_2, 1 : [39] \rangle$
analytics, 1	$\langle d_5, 1 : [87] \rangle$
and, 8	$\langle d_1, 1 : [63]; d_2, 3 : [68, 124, 143]; d_3, 1 : [39]; d_4, 2 : [44, 114]; d_5, 1 : [47] \rangle$
approaches, 1	$\langle d_2, 1 : [84] \rangle$
are, 1	$\langle d_3, 1 : [26] \rangle$
as, 1	$\langle d_1, 1 : [37] \rangle$
assessment, 3	$\langle d_3, 1 : [43]; d_4, 1 : [65]; d_5, 1 : [36] \rangle$
being, 1	$\langle d_1, 1 : [48] \rangle$
between, 1	$\langle d_5, 1 : [15] \rangle$
central, 1	$\langle d_1, 1 : [98] \rangle$
concerned, 1	$\langle d_4, 1 : [79] \rangle$
considerations, 2	$\langle d_1, 1 : [17]; d_2, 1 : [17] \rangle$
context, 1	$\langle d_5, 1 : [67] \rangle$
dealing, 1	$\langle d_2, 1 : [98] \rangle$
deals, 1	$\langle d_4, 1 : [10] \rangle$
design, 1	$\langle d_2, 1 : [61] \rangle$
discussed, 1	$\langle d_5, 1 : [100] \rangle$
enhance, 1	$\langle d_4, 1 : [140] \rangle$
epistemological, 2	$\langle d_1, 1 : [1]; d_2, 1 : [1] \rangle$
epistemology, 1	$\langle d_5, 1 : [23] \rangle$
evaluating, 1	$\langle d_4, 1 : [94] \rangle$
feedback, 1	$\langle d_4, 1 : [128] \rangle$
how, 1	$\langle d_1, 1 : [67] \rangle$
in, 1	$\langle d_5, 1 : [60] \rangle$
incompleteness, 1	$\langle d_2, 1 : [128] \rangle$
interplay, 1	$\langle d_5, 1 : [5] \rangle$

Continued

TABLE 8 Positional Inverted Index for a Tiny Corpus—Cont'd

Dictionary	Postings With Positional Information
is, 4	$\langle d_1, 2 : [45, 71]; d_4, 1 : [76]; d_5, 1 : [97]\rangle$
issues, 1	$\langle d_3, 1 : [19]\rangle$
it, 1	$\langle d_1, 1 : [74]\rangle$
learning, 5	$\langle d_2, 1 : [52]; d_4, 3 : [35, 105, 148]; d_5, 1 : [78]\rangle$
measured, 2	$\langle d_1, 2 : [54, 77]\rangle$
measurement, 1	$\langle d_2, 1 : [72]\rangle$
models, 1	$\langle d_2, 1 : [172]\rangle$
nature, 1	$\langle d_4, 1 : [25]\rangle$
next, 1	$\langle d_5, 1 : [110]\rangle$
of, 3	$\langle d_2, 1 : [158]; d_4, 1 : [32]; d_5, 1 : [75]\rangle$
other, 1	$\langle d_3, 1 : [5]\rangle$
pedagogy, 3	$\langle d_3, 1 : [30]; d_4, 1 : [1]; d_5, 1 : [51]\rangle$
play, 1	$\langle d_1, 1 : [91]\rangle$
prediction, 1	$\langle d_2, 1 : [161]\rangle$
providing, 1	$\langle d_4, 1 : [118]\rangle$
related, 1	$\langle d_3, 1 : [11]\rangle$
role, 1	$\langle d_1, 1 : [106]\rangle$
should, 1	$\langle d_2, 1 : [32]\rangle$
subjectivity, 1	$\langle d_2, 1 : [111]\rangle$
such, 1	$\langle d_1, 1 : [32]\rangle$
teaching, 1	$\langle d_4, 1 : [48]\rangle$
the, 3	$\langle d_4, 1 : [21]; d_5, 2 : [1, 63]\rangle$
to, 2	$\langle d_2, 1 : [95]\rangle\langle d_4, 1 : [137]\rangle$
two, 1	$\langle d_3, 1 : [1]\rangle$
validation, 1	$\langle d_2, 1 : [147]\rangle$
what, 1	$\langle d_1, 1 : [40]\rangle$
whereas, 1	$\langle d_4, 1 : [57]\rangle$
with, 3	$\langle d_2, 1 : [106]; d_4, 2 : [16, 89]\rangle$

lemmatization. Instead of the byte offset, one may use the number of the word from the beginning of the document. Under this scheme, the first entry in the index would be recorded as: a, 1 $\langle d_1, 1 : [16]\rangle$—the term *a* is the 16th word on document d_1.

7.1 Processing Boolean Queries Using the Positional Inverted Index

Consider processing the Boolean query: assessment *AND* measurement. The documents in the postings list for the term assessment are $\{d_3, d_4, d_5\}$ and the same for the term measurement is $\{d_2\}$. The Boolean *AND* operator corresponds to the classical *set intersection* operation. Therefore, to answer the query assessment *AND* measurement, take the intersection of the corresponding document sets. Since $\{d_3,d_4,d_5\} \cap \{d_2\} = \emptyset$, no documents are retrieved for this query.

Next consider the Boolean query: learning *OR* pedagogy. The Boolean *OR* operator corresponds to the classical *set union* operation. The documents in the postings list for the term learning are $\{d_2, d_4, d_5\}$ and the same for the term pedagogy are $\{d_3, d_4, d_5\}$. Since $\{d_2,d_4,d_5\} \cup \{d_3,d_4,d_5\} = \{d_2,d_3,d_4,d_5\}$, documents retrieved for this query are $\{d_2, d_3, d_4, d_5\}$.

Lastly, consider the Boolean query: *NOT* pedagogy. The Boolean *NOT* operator corresponds to the classical *set complement* operation. The complement is with respect to the universal set, which is all the documents in the collection—$\{d_1, d_2, d_3, d_4, d_5\}$. The documents in the postings list for the term pedagogy are $\{d_3, d_4, d_5\}$ and its complement is: $\{d_1, d_2, d_3, d_4, d_5\} - \{d_3, d_4, d_5\} = \{d_1, d_2\}$. Therefore, the documents retrieved for this query are $\{d_1, d_2\}$. In passing, it should be noted that term positional information is not required for processing Boolean queries. In contrast, term positional information is essential for processing *phrase* and *proximity* queries.

7.2 Processing Phrase Queries Using the Positional Inverted Index

Often, there is a need to identify documents that contain multiword compounds or phrases, for example, *epistemological considerations*. Phrase queries will retrieve only those documents in which the terms *epistemological* and *considerations* occur next to each other with no intervening words in between. Though some documents may contain both these terms, however, they should not be retrieved if the terms do not occur in the same order in consecutive positions in the document.

From Table 8, consider the following positional postings lists: epistemological, 2 $\langle d_1, 1 : [1]$; $d_2, 1 : [1]\rangle$; considerations, 2 $\langle d_1, 1 : [17]$; $d_2, 1 : [17]\rangle$. The terms *epistemological* and *considerations* occur in both d_1 and d_2. First consider document d_1. The byte offset of the term *epistemological* is 1, and the length of the term is 15. We include d_1 in the result set if the byte offset of the term *considerations* is 17, which is indeed the case. How did we come up

with the number 17? It is the sum of the byte offset of the term *epistemological*, length of *epistemological* (which is 15), and 1 (to account for space character between the terms). By the same reasoning, we also include document d_2 in the result set. Therefore, the result set for the phrase query *epistemological considerations* is $\{d_1, d_2\}$. As another example, verify that the result set for the three-term phrase query *learning and teaching* is $\{d_4\}$.

7.3 Processing Proximity Queries Using the Positional Inverted Index

Proximity queries are a variation on phrase queries. We relax the constraint that the terms need not occur in consecutive positions. Instead, proximity queries allow the terms to occur within k words of each other. Consider the proximity query: *learning measurement* ~ 2. The query states that the terms *learning* and *measurement* must occur in the document separated by no more than 2 intervening terms. Verify that the result set for this query is $\{d_2\}$. A slight modification to the logic used for processing the phrase queries is required for executing proximity queries. For algorithmic details, the reader is referred to Manning et al. (2008) and Büttcher et al. (2010).

7.4 Recovering Document Source Text Using the Positional Inverted Index

We can also reconstruct the original document source text from the positional inverted index. Consider recovering document d_3. First, construct the first two columns of Table 9 by scanning the positional inverted index of Table 8 for document d_3. Next, construct the last two columns of the table by sorting

TABLE 9 Reconstructing a Document Source From Positional Inverted Index

Term	Positions in Documents	Term	Positions in Documents Sorted by Byte Offset
and	d_3, 1 : [39]	two	d_3, 1 : [1]
are	d_3, 1 : [26]	other	d_3, 1 : [5]
assessment	d_3, 1 : [43]	related	d_3, 1 : [11]
issues	d_3, 1 : [19]	issues	d_3, 1 : [19]
other	d_3, 1 : [5]	are	d_3, 1 : [26]
pedagogy	d_3, 1 : [30]	pedagogy	d_3, 1 : [30]
related	d_3, 1 : [11]	and	d_3, 1 : [39]
two	d_3, 1 : [1]	assessment	d_3, 1 : [43]

terms based on their byte offset. Read terms in column three from top to bottom and concatenate them to reconstruct d_3, which gives: *two other related issues are pedagogy and assessment*.

8 TERM WEIGHTING

Terms are content descriptors which are assigned to documents, and this process is called *indexing*. These terms in turn are used to assess relevance of documents to queries. Terms are of two types: objective and nonobjective. Objective terms apply integrally to the document and usually there is no disagreement about how to assign them, for example, author names, document publication date, and document location. In contrast, *nonobjective terms* are assigned to reflect the information content of the document. They can be assigned manually or automatically. Optionally, a weight may be assigned to nonobjective terms to indicate the degree to which they reflect the document's information content. Such terms are referred to as weighted terms and the process is referred to as *term weighting*. Our focus in this section is on *nonobjective terms* and we refer to them as just *terms* for simplicity.

The effectiveness of an indexing system is governed primarily by two parameters—indexing exhaustivity and term specificity (Salton and Buckley, 1988). *Indexing exhaustivity* refers to the degree to which all the concepts manifested in a documents are captured by the indexing system. When indexing is exhaustive, typically more terms are assigned to documents; when indexing is *nonexhaustive*, fewer terms are assigned. *Term specificity* refers to the degree of generality or narrowness of the terms. Broad terms contribute to retrieving several useful documents along with a significant number of nonrelevant ones. In contrast, narrow terms retrieve relatively fewer documents and may miss many relevant ones. *High term specificity* refers to terms that are narrow and specific, and it contributes to higher precision at the expense of recall. In contrast, broad and nonspecific terms contribute to higher recall while decreasing precision.

8.1 Log Frequency Term Weighting

Intuitively, the larger the frequency of a term t in a document, the greater should be the weight assigned to t to reflect its importance. However, if a term t occurs five times in document d_1 and one time in another document d_2, it is not the case that for a query with term t, d_1 is five times more relevant compared to d_2. Therefore, *raw term frequency* is not a suitable measure. Typically, transformations are applied on the *raw term frequency* and the result is used as the term weight. One such method is *log transform* and the weighting scheme is referred to as *log frequency weighting*. The *log frequency weight* of a term t in document d (denoted $w_{t,d}$) is defined as:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The reasoning behind the *log transform* is that the first occurrence of a term in a document is important, the second occurrence is also important but the degree of importance is relatively less, the third occurrence is even less important than the second occurrence, and so on. One is added to $\log_{10} \mathrm{tf}_{t,d}$ to avoid smaller weights to terms that occur a small number of times, and thus circumvent arithmetic *underflow problems*. The *base* of the logarithm does not matter as long as it is consistently used throughout.

A variation of Eq. (1) is also used for term weighting:

$$\mathrm{w}_{t,d} = \begin{cases} 1 + \log_{10}(1 + \log_{10} \mathrm{tf}_{t,d}) & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Table 10 shows $1 + \log_{10} \mathrm{tf}_{t,d}$ and $1 + \log_{10}(1 + \log_{10} \mathrm{tf}_{t,d})$ values for select values of $\mathrm{tf}_{t,d}$. As the term frequency increases from 0 to 10,000, the *log* frequency transformed values range from 0 to 5, and *log–log* frequency transformed values range from 0 to 1.6989.

Fig. 5 depicts how $1 + \log_{10} \mathrm{tf}_{t,d}$ and $1 + \log_{10}(1 + \log_{10} \mathrm{tf}_{t,d})$ transformations dampen term weights compared to original frequency values. The $y = x$ line is an *identity transform* of term weights, where x is frequency count of a term before the transformation, and y is the term frequency count after the transformation.

8.2 TF-IDF Weighting

Representing documents and queries using term frequency helps to achieve only one of the indexing goals—recall. The term frequency measure does

TABLE 10 Log Frequency Weighting of Terms

	$\mathrm{w}_{t,\,d}$	
$tf_{t,d}$	$1 + \log_{10} \mathrm{tf}_{t,d}$	$1 + \log_{10}(1 + \log_{10} \mathrm{tf}_{t,d})$
0	0	0
1	1.0	1.0
2	1.3010	1.1143
3	1.4771	1.1694
10	2.0	1.3010
500	3.6989	1.5681
5,000	4.6989	1.6720
10,000	5.0	1.6989

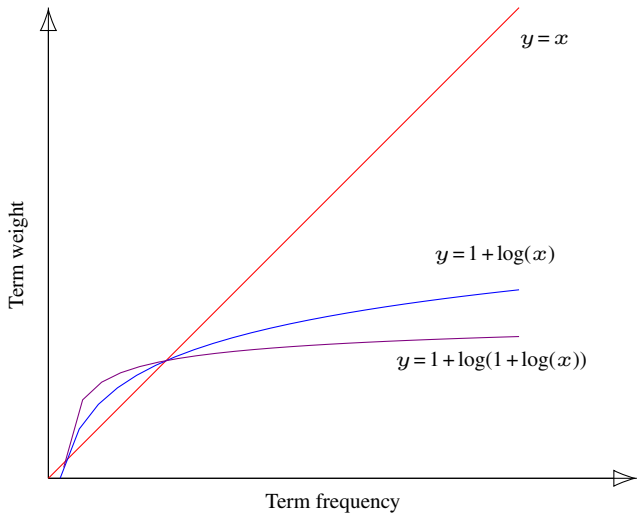

FIG. 5 Log frequency weighting of terms.

not capture the importance of terms that occur rarely across individual documents of a collection. Such rare terms are useful in distinguishing documents in which they occur from those in which they do not occur. Inverse Document Frequency (IDF) measure is an appropriate indicator of a rare term as a document discriminator. The IDF measure helps to improve precision. This suggests that both the term frequency and the inverse document frequency measures can be combined into a single frequency-based measure using the TF-IDF-based term weighting to improve both precision and recall.

Table 11 defines different *frequency types*. The inverse document frequency of a term t, denoted idf_t, is given by:

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t} \tag{3}$$

where N is the number of documents in the collection, and df_t is the document frequency of term t. The TF-IDF weight of a term t with respect to document d is defined as:

$$\text{w}_{t,d} = (1 + \log_{10} \text{tf}_{t,d}) \cdot \left(\log_{10} \frac{N}{\text{df}_t} \right) \tag{4}$$

The TF-IDF weight increases with term frequency in the document. It also increases with the rarity of the term in the collection (i.e., IDF). The TF-IDF weighting is one of the best known weighting schemes in IR (Salton, 1988).

Table 12 shows various frequency counts for the mini Gutenberg text corpus. For the term *the*, its IDF is 0 since it occurs in all the 18 documents.

TABLE 11 Different Types of Term Frequencies

Frequency Name	Description
Term frequency ($tf_{t,d}$)	The number of times a term t occurs in a document d
Document frequency (df_t)	The number of documents in the collection in which a term t occurs
Collection frequency (cf_t)	The number of times a term t occurs across all the documents in the collection
Inverse document frequency (idf_t)	Let N be the total number of documents in a collection, df_t be the document frequency of term t. The inverse document frequency of a term t, denoted idf_t, is given by $\log \frac{N}{df_t}$

TABLE 12 Various Frequencies for the Mini Gutenberg Text Corpus

Term	Document Frequency	Collection Frequency	IDF	Term Frequency in Document d_1
the	18	133,572	0.0	648
people	17	2,781	0.0248	4
thee	11	4,807	0.2139	61
truths	7	14	0.4101	2
fixing	5	14	0.5563	0
manifested	2	14	0.9542	0
vigil	1	13	1.2553	0

The term *people* occurs in 17 documents and its IDF is above zero at 0.0248. Lastly, the term *vigil* appears in only one document and its collection frequency is 13. This term is considered a *rare term* compared to *the*, and the IDF of *vigil* is 1.2533.

The IDF affects the ranking of documents for queries that have at least two terms, and it has very little effect on ranking of documents for one-term queries. Various interpretations of TF-IDF—based on binary independence retrieval, Poisson, information theory, and language modeling—are reviewed in Roelleke and Wang (2008).

8.3 Term Discrimination Value

Another statistical approach similar to TF-IDF is based on the notion of *term discrimination value*. Assume that we have a collection of documents and each document is characterized by a certain number of terms. We can think of each document as a point in the *document space*. The distance between any two documents is the distance between the corresponding points in the document space. If the distance is small, the documents are considered similar. In other words, the similarity between two documents is inversely proportional to the distance between the corresponding points in the document space. If documents are assigned similar terms, the corresponding points in the document space will be closer and the *density of the document space* is increased. On the other hand, if the documents are assigned different terms, the density of the document space is decreased.

We can approximate the weight of a term based on the type of change it produces in the document space when the term is assigned to all documents in the collection. The density change is quantified based on the increase or decrease in the average distance between the documents in the collection. Under this method, a term has a good discrimination value if it increases the average distance between the documents, or equivalently, terms with good discrimination value decrease the density of the document space. Typically, medium document frequency terms decrease the density and entail positive discrimination value; high document frequency terms increase the density and entail negative discrimination value; and low document frequency terms produce no change in the document density and their discrimination values are closer to zero. The reader should consult Salton (1988) for methods used for computing document space densities. A term weighting scheme which combines term frequency and discrimination value is defined as:

$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \cdot dv_t \tag{5}$$

where dv_t is the discrimination value of term t. This is similar to TF-IDF weighting scheme, but produces somewhat a different ranking compared to TF-IDF.

8.4 Document Length Normalization

Document length normalization is a technique to modify term weight as a function of document length. A long document may contain spurious occurrences of a term in different sections. That is, the context in which the term appears in one section of the document may not be related to the context of the same term appearing in a different section of the document. Furthermore, if we append a copy of the document d to itself, the relevance of the resulting longer document d' to a query is not greater than the relevance of d to the same query. Generally, longer documents score higher to a query compared to shorter documents.

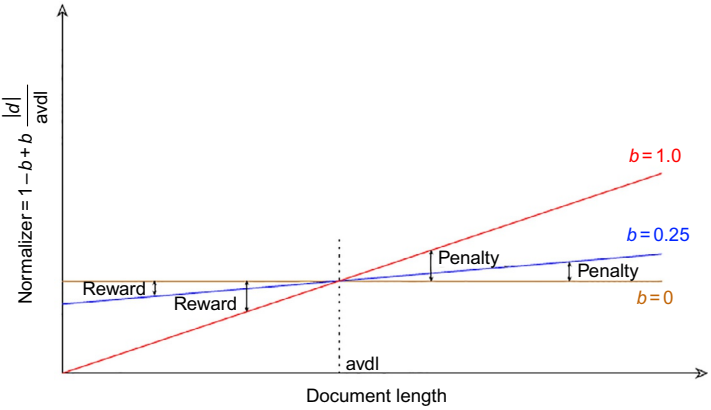

FIG. 6 Pivoted document length normalization.

The essence of document length normalization is to have a principled approach to reward short documents and penalize long documents. It essentially involves either increasing (for short documents) or decreasing (for short documents) the actual frequency occurrence of a term. The average document length of a collection, denoted *avdl*, is central to this approach. Some documents in the collection will be shorter than *avdl*, while others will be longer.

Fig. 6 illustrates *pivoted document length normalization*, where *avdl* is the pivot point. Documents whose length is less than the *avdl* are considered short documents, and are rewarded proportionately by increasing their relevance score through boosting the actual frequency of term occurrence. Likewise, documents whose length is greater than *avdl* are considered long documents, and are penalized by decreasing their relevance score. The reward/penalty increases linearly, and is dependent on how short/long a document is with reference to *avdl*. The parameter b in the normalizer expression $1-b+b\frac{|d|}{\text{avdl}}$ controls the reward/penalty. The value of b is in the range [0, 1].

Consider the normalizer expression $1-b+b\frac{|d|}{\text{avdl}}$. When $b = 0$, the expression evaluates to 1. This means that the term frequency is not altered, and there is neither penalty nor reward. The other extreme case occurs when $b = 1$. The expression evaluates to $\frac{|d|}{\text{avdl}}$, and its value is greater than 1 for documents whose lengths are greater than avdl. When we divide the term frequency with a value that is greater than 1, the term frequency diminishes, and reduces the relevance score. In contrast, when $b = 0$, the value of the expression $\frac{|d|}{\text{avdl}}$ for short documents will be less than 1. When the term frequency is divided by a value which is less than 1, the term frequency increases and results in boosting the contribution of the term to the relevance score. Thus, we can control the reward/penalty by choosing a value for b in the range [0, 1].

For example, when $b = 0.25$, the penalty/reward is relatively smaller compared to the penalty/reward when $b = 1$. In summary, reward/penalty is implemented by dividing the term weights $w_{t,d}$ (given by Eqs. 1 or 2) by the pivoted length normalizer value (Fig. 6).

8.5 BM25 Term Weighting

Consider the following function, where y is the term weight, x is the term frequency, and k is a free parameter whose value is a positive real number.

$$y = \frac{(k+1)x}{k+x} \tag{6}$$

Usually, k value is chosen from the range [1.2, 2.0]. The upper bound of y is controlled by k. In $\frac{(k+1)x}{k+x}$, the numerator will never be able to exceed the denominator. In other words, y is upper bounded by $(k + 1)$. The upper bound is useful to moderate the influence of terms. It also ensures that all the terms in the document have representation in computing the relevance. Lastly, the upper bound is critical to preventing spammers from influencing web search engines to effect unfair ranking of web pages through *keyword stuffing*. Note that the log transformation discussed in Section 8.1 does not have an upper bound.

Fig. 7 shows the shape of BM25 term weighting for values of $k = 3$, 20, 55. When $k = 3$, note that y is upper bounded by the horizontal line

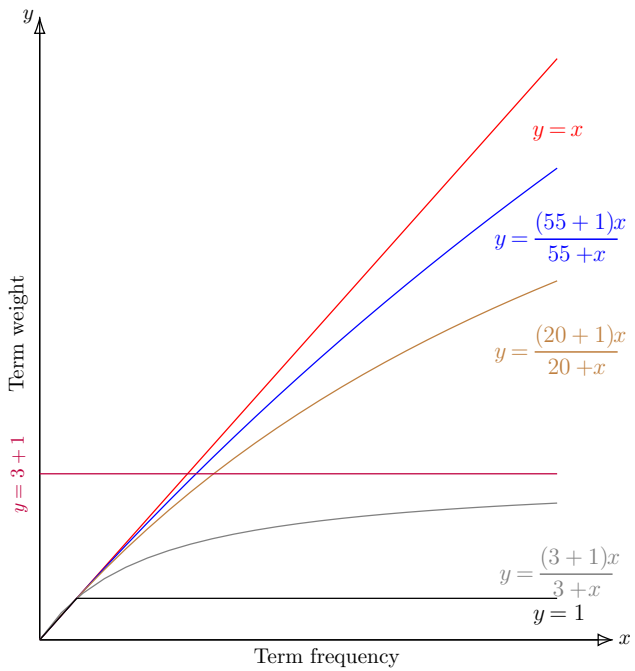

FIG. 7 BM25 term weighting.

$y = k + 1 = 3 + 1 = 4$. Also, note the horizontal line $y = 1$, which corresponds to binary term weighting—value 1 if the term is present one or more times, and 0 otherwise. What matters is whether or not the term is present in the document, and not on the frequency of occurrence of the term.

While $y = 1$ represents one end of the spectrum for term weighting, and $y = k$ (the 45° line) represents the other extreme—the linear transformation (i.e., term weight is equal to the raw frequency of the term). The BM25 term weighting is very flexible and the desired term frequency transformation is controlled by choosing an appropriate value for k.

9 VECTOR SPACE IR MODEL

The Vector Space Model (VSM) is based on the notion of *similarity*. The model assumes that the relevance of a document to query is roughly equal to the document-query similarity. Both the documents and queries are represented using the *bag-of-words* model. For a document collection, we first determine a set of terms (i.e., vocabulary) and order the terms. Next, documents are represented as n-dimensional vectors, where each dimension corresponds to a term. Terms are weighted using TF-IDF or BM25 methods. Queries are also represented as n-dimensional vectors.

Fig. 8 shows a three-dimensional vector space spanned by three terms—Data, Analytics, and Learning. There are three documents—D_1, D_2, and

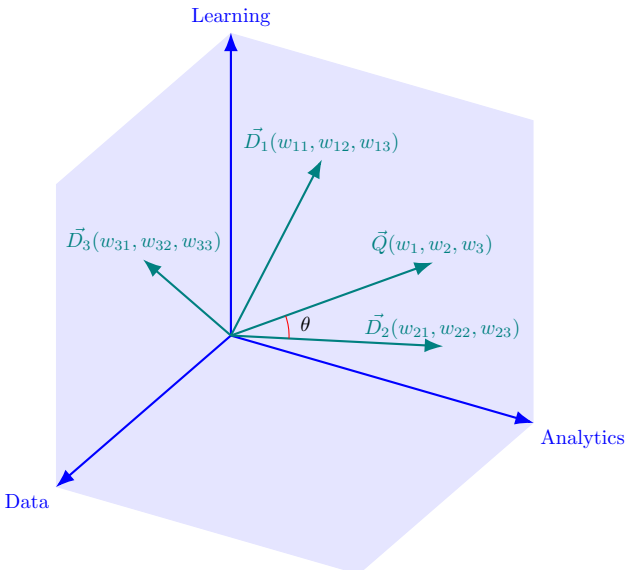

FIG. 8 The vector space model.

D_3—and corresponding to each document is a three-dimensional vector, denoted $\vec{D_1}$, $\vec{D_2}$, and $\vec{D_3}$. The term weights of $\vec{D_1}$ are (w_{11}, w_{12}, w_{13}). The term weights determine the document's orientation/placement in the vector space. Like documents, queries are also represented as three-dimensional vectors (query $\vec{Q}\,(w_1, w_2, w_3)$ in Fig. 8), and the query terms can also be weighted. Lastly, the document and query vectors are normalized to transform them into *unit vectors*. A vector is turned into a unit vector by dividing the vector by its magnitude. The unit vector of $\vec{D_1}$ (w_{11}, w_{12}, w_{13}), denoted $\vec{d_1}$, is defined as:

$$\vec{d_1} = \frac{\vec{D_1}}{\sqrt{w_{11}^2 + w_{12}^2 + w_{13}^2}} \tag{7}$$

The similarity between document D_2 and query Q is given by the *dot product* between the corresponding unit vectors:

$$\text{sim}(D_2, Q) = \vec{d_2} \cdot \vec{q} \tag{8}$$

The dot product of $\vec{d_2} \cdot \vec{q}$ is the sum of the product of the corresponding vector components—$w_{21} \cdot w_1 + w_{22} \cdot w_2 + w_{23} \cdot w_3$. sim($D_2$, Q) is also equal to the cosine of the angle between $\vec{d_2}$ and $\vec{q}$.

To rank documents in a collection with respect to a query, we compute the cosine of the angle between the query and each document in the collection. Thus, we have a Relevance Status Value (RSV) for each document-query pair. We rank documents in the decreasing order of RSVs.

The VSM is really a framework since it does not prescribe how to accomplish three primary tasks—defining the dimensions of the vector space, term weighting for documents and queries, and similarity measure between document and query vectors. Using vocabulary terms as the dimensions of the vector space, TF-IDF term weighting, and cosine similarity measure discussed above is one instantiation of the model. We could easily replace TF-IDF term weighting with BM25. Also, we can replace cosine similarity measure with something else. Therefore, VSM is heuristic in nature, enables exploration and experimentation, and provides a good baseline to assess other retrieval models. The model assumes that the terms that span the vector space are orthogonal to each other. However, this assumption is not always true in reality. Also, the notion of similarity does not necessarily translate into relevance.

Interested readers should consult Raghavan and Wong (1986) for a critical analysis of VSM. Wong et al. (1985) proposed a Generalized Vector Spaces Model (GVSM), which overcomes the term orthogonality assumption. Turney and Pantel (2010) provides a survey of VSMs for semantic processing of text.

10 PROBABILISTIC IR MODELS

Probabilistic retrieval models are a family of models. In contrast with the vector space model, probabilistic IR models rank documents based on the *probability that the document is relevant* to the query. We assume that both documents and queries are *observations* of the associated *random variables* D and Q. We also assume that relevance is a binary random variable R, which takes only two values—r (relevant) or $\overline{r}$ (nonrelevant). The relevance of a document to a query is formulated in terms of the *probability* $p(R \mid D, Q)$. This is the basis of *Probability Ranking Principle* (PRP) (van Rijsbergen, 1979, chap. 6, pp. 113–114):

> *If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.*

Using the Bayes theorem and algebraic manipulation, $p(R \mid D, Q)$ can be expressed as:

$$p(R \mid D,Q) = \log \frac{p(D \mid Q,r)}{p(D \mid Q,\overline{r})} \tag{9}$$

Eq. (9) is at the core of probabilistic IR models. Various methods are used to estimate the relevance of a document to a query.

10.1 Binary Independence Model

One approach to estimating the value of Eq. (9) gives a model known as the Binary Independence Model (BIM). This involves making certain assumptions (Büttcher et al., 2010). First, only the presence or absence of terms in documents and queries are considered. Random variable D is redefined as a vector of binary random variables $\{D_1, D_2, \ldots\}$, with one dimension for each vocabulary term. $D_i = 1$ indicates that the term is present, and $D_i = 0$ denotes the absence of the term. Query Q is also represented as a vector of binary random variables $\{Q_1, Q_2, \ldots\}$.

The BIM makes two strong assumptions. Given that a document is relevant, the presence/absence of a term does not depend on the presence/absence of any other term. Second, the presence of a term in a document depends on relevance only when the term is present in the query. This assumption links the appearance of a term in a query to the probability of its appearance in a relevant document. Simplification of Eq. (9) under the above assumptions results in the following:

$$\mathrm{RSV}_d = \sum_{t \in (q \cap d)} \log \frac{p(D_t = 1|r)\ p(D_t = 0|\overline{r})}{p(D_t = 1|\overline{r})\ p(D_t = 0|r)} \tag{10}$$

In the above equation, the relevance of a document d to query q is the summation of contributions by each of the terms that are common to d and q. $p(D_t = 1 \,|\, r)$ denotes the probability that the term t occurs in a relevant document, and $p(D_t = 0 \,|\, \overline{r})$ indicates the probability that the term t does not occur in a nonrelevant document. Likewise, $p(D_t = 1 \,|\, \overline{r})$ is the probability the term t occurs in a nonrelevant document, and $p(D_t = 0 \,|\, r)$ is the probability that the term t does not occur in a relevant document.

Let $p_t = p(D_t = 1 \,|\, r)$ and $\overline{p}_t = p(D_t = 1 \,|\, \overline{r})$. $p(D_t = 0 \,|\, r)$ can be written as $1 - p(D_t = 1 \,|\, r) = 1 - p_t$. Likewise, $p(D_t = 0 \,|\, \overline{r}) = 1 - p(D_t = 1 \,|\, \overline{r}) = 1 - \overline{p}_t$. Then, Eq. (10) can be rewritten as:

$$\mathrm{RSV}_d = \sum_{t \in (q \cap d)} \log \frac{p_t\,(1 - \overline{p}_t)}{\overline{p}_t\,(1 - p_t)} \tag{11}$$

We illustrate the BIM with an example. Shown in Table 13 are documents, and occurrence of terms—learning, analytics, evaluation—in the documents (1 means term occurs, 0 indicates absence of the term). The last row of the table indicates whether or not the document is relevant to the query q (N = nonrelevant, R = relevant). We want to rank the relevance of documents to the query q.

Solution: There are three terms: learning, analytics, evaluation. We first compute $p_1, p_2, p_3, \overline{p}_1, \overline{p}_2, \overline{p}_3$ using the formulas: $p_i = \frac{r_i}{r}$ and $\overline{p}_i = \frac{f_i - r_i}{f - r}$, where f is the number of documents, r is the number of documents judged relevant, f_i is the number of documents in which term i is present, and r_i is the number of documents judged relevant in which the term i occurs.

Using the data in Table 13, we have $f = 10, r = 6, f_1 = 6, f_2 = 7, f_3 = 7, r_1 = 3, r_2 = 4, r_3 = 5$. Next, compute p_i, $1 \le i \le 3$: $p_1 = \frac{3}{6} = 1/2, p_2 = \frac{4}{6} = 2/3, p_3 = \frac{5}{6} = 5/6$. Likewise, compute $\overline{p}_i, 1 \le i \le 3$: $\overline{p}_1 = \frac{6-3}{10-6} = 3/4, \overline{p}_2 = \frac{7-4}{10-6} = 3/4, \overline{p}_3 = \frac{7-5}{10-6} = 1/2$.

TABLE 13 Data for Ranking by BIM

docID	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
learning	1	0	1	0	0	1	1	0	1	1
analytics	1	1	0	1	0	1	1	1	1	0
evaluation	1	0	1	1	1	0	1	0	1	1
$r(q, d_i)$	R	R	N	R	R	N	R	N	N	R

The base of the logarithm does not change the rank ordering of documents. In the following, we use the natural logarithm, ln. For each term t, we compute its contribution to relevance: $w_t = \ln \frac{p_t\,(1-\overline{p}_t)}{\overline{p}_t\,(1-p_t)}$; $w_1 = \ln \left[\frac{\frac{1}{2}(1-\frac{3}{4})}{\frac{3}{4}(1-\frac{1}{2})}\right] = -1.099$. Likewise, $w_2 = -0.405$, $w_3 = 1.609$.

Lastly, we compute Relevance Status Value (RSV) for each document as shown below.

Document	RSV Components	Component Values	RSV Value
d_1	$w_1 + w_2 + w_3$	− 1.0986 − 0.4055 + 1.6094	0.1053
d_2	w_2	− 0.4055	− 0.4055
d_3	$w_1 + w_3$	− 1.0986 + 1.6094	0.5108
d_4	$w_2 + w_3$	− 0.4055 + 1.6094	1.2039
d_5	w_3	1.6094	1.6094
d_6	$w_1 + w_2$	− 1.0986 − 0.4055	− 1.5041
d_7	$w_1 + w_2 + w_3$	− 1.0986 − 0.4055 + 1.6094	0.1053
d_8	w_2	− 0.4055	− 0.4055
d_9	$w_1 + w_2 + w_3$	− 1.0986 − 0.4055 + 1.6094	0.1053
d_{10}	$w_1 + w_3$	− 1.0986 + 1.6094	0.5108

The rank order—most relevant to least relevant—of the documents is: d_5, d_4, $\{d_3, d_{10}\}$, $\{d_1, d_7, d_9\}$, $\{d_2, d_8\}$, d_6. The notation $\{d_3, d_{10}\}$ means that the documents d_3 and d_{10} are at the same rank.

10.2 Okapi BM25 Model

Okapi Best Match 25 (BM25) model evolved from a series of previous models—BM1, BM11, BM12, and BM 15 (Robertson, 1997). BM25 builds on Eq. (9) and uses BM25 term weighting (Section 8.5), inverse document frequency (Section 8.2), and document length normalization (Section 8.4). BM25 combines the features of both BM11 and BM15. Relevance Status Value (RSV) of a document d with respect to a query q is given by:

$$\mathrm{RSV}_d = \sum_{t\in(q\cap d)} \left[\log \left(\frac{(r_t+0.5)/(N_r-r_t+0.5)}{(\mathrm{df}_t-r_t+0.5)/(N-N_r-\mathrm{df}_t+r_t+0.5)} \right) \cdot \left(\frac{\mathrm{tf}_{t,d}+k_1\cdot \mathrm{tf}_{t,d}}{\mathrm{tf}_{t,d}+k_1\left((1-b)+b\cdot \frac{\mathrm{dl}}{\mathrm{avdl}}\right)} \right) \cdot \left(\frac{\mathrm{tf}_{t,q}+k_2\cdot \mathrm{tf}_{t,q}}{\mathrm{tf}_{t,q}+k_2} \right) \right] \tag{12}$$

where

- r_t the number of relevant documents (in the collection) in which term t occurs
- df_t is the document frequency of term t
- N is the total number of documents in the collection
- N_r is the number of documents relevant to the query q
- $\text{tf}_{t,d}$ is the number of times the term t occurs in document d
- $\text{tf}_{t,q}$ is the number of times the term t occurs in the query q
- dl is document length
- avdl is the average document length—the average length across all documents in the collection
- k_1, k_2, and b are empirically set parameters. Typical values are: $k_1 = 1.2$, $0 \leq k_2 \leq 1000$, and $b = 0.75$

Let us analyze Eq. (12). Each term t that is common to both the document d and query q contributes to relevance. Relevance is cumulative contribution from such terms. Relevance contribution from each term draws from three factors: IDF-like score, term frequency (within a document) upper bounded and length normalized, and query term frequency. These three factors correspond to the three expressions (left to right) in Eq. (12), respectively. Now consider the first factor:

$$\log\left(\frac{(r_t+0.5)/(N_r-r_t+0.5)}{(\text{df}_t-r_t+0.5)/(N-N_r-\text{df}_t+r_t+0.5)}\right) \tag{13}$$

The numerator is the ratio of the number of relevant documents (in the collection) in which term t occurs to the number of relevant documents in which the term does not occur. We add 0.5 to smooth the counts. The denominator denotes the same ratio for nonrelevant documents. In other words, the numerator is the likelihood ratio which is the *evidence of relevance* the term provides. Likewise, the denominator provides the *evidence of nonrelevance*. Next, consider the part of the second factor shown below:

$$\frac{\text{tf}_{t,d}+k_1\cdot\text{tf}_{t,d}}{\text{tf}_{t,d}+k_1} \tag{14}$$

This equation is similar to Eq. (6), where $k_1 = k$ and $\text{tf}_{t,\,d} = x$. This is BM25 term weighting, which is discussed in Section 8.5, and k_1 is the empirically set parameter for upper bounding the *within document term frequency*. The expression $(1-b)+b\cdot\frac{\text{dl}}{\text{avdl}}$ in the denominator performs pivoted length normalization which we discussed in Section 8.4. The value of b controls the desired significance of length normalization in relevance computation. When $b = 0$, no length normalization is applied; and $b = 1$ effects length normalization to the fullest; other values of b in the range (0, 1) are chosen to achieve length normalization effect between these two extremes. Documents that are longer than the average document length make the denominator larger, and thus reduce the relevance contribution of a term. The third factor

is the frequency of the term in the query, which is upper bounded just like the corresponding term in the document (k_2 is the empirically set parameter for upper bounding the query term frequency).

We illustrate BM25 algorithm with an example. Assume that a collection has 50,000 documents. The terms *information* and *retrieval* occur in 40,000 and 30,000 documents, respectively. Consider a document d_1 in which *information* and *retrieval* occur 40 and 30 times, respectively. Assume the ratio of length of d_1 to average document length is 0.9 (i.e., $\frac{\text{dl}}{\text{avg(dl)}} = 0.9$). Given that the query q is *information retrieval*, compute RSV for (d_1, q) pair using the BM25 algorithm.

Following are the solution steps:

- Both the *query terms—information* (denoted by t_1) and *retrieval* (denoted by t_2)—occur in document d_1. The term frequencies in d_1 and q are: $\text{tf}_{t_1,d_1} = 40$, $\text{tf}_{t_2,d_1} = 30$, $\text{tf}_{t_1,q} = 1$, and $\text{tf}_{t_2,q} = 1$.
- The number of documents in the collection (N) is 50,000. The document frequencies of the terms t_1 and t_2 are: $\text{df}_{t_1} = 40{,}000$ and $\text{df}_{t_2} = 30{,}000$. Also, dl/avdl is given as 0.9.
- Since no relevance judgments data is given, we assume $N_r = 0$, $r_{t_1} = 0$, and $r_{t_2} = 0$. Also, $k_1 = 1.2$, $k_2 = 100$, and $b = 0.75$.
- Both the terms—*information* (t_1) and *retrieval* (t_2)—contribute to RSV. Instantiation of Eq. (12) for t_1 is:

$$\log\left(\frac{(r_{t_1}+0.5)/(N_r-r_{t_1}+0.5)}{(\text{df}_{t_1}-r_{t_1}+0.5)/(N-N_r-\text{df}_{t_1}+r_{t_1}+0.5)}\right)\cdot \left(\frac{\text{tf}_{t_1,d_1}+k_1\cdot\text{tf}_{t_1,d_1}}{\text{tf}_{t_1,d_1}+k_1\left((1-b)+b\cdot\frac{\text{dl}}{\text{avdl}}\right)}\right)\cdot\left(\frac{\text{tf}_{t_1,q}+k_2\cdot\text{tf}_{t_1,q}}{\text{tf}_{t_1,q}+k_2}\right) \tag{15}$$

- Substitution of given values into Eq. (15) yields:

$$\log\left(\frac{(0+0.5)/(0-0+0.5)}{(40{,}000-0+0.5)/(50{,}000-0-40{,}000+0+0.5)}\right)\cdot \left(\frac{40+1.2\cdot 40}{40+1.2((1-0.75)+0.75\cdot 0.9)}\right)\cdot\left(\frac{1+100\cdot 1}{1+100}\right) \tag{16}$$

- Eq. (16) evaluates to $(-0.6020)\cdot(2.141)\cdot(1) = -1.289$
- We repeat the above process for t_2, which gives a value of -0.3735.
- RSV of $(d_1, q) = -1.289 - 0.3735 = -1.6625$

11 LANGUAGE MODEL-BASED IR

First we discuss statistical language modeling, followed by the query likelihood model.

11.1 Statistical Language Modeling

A language model is a probability distribution over all the strings in the language. In other words, the language model assigns a probability to every word/term in the language (Gudivada et al., 2015). Therefore, the primary task in language modeling is to construct a probability distribution function. Once the probabilities for individual words are known, we can compute probabilities for any phrase or sentence in the language. The higher the probability of a sentence, the more likely the sentence is a valid construction in the language. For example, assume that the probabilities associated with the words—information, retrieval, models—are 0.1, 0.15, and 0.05, respectively. Then the probability of the phrase *information retrieval models* is $(0.1) \cdot (0.15) \cdot (0.05) = 0.00075$.

Let $\mathcal{V}$ be a finite vocabulary and $\mathcal{V}^*$ be the set of strings in the language defined by $\mathcal{V}$. For example, $\mathcal{V} = \{\text{machine, learning, information, retrieval}\}$ and $\mathcal{V}^* = \{\text{machine, learning, information, retrieval, machine learning, information retrieval, machine information, machine learning information, } \ldots\}$. The probability distribution function p for this language is one that satisfies:

$$\sum_{x \in \mathcal{V}^*} p(x) = 1, \text{ and } p(x) \geq 0 \text{ for all } x \in \mathcal{V}^*$$

Constructing a sentence by choosing one word/term at a time is viewed as a stochastic process. Let $x_{[1:n]}$ denote a sentence of length n, where x_1 is the first word, x_2 is the second word, and so on. Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables, each of which takes a value randomly from the set $\mathcal{V}^*$. Also, let the words in the sentence $x_{[1:n]}$ correspond to the random variables $X_1, X_2, \ldots, X_n$. We want to model $p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$.

Let $q(x_1)$ denote the probability of occurrence of the word x_1, and $q(x_i|x_{[1:i-1]})$ denote the probability of occurrence of the ith word x_i given that the previous $i - 1$ words are $x_{[1:i-1]}$. The joint probability $p(x_1 x_2 \ldots x_n)$ is computed using the chain rule as:

$$p(x_1 x_2 \ldots x_n) = q(x_1) \prod_{i=2}^{n} q(x_i|x_{[1:i-1]}) \tag{17}$$

For example, $p(\text{information retrieval models}) = q(\text{information}) \times q(\text{retrieval} \mid \text{retrieval}) \times q(\text{models} \mid \text{information, retrieval})$. We estimate the probability distribution of q using counts of phrase occurrences in training data. Using the Markov assumption, Eq. (17) is rewritten as:

$$p(x_1 x_2 \ldots x_n) \approx \prod_{i=1}^{n} p(x_i|x_{i-k}\ x_{(i-k)+1}\ x_{(i-k)+2} \ldots x_{i-1}),\ 1 \leq k < i \tag{18}$$

When $k = 1$, the values of the current state are dependent on just the immediately preceding state (aka *first-order Markov process*). Likewise,

when $k = 2$, the values of the current state are dependent on just the two immediately preceding states (aka *second-order Markov process*). We approximate each component in the product of Eq. (18) as:

$$p(x_i \mid x_1 x_2 \ldots x_{i-1}) \approx q(x_i \mid x_{i-k}\ x_{(i-k)+1}\ x_{(i-k)+2} \ldots x_{i-1}),\ 1 \leq k < i \tag{19}$$

Different values of k in Eq. (19) result in different language models. For example, if $k = 0$, we get the *unigram language model*:

$$p(x_1 x_2 \ldots x_n) \approx \prod_{i=1}^{n} q(x_i)$$

In the unigram model, the probability of observing a given word does not depend on the context in which the word occurs. Similarly, by setting $k = 1$ and $k = 2$, we get the *bigram* and *trigram* language models:

$$p(x_1 x_2 \ldots x_n) \approx \prod_{i=1}^{n} q(x_i \mid x_{i-1})$$

$$p(x_1 x_2 \ldots x_n) \approx \prod_{i=1}^{n} q(x_i \mid x_{i-2} x_{i-1})$$

In the bigram model, the probability of observing a given word depends on just the immediately preceding word; in the trigram model, it depends on the *two* immediately preceding words. Given the sparsity of data in language modeling, q is *smoothed* using linear interpolation and discounting methods. Linear interpolation methods estimate parameter values by a linear combination of maximum likelihood estimates from the unigram, bigram, and trigram models as:

$$q(x_i \mid x_{i-2} x_{i-1}) = \lambda_1\ q_{ml}(x_i \mid x_{i-2}, x_{i-1}) + \lambda_2\ q_{ml}(x_i \mid x_{i-1}) + \lambda_3\ q_{ml}(x_i)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, $\lambda_i \geq 0$, $1 \leq i \leq 3$, and q_{ml} denotes maximum likelihood estimation of parameters. Values of λ_1, λ_2, and λ_3 are determined by optimizing a function whose domain is (λ_1, λ_2, λ_3).

11.2 The Query Likelihood Model

Language modeling is a formal approach to IR, and there are many approaches to realizing it. The *query likelihood model* is one of the earlier approaches. We rank documents based on $p(d \mid q)$—interpreted as the likelihood that the document d is relevant to the query. Using Bayes rule, we have the following:

$$p(d \mid q) = \frac{p(q \mid d) p(d)}{p(q)} \tag{20}$$

We can discard $p(q)$ as it is the same for all documents—all queries are equally likely. Also, the prior probability $p(d)$ is often considered as the same for all documents. How do we compute $p(q|d)$? In language model-based approaches to IR, there is a language model associated with each document (Lafferty and Zhai, 2001; Ponte and Croft, 1998). Let M_{d_1} and M_{d_2} be the language models associated with the documents d_1 and d_2. A query is viewed as a string generated by the language model associated with a document. Therefore, we model the query generation process as the probability that a query q would be observed as a random sample from a document model. Let $p(q|M_{d_1})$ and $p(q|M_{d_2})$ be the probabilities that q has been generated from the language models corresponding to d_1 and d_2, respectively. If $p(q|M_{d_1}) > p(q|M_{d_2})$, d_1 is considered more relevant to q than d_2. Assuming conditional independence, we can estimate $p(q|M_{d_1})$ as:

$$p(q|M_{d_1}) = p(t_1, t_2, \ldots, t_{|q|}|M_{d_1}) = \prod_{1 \le k \le |q|} p(t_k|M_{d_1}) \tag{21}$$

where $|q|$ is the number of terms in the query. To account for multiple occurrences of terms in the query, the above equation can be rewritten as:

$$p(q|M_{d_1}) = \prod_{\text{distinct term } t \text{ in } q} p(t|M_{d_1})^{\text{tf}_{t,q}} \tag{22}$$

where $\text{tf}_{t,q}$ is the number of occurrences of term t in query q. We can compute $p(q|M_{d_1})$ if we know how to estimate $p(t|M_{d_1})$. One approach to estimation is *maximum likelihood* as:

$$\hat{p}(t|M_{d_1}) = \frac{\text{tf}_{t,d_1}}{|d_1|} \tag{23}$$

In Eq. (22), a single term with $p(t|M_{d_1})$ will make $\prod_{1 \le k \le |q|} p(t_k|M_{d_1})$ zero. Therefore, we *smooth* estimates to avoid this problem. Jelinek-Mercer smoothing is one such approach. It uses the collection model M_c, which is the document resulting from concatenating all documents in the collection into one large document. Next, *collection frequency* of a term t is derived from M_c:

$$\hat{p}(t \mid M_c) = \frac{\text{cf}_t}{T} \tag{24}$$

where T is the number of tokens in the collection—$T = \sum_t \text{cf}_t$

Jelinek-Mercer smoothing combines the probability of a term from the document with the general collection frequency of the term as:

$$\hat{p}(t|d_1) = \lambda\, \hat{p}(t|M_{d_1}) + (1 - \lambda)\, \hat{p}(t|M_c) \tag{25}$$

where $0 \leq \lambda \leq 1$. This is a *linear interpolation* language model, and choosing the right value for λ is critical to the performance of the model. A high value for λ entails a *conjunctive-like* search and tends to retrieve documents that contain all the query terms. On the other hand, a low value for λ entails a more *disjunctive* search and is suitable for long queries.

In summary, the query likelihood model assumes that a user has a document in mind and generates the query from this document. The following equation represents the probability that the document the user had in mind is d_1:

$$p(q|d_1) \propto \prod_{1 \leq k \leq |q|} [\lambda \, \hat{p}(t_k|M_{d_1}) + (1-\lambda)\, \hat{p}(t_k|M_c)] \tag{26}$$

We illustrate the *query likelihood* language model with an example. Let d_1 = The woman who opened fire at YouTube headquarters in Northern California may have been a disgruntled user of the video-sharing site, d_2 = Authorities are investigating a website that appears to show the same woman accusing YouTube of restricting her videos, and q = YouTube fire. We want to rank d_1 and d_2 with respect to query q using Jelinek-Mercer smoothing with $\lambda = 0.5$.

Solution: We perform text normalization by converting tokens to lowercase and removing stopwords. After normalization, d_1 = woman opened fire youtube headquarters northern california disgruntled user video sharing site, and d_2 = authorities investigating website appears show same woman accusing youtube restricting videos. The number of tokens in d_1 and d_2 after normalization are 12 and 11 (i.e., $|d_1| = 12$ and $|d_2| = 11$).

The collection model M_c is the concatenation of d_1 and d_2. Therefore, M_c = woman opened fire youtube headquarters northern california disgruntled user video sharing site authorities investigating website appears show same woman accusing youtube restricting videos. Total number of tokens in M_c = 23 (i.e., $T = \sum_t \mathrm{cf}_t = 23$).

The query terms after text normalization are: t_1 = youtube and t_2 = fire. The frequency of each of the query terms in the query is 1 (i.e., $\mathrm{tf}_{(\text{youtube},\, q)} = 1$, and $\mathrm{tf}_{(\text{fire},\, q)} = 1$). Since query has two terms, we need to estimate only the following four parameters using maximum likelihood method: $\hat{p}(t_1|M_{d_1})$, $\hat{p}(t_2|M_{d_1})$, $\hat{p}(t_1|M_{d_2})$, and $\hat{p}(t_2|M_{d_2})$. The general formula for parameter estimation using maximum likelihood is:

$$\hat{p}(t \mid M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

For example, we calculate $\hat{p}(\text{youtube} \mid M_{d_1})$ as

$$\hat{p}(\text{youtube} \mid M_{d_1}) = \frac{\text{number of times the term youtube occurs in } d_1}{\text{length of } d_1} = \frac{1}{12}$$

Likewise, we calculate $\hat{p}(\text{fire} \mid M_{d_1}) = \frac{1}{12}$, $\hat{p}(\text{youtube} \mid M_{d_2}) = \frac{1}{11}$, $\hat{p}(\text{fire} \mid M_{d_2}) = \frac{0}{11}$, $\hat{p}(\text{youtube} \mid M_c) = \frac{2}{23}$, and $\hat{p}(\text{fire} \mid M_c) = \frac{1}{23}$.

Next, we compute $\hat{p}(t|d)$ using Jelinek-Mercer smoothing.

$$\begin{aligned}
p(\text{youtube}|d_1) &= \lambda\,\hat{p}(\text{youtube}|M_{d_1}) + (1-\lambda)\,\hat{p}(\text{youtube}|M_c) \\
&= 0.5 \cdot \frac{1}{12} + 0.5 \cdot \frac{2}{23} = 0.0851 \\
p(\text{fire}|d_1) &= \lambda\,\hat{p}(\text{fire}|M_{d_1}) + (1-\lambda)\,\hat{p}(\text{fire}|M_c) \\
&= 0.5 \cdot \frac{1}{12} + 0.5 \cdot \frac{1}{23} = 0.0634 \\
p(\text{youtube}|d_2) &= \lambda\,\hat{p}(\text{youtube}|M_{d_2}) + (1-\lambda)\,\hat{p}(\text{youtube}|M_c) \\
&= 0.5 \cdot \frac{1}{11} + 0.5 \cdot \frac{2}{23} = 0.0889 \\
p(\text{fire}|d_2) &= \lambda\,\hat{p}(\text{fire}|M_{d_2}) + (1-\lambda)\,\hat{p}(\text{fire}|M_c) \\
&= 0.5 \cdot \frac{0}{11} + 0.5 \cdot \frac{1}{23} = 0.0217
\end{aligned}$$

Now we have all the information to compute $\hat{p}(q|M_{d_1})$ and $\hat{p}(q|M_{d_2})$.

$$\begin{aligned}
p(q|M_{d_1}) &= \prod_{\text{distinct term } t \text{ in } q} p(t|M_{d_1})^{\text{tf}_{t,q}} \\
&= 0.0851^1 \cdot 0.0634^1 = 0.005398 \\
p(q|M_{d_2}) &= \prod_{\text{distinct term } t \text{ in } q} p(t|M_{d_2})^{\text{tf}_{t,q}} \\
&= 0.0889^1 \cdot 0.0217^1 = 0.001933
\end{aligned}$$

Since $p(q|M_{d_1}) > p(q|M_{d_2})$, rank order is d_1 followed by d_2.

Extensions to the original language modeling approach to IR include *document likelihood model*, which is based on the idea of a query language model M_q generating the document. This way of modeling relevance enables incorporating pseudo-relevance feedback (discussed in Section 13) into language models. The initial query is executed and the top 10–25 documents are considered relevant to the query. The query is expanded with the terms from the relevant documents and the query language model M_q is updated (Zhai and Lafferty, 2001). Estimating the probabilities of terms in the relevant documents for a query by using the query alone (no training data) is discussed in Lavrenko and Croft (2001). Lavrenko and Croft demonstrate that their approach produces highly accurate relevance models, which also address the notions of synonymy and polysemy. This model is indeed a *document likelihood model*.

Another extension to the original language model is based on quantifying the difference between the *document likelihood* and *query likelihood* models. In other words, two probability distributions are compared. Several measures are available for quantifying the dissimilarity between probability distributions. One such measure is called *Kullback-Leibler Divergence*

(KL Divergence). Let $p(x)$ and $q(x)$ be two probability distributions over $\mathcal{X}$. KL divergence (aka *relative entropy*) $D(p \parallel q)$ is defined as:

$$D(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{q(x)}$$

KL divergence is applicable to any model—$p(x)$ and $q(x)$ may represent the models of a document and query (i.e., $D_{KL}(M_d \parallel M_q)$), or models of a document and corpus (i.e., $D_{KL}(M_d \parallel M_c)$). KL divergence is not symmetric and is more appropriate when there is a natural asymmetry, for example, document to corpus. *Mutual information* (MI) is an alternative to KL divergence for symmetric situations. The MI of two random variables is a measure of the mutual dependence between the two variables. MI between two random variables X and Y, denoted $I(X, Y)$, is defined as:

$$I(X,Y) = D_{KL}[\, p(X,Y) \parallel p(X)\, p(Y)\,]$$

KL divergence can be used to measure the relevance of a document to a query as:

$$RSV(d,q) = D_{KL}(M_d \parallel M_q) = \sum_{t \in \mathcal{V}} p(t|M_q) \ln \frac{p(t|M_q)}{p(t|M_d)}$$

KL divergence retrieval model surpasses both *query likelihood* and *document likelihood* models in retrieval performance (Lafferty and Zhai, 2001). It generalizes the query likelihood model and also incorporates pseudo-relevance feedback. In summary, relative to traditional IR models such as the Vector Space and classical probabilistic models, language model-based approaches place emphasis on parameter estimation and provide formal guidance on term weighting.

12 EVALUATING IR SYSTEMS

Unlike SQL and Boolean queries, free text queries are not formal representations of users' needs. *Retrieval effectiveness* is a measure of how well the documents retrieved by a system meet users' needs. The process of determining the retrieval effectiveness for a given query is referred to as *effectiveness evaluation*. Direct human evaluation of retrieval effectiveness is too expensive, tedious, and slow. Retrieval effectiveness is influenced by too many parameters including term weighting schemes, document length normalization formula, the number of top documents used for parameter estimation in probabilistic retrieval, and approaches used for relevance feedback. Therefore, automated approaches to retrieval effectiveness evaluation are imperative.

The general approach to effectiveness evaluation is comprised of the following steps: (a) User has an *information need* and expresses this need as a *query*, (b) An IR system runs the query against a document corpus, and

returns a *ranked list* of documents, and (c) Effectiveness is quantified in terms of how well the ranked list fulfills user's information need.

Two simplifying assumptions are made for effectiveness evaluation: retrieval is *ad hoc*, and effectiveness is based on relevance. Ad hoc retrieval is characterized by the facts that a query is constructed only once with no opportunity for refinement, the system has no prior knowledge about the user, and the system has no prior knowledge about the behavior of other users for this query. Effectiveness evaluation should consider *relevance to user's information need* rather than relevance to user query. Furthermore, relevance is binary—a document is either wholly relevant or nonrelevant, and relevance of a document in the ranked list is independent of relevance of other documents in the list. In addition to these assumptions, we also need a reusable test collection, which consists of three components: a corpus; a set of queries to run against the corpus; and for each document-query pair, a human judgment about the document's relevance to the information need underlying the query. Queries are often augmented by detailed description of the information need. Such augmented queries are referred to as *topics*.

12.1 Precision and Recall for Unranked Retrieval

One approach to measuring effectiveness of an IR system is in terms of precision and recall (Harman, 2011). *Precision* is the fraction of the retrieved documents that are relevant to the query. For example, if an IR system retrieved 10 documents to a query and only 3 of these documents are relevant, then precision is $(3/10) \times 100 = 30\%$. *Recall* refers to the fraction of relevant documents that are retrieved. As continuation of the precision example, assume that there are 5 relevant documents in the collection for the query. The system has retrieved 3 of these 5 relevant documents. Therefore, recall is $(3/5) \times 100 = 60\%$. In other words, precision indicates the percent of relevant documents in the retrieved set, whereas recall denotes the percent of relevant documents in the retrieved set (Fig. 9).

Another equivalent way to define precision and recall is in terms of true/false positives and true/false negatives as shown in Table 14.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision and recall oppose each other and are inversely related. For example, we can increase *recall* by returning more documents. In the extreme case, an IR system will achieve a 100% recall by returning all the documents in a collection. Also, a system can achieve a high *precision* for a very low *recall*. The recall is a nondecreasing function of the number of documents retrieved.

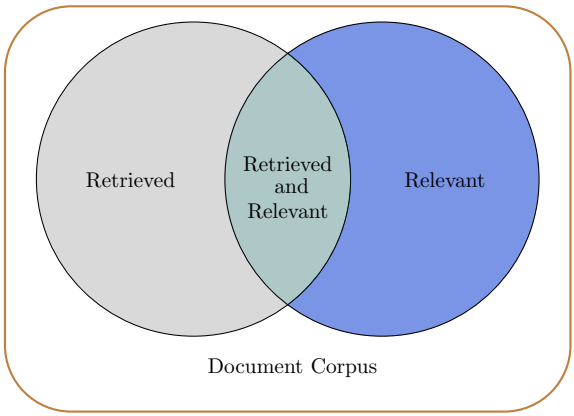

FIG. 9 Precision and recall as measures of an IR system effectiveness.

TABLE 14 Defining Precision and Recall in Terms of True/False Positives and True/False Negatives

	Relevant	Nonrelevant
Retrieved	True Positives (TP)	False Positives (FP)
Not Retrieved	False Negatives (FN)	True Negatives (TN)

12.2 The *F*-measure

The *F-measure* combines both precision and recall into one measure, and is defined as:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}, \text{ where } \beta^2 = \frac{1-\alpha}{\alpha}, \alpha \in [0,1], \text{ and } \beta^2 \in [0, \infty]$$

We have a balanced F (aka as F_1), when $\beta = 1$ or $\alpha = 0.5$. It is the harmonic mean of precision and recall:

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)$$

Harmonic mean is always less than either the arithmetic or geometric mean, and often close to the minimum of the two numbers—precision and recall. In general, when the values of two numbers differ greatly, the harmonic mean is closer to their minimum than to the arithmetic mean.

12.3 Retrieval Effectiveness for Ranked Retrieval

The definition of precision and recall measures assumes that the document collection is partitioned into two sets with respect to a query—the retrieved set, and the rest of the documents (not retrieved set) (Croft et al., 2010). However, most IR models produce *ranked results* as shown in Fig. 10. Consider the IR System 1 ranking in the figure. The rank of the leftmost document is 1, and the rank numbers increase as we go right. Higher the rank number, the lower is the document relevancy. However, some nonrelevant documents are placed ahead of relevant documents in the rank ordering. In the collection, we assume that there are 5 relevant documents to the query.

For ranked retrieval, we compute precision and recall *at each rank position*. For example, precision for IR System 1 at rank position one is 1.00 (since one out of one document(s) is relevant); precision is 0.20 (since one out of the five relevant documents is retrieved). Likewise, at rank position six, precision is 0.67 (out of six documents, four are relevant) and recall is 0.80 (four out of the five relevant documents are retrieved).

The above method of enumerating precision and recall at every rank position is unsuitable if the system retrieves a large number of relevant documents for a query, or the relevant documents are interspersed with nonrelevant ones throughout the rank ordering. One solution to this problem is to compute precision and recall only at a small number of *predefined rank positions—precision at rank position p*. Under this approach, comparing effectiveness of two IR systems requires considering precision only. This is because if the precision of an IR System 1 at rank position p is greater than that of the IR System 2 at the same position, then the recall of the IR System 1 will also be greater than that of the IR System 2. For example, in Fig. 10, consider the rank position 4. For IR System 1, precision and recall are 0.75 and 0.60; the same for IR System 2 are 0.50 and 0.40.

Though many values are possible for the rank position p, precision @10 and precision @20 are most commonly used. This measure does not differentiate rank orderings for two systems for positions 1 through p as seen in Fig. 10—precision @10 is the same for both systems. However, this distinction may be important for some search applications.

Another approach to measuring the effectiveness of a ranking is to compute precision at standard recall levels from 0.0 to 1.0 in increments of 0.1. As seen in Fig. 10, values of precision at all these standard recall levels are typically not available. However, such precision values can be computed using *interpolation*.

The third approach, named *average precision*, is widely used in practice for measuring the effectiveness of a ranking. It involves averaging the precision values that correspond to rank positions where a relevant document is retrieved. These are the rank positions where recall values increase.

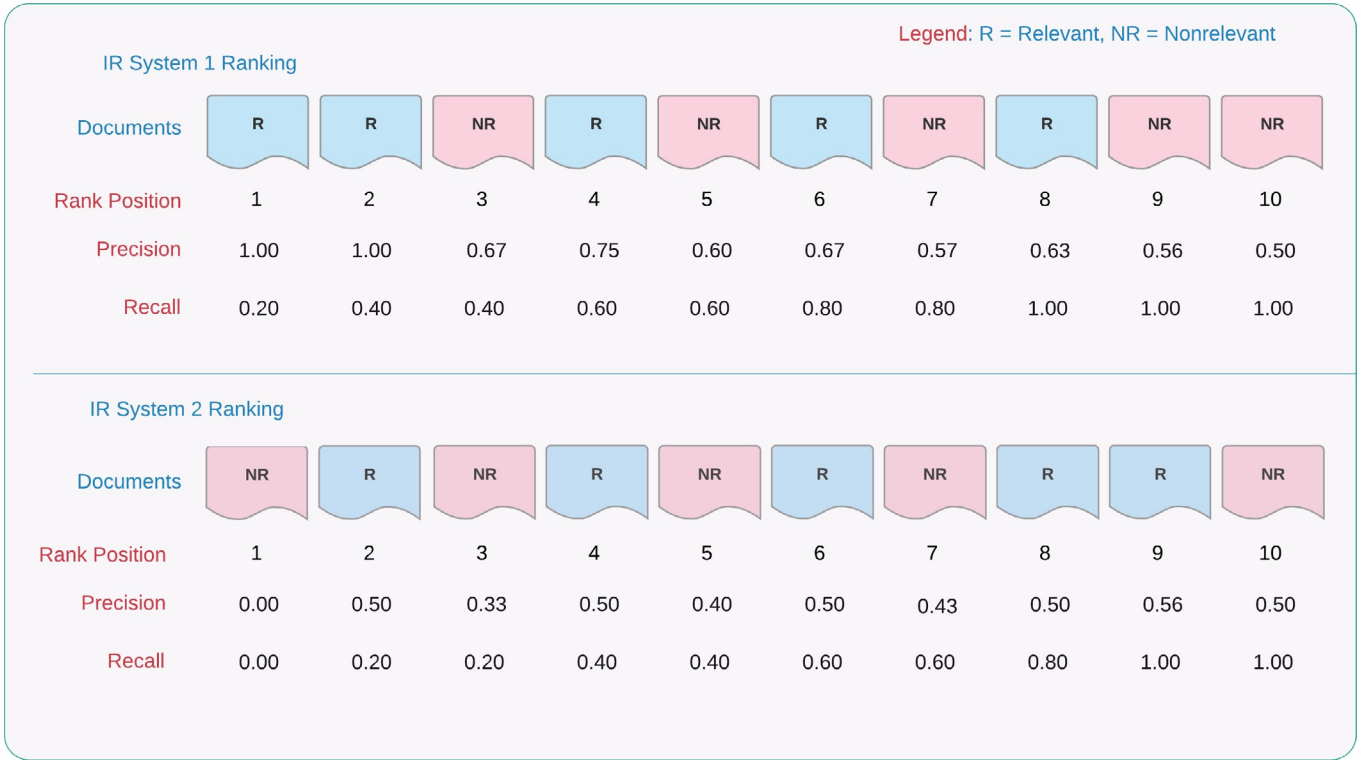

FIG. 10 Ranking effectiveness.

For example, for the IR System 1, such rank positions are 1, 2, 4, 6, and 8. The average precision for IR System 1 ranking is (Fig. 10):

$$(1.00 + 1.00 + 0.75 + 0.67 + 0.63)/5 = 0.81$$

The advantage of the average precision measure is that it provides just one number. Though this number is profoundly influenced by highly ranked documents, this measure is appropriate for situations that value retrieving as many relevant documents as possible and consider the top-ranked documents as the most important/relevant documents (e.g., web search).

The measure that extends the *average precision* for multiple queries for a given IR system is *mean average precision*, which is the average of the average precisions. Again, consider Fig. 10 and assume that the two rankings shown are for the same system but for different queries. We have already calculated average precision for the first ranking as 0.81. The average precision for the second ranking is:

$$(0.50 + 0.50 + 0.50 + 0.50 + 0.56)/5 = 0.512$$

Then the *mean average precision* is (0.81 + 0.512)/2 = 0.661. Note how the *average precision* is heavily influenced by highly ranked documents by observing the two rankings of Fig. 10.

12.4 Precision-Recall Graphs

Though the *mean average precision* provides effectiveness of a system over several queries using just one number, much information is lost in the process. This is where *precision-recall* graphs come into play. Shown in Fig. 11 are the precision-recall graphs for the two rankings of Fig. 10.

12.5 Reciprocal Rank

In applications such as *question-answering*, there is just one relevant document. In other applications such as *web search*, a user is interested in only the first few documents. In these types of applications, effectiveness measures should emphasize how well an IR system retrieves relevant documents at very high rank positions—closer to the top of rank ordering. Precision/recall are not appropriate effectiveness measures for these situations. Another measure called the *reciprocal rank* is used instead. Reciprocal rank is defined as the reciprocal of the rank at which the first relevant document appears in the rank ordering. For example, in Fig. 10, the reciprocal rank for IR System 1 is 1/1 since the first relevant document appears in rank position 1. The reciprocal rank for IR System 2 is 1/2 since the first relevant document appears in rank position 2. The Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks over several queries. If we assume that the two rank orderings in Fig. 10 correspond to two different queries, MRR = (1 + 1/2)/2 = 0.75.

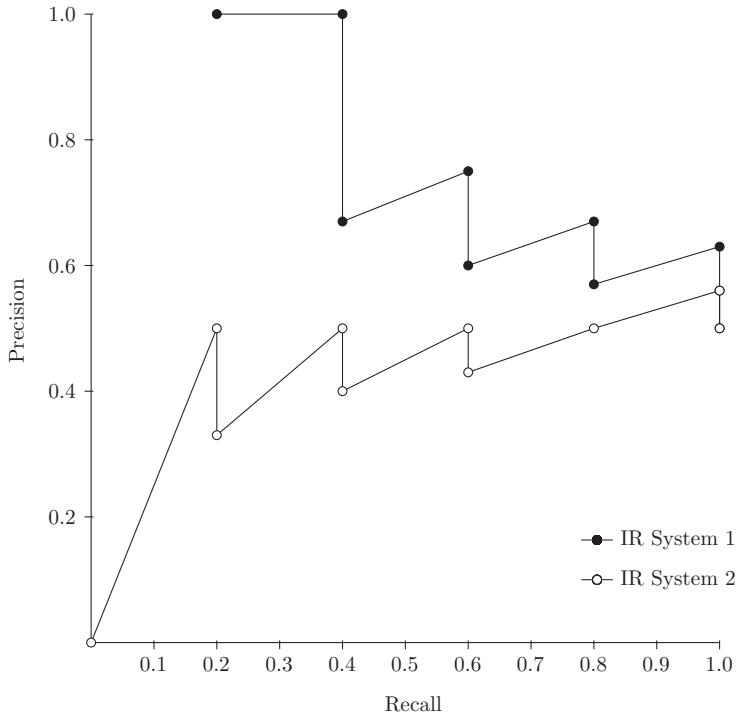

FIG. 11 Precision-recall graphs for the rankings shown in Fig. 10.

12.6 Discounted Cumulative Gain

Discounted Cumulative Gain (DCG) is a popular measure for evaluating web search-type applications (Croft et al., 2010). DCG makes two assumptions: highly relevant documents are more useful than marginally relevant documents, and the user is less likely to examine relevant documents if they are placed farther down in the rank ordering. DCG uses the notion of graded/multilevel relevance as a measure of usefulness/gain from examining a document. The gain is accumulated starting at the top of the rank ordering and is discounted at lower ranks. The DCG is the total gain accumulated at a specific rank p, and is defined:

$$\mathrm{DCG}_p = \mathrm{rel}_1 + \sum_{i=2}^{p} \frac{\mathrm{rel}_i}{\log_2 i} \tag{27}$$

where rel_i is the graded relevance of the document retrieved at rank position i. The $\log_2$ term in the denominator discounts/reduces the gain factor. Graded relevance indicates the degree of relevance. Table 15 shows the five levels of relevance of documents to queries in the Cranfield dataset. The second column in the table is reproduced verbatim from the original source. It should be noted that the numbering scheme used for indicating the level of relevance in

TABLE 15 Graded Relevance for Cranfield Collection Queries

Relevance Level	Description
1	References which are a complete answer to the question
2	References of a high degree of relevance, the lack of which either would have made the research impracticable or would have resulted in a considerable amount of extra work
3	References which were useful, either as general background to the work or as suggesting methods of tackling certain aspects of the work
4	References of minimum interest, for example, those that have been included from an historical viewpoint
5	References of no interest

Cranfield dataset is just the reverse of the scheme used for computing DCG. For DCG, 1 indicates the lowest relevance, and higher numbers indicate greater relevance.

Assume that an IR system has retrieved documents in the following rank order for a query: d_1, d_2, d_3, d_4, d_5. Also, assume three levels of relevance and relevance for the rank order as 3, 3, 1, 1, 2. We compute DCG using Eq. (27):

$$\mathrm{DCG}_p = 3 + (\frac{3}{1} + \frac{1}{1.58} + \frac{1}{2} + \frac{2}{2.32}) = 3 + (3 + 0.63 + 0.50 + 0.86) = 7.99$$

The DCG at each rank is formed by accumulating the preceding numbers—3, 6, 6.63, 7.13, and 7.99. In other words, DCG at rank 1 is 3; rank 2 is 6; rank 3 is 6.63, and so on.

Assume that the rank ordering d_1, d_2, d_3, d_4, d_5 contains all the relevant documents for a query. Then the *perfect ranking* would have the following gain values for the query: 3, 3, 2, 1, 1. The *ideal DCG* (IDCG) values at various ranks are computed as: $3 + (\frac{3}{1} + \frac{2}{1.58} + \frac{1}{2} + \frac{1}{2.32}) = 3 + 3 + 1.27 + 0.50 + 0.43$. The DCG at each rank is formed by accumulating the preceding numbers—3, 6, 7.27, 7.77, and 8.2.

To enable averaging across queries with different number of relevant documents, we normalize DCG. The normalized DCG is denoted NDCG_p, and is defined as:

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}$$

Continuing the above example, $\mathrm{NDCG}_1 = \frac{\mathrm{DCG}_1}{\mathrm{IDCG}_1} = \frac{3}{3} = 1; \mathrm{NDCG}_3 = \frac{\mathrm{DCG}_3}{\mathrm{IDCG}_3} = \frac{6.63}{7.27} = 0.91; \mathrm{NDCG}_5 = \frac{\mathrm{DCG}_5}{\mathrm{IDCG}_5} = \frac{7.99}{8.2} = 0.97$. Note that NDCG is always ≤ 1 for any rank position.

12.7 Eliciting Relevance Judgments Using Pooling

We conclude this section by noting the difficulty involved in eliciting query-document relevance judgments. This is a manually intensive task, often tedious and error-prone. For large collections, manual elicitation of relevance judgments is impractical. *Pooling* is a technique which is used to automate this task. For a given query, top k items from the rank orderings produced by various IR systems are merged into a pool, duplicates are eliminated, and the items are presented in a random order to experts who perform relevance judgments. This process leads to another issue—*interrater reliability* or *concordance*. Relevance judgments for a query provided by multiple experts tend to vary. A common measure to quantify agreement between various raters/judges is the *kappa statistic*, which is defined as:

$$\text{kappa} = \frac{p(A) - p(E)}{1 - p(E)}$$

where $p(A)$ is the proportion of times the judges agreed, and $p(E)$ is the proportion of times they are expected to agree by chance. If two judges always agree, then *kappa* value is equal to 1; 0 if they agree only at the rate given by chance; and a negative value if their agreement rate is worse than random.

13 RELEVANCE FEEDBACK AND QUERY EXPANSION

The notion of *relevance feedback* is based on the premise that users' of IR systems often find it difficult to express their information needs precisely in the form of a query. The IR system retrieves documents based on the initial query, and the user is asked to provide *relevance judgments* in the form of whether or not the retrieved documents are relevant to the user's information need. Relevance judgments can be in the form of *two-level* or *multilevel* relevance relations. In a two-level relevance, a user may label a document as *relevant* or *nonrelevant*; multilevel relevance, for example, may involve labeling a document as *relevant*, *somewhat relevant*, or *nonrelevant*. The set of documents considered relevant comprise the *positive feedback* set, and the nonrelevant ones the *negative feedback* set. In rank ordered outputs, relevance judgments may be limited to the top k documents for practicality reasons.

The IR system will utilize relevance judgments in one of two ways: modifying the query representation or modifying document representations. The effect of the first approach is limited to the current user query session and has no effect on other queries of the current or other users. In contrast, modifications to document representations entail affecting the effectiveness of future queries.

13.1 Modifying Query Representation

Consider the modification of query representation based on relevance feedback. To make the discussion concrete, we assume the Vector Space Model

(VSM). In the first approach, query term weights are adjusted by adding document vectors in the positive feedback set to the query vector. In addition, negative feedback can be made use of by subtracting the document vectors in the negative feedback set from the query vector. This process is repeated a few times until the user is satisfied with the retrieved documents (Rocchio and Salton, 1965). The reformulated query is expected to retrieve additional relevant documents that are similar to the documents in the positive feedback set. In general, positive feedback is consistently more effective in improving recall. Another technique called *dec hi* uses all the documents in the positive feedback set, but only highest ranked nonrelevant documents in the negative feedback set (Harman, 1992).

In the second approach, the original query is modified by adding new terms. The new terms are selected from the positive feedback set, sorted, and filtered, and a predefined number of top terms are added to the query. No effectiveness improvement was observed when more than 20 terms are added to the query (Harman, 1992). Other work in this direction includes concept-based query expansion (Qiu and Frei, 1993), improving automatic query expansion (Mitra et al., 1998), incremental relevance feedback (Aalbersberg, 1992), evaluation of ranking algorithms for interactive query expansion using user choices (Efthimiadis, 1995), and relevance feedback and inference networks (Haines and Croft, 1993).

In some cases, the techniques discussed above do not produce good results. This happens when the documents in the positive feedback set are not homogeneous, or when the nonrelevant documents are scattered among the relevant ones. The documents in the positive feedback set are *clustered* to determine if multiple clusters exist. If so, the original query is split into a number of subqueries such that each subquery is a representative of one of the clusters. The term weights in the subqueries are adjusted or expanded using the two methods discussed above.

13.2 Modifying Document Representation

This involves adjusting document vector representations using relevance feedback. This technique is also referred to as *user-oriented clustering* (Bhuyan et al., 1997; Deogun et al., 1989). The term weights of the retrieved and relevant document vectors are adjusted in way to move them closer to the query vector. Likewise, the term weights of retrieved and nonrelevant documents vectors are adjusted to move them away from the query vector. Experimental results indicate that no gains are achieved after two to three iterations.

13.3 Pseudo-Relevance Feedback

Pseudo-relevance refers to eliciting relevance feedback without involving the search user. For example, the top k documents retrieved by an IR system for an initial query are assumed relevant and comprise the positive feedback set.

These pseudo-feedback documents are analyzed, terms extracted and filtered, and added to the initial query. Pseudo-relevance is based on the premise that the most frequent terms in the pseudo-feedback documents are useful for the retrieval. However, in Cao et al. (2008), it is shown that this assumption does not hold in reality. This study shows that pseudo-feedback documents even when supplemented by the collection documents are not effective in distinguishing good terms from bad ones for query expansion. Their approach integrates a term classification scheme to determine the usefulness of a term for query expansion. Furthermore, the approach integrates multiple additional features and experimental results on three TREC collections validate the effectiveness of the approach.

Other pseudo-relevance techniques for query expansion include thesaurus and query logs. A thesaurus groups words into sets of synonyms called *synsets*. Typically grouping is performed at the word level and the context is ignored. Thesauri can be built manually or automatically. For example, PubMed and WordNet are manually built thesauri. An approach to constructing a thesaurus automatically is based on word co-occurrences. It is based on the assumption that the words that occur together are similar or related in meaning. The advantage of this approach is that *unsupervised learning* (e.g., Latent Semantic Indexing) can be used for the construction. However, synsets may include false synonyms.

Another approach to thesaurus construction uses *supervised learning* and is based on *parallel corpus*. A parallel corpus is essentially a set of sentences in a language L_1 and the corresponding sentences in another language L_2. Machine learning-based translation systems use parallel corpus as training data to learn translation. A set of words s_1 in L_1 may translate into the same word in L_2. Then the words in s_1 are considered as synonyms.

Search engines generally store user queries and this information is referred to as *query logs*. One use for query logs is in automatic query expansion. Cui et al. investigated an approach to query expansion based on query logs (Cui et al., 2002). The query logs are used to extract probabilistic correlations between query and document terms. These correlations are used to determine high-quality terms for expanding the original query. Based on experimental results, the authors claim that query expansion based on query logs data significantly improves the search performance over existing methods. Other uses for query logs include personalization of search results, search terms autocompletion, and correction of spelling errors in user queries, among others. For example, query logs have been used for acquiring knowledge about search user higher-level goals (Strohmaier and All, 2012); sampling for improved query probing in noncooperative distributed information retrieval environments (Shokouhi et al., 2007); capturing the hidden semantics of search queries (Bing et al., 2018); and discovering children's web search behavior (Duarte Torres et al., 2010).

13.4 Theoretical Optimal Query: Rocchio's Algorithm

In the Vector Space model, recall that documents and queries are represented as n-dimensional vectors in a space spanned by n terms. The centroid of a given set of documents D is denoted by $\vec{\mu}(D)$, and is computed as:

$$\vec{\mu}(D) = \frac{1}{|D|}\sum_{d \in D} \vec{v}(d) \tag{28}$$

where $\vec{v}(d)$ is the vector associated with document d.

Rocchio's algorithm is used to implement relevance feedback in the Vector Space model. Using the user relevance judgments, we septate the documents retrieved by an IR System into two sets—relevant documents (D_r) and nonrelevant documents (D_{nr}). We want to find an optimal query, $\vec{q}_{\text{opt}}$, that maximally separates documents in D_r from those in D_{nr}. $\vec{q}_{\text{opt}}$ is defined as:

$$\vec{q}_{\text{opt}} = \arg\max_{\vec{q}} \left[\text{sim}(\vec{q}, \mu(D_r)) - \text{sim}(\vec{q}, \mu(D_{nr}))\right]$$

We make two assumptions related to Rocchio's method. First, the user knows the terms in the document collection quite well to formulate an initial query. Second, relevant documents contain similar terms. Under these assumptions, we can rewrite $\vec{q}_{\text{opt}}$ as:

$$\vec{q}_{\text{opt}} = \mu(D_r) + [\mu(D_r) - \mu(D_{nr})] \tag{29}$$

$$= \frac{1}{|D_r|}\sum_{\vec{d}_i \in D_r} \vec{d}_j + \left[\frac{1}{|D_r|}\sum_{\vec{d}_i \in D_r} \vec{d}_i - \frac{1}{|D_{nr}|}\sum_{\vec{d}_i \in D_{nr}} \vec{d}_i\right] \tag{30}$$

In Eq. (29), the optimal query is obtained by adding the difference between the centroid vectors of relevant and nonrelevant documents to the centroid of the relevant documents. Eq. (30) is derived from Eq. (29) by substituting expressions for centroids (see Eq. (28)).

In practice, Rocchio's algorithm is used in a manner similar to that in the SMART system (Salton, 1983):

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr}) \tag{31}$$

$$= \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_i \in D_r} \vec{d}_i - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_i \in D_{nr}} \vec{d}_i \tag{32}$$

where $\vec{q}_0$ is the initial query vector; $\vec{q}_m$ is the modified query vector; and α, β, and γ are user-specifiable weights. The weight α specifies the importance/confidence given to the initial query. The initial query is moved toward the

relevant documents (controlled by the β factor), and away from the nonrelevant documents (controlled by the γ factor). If the user provides relevance judgments on several documents, we assign higher values for β and γ. For example, we may assign $\beta = 0.8$ and $\gamma = 0.2$ to reflect greater importance to positive feedback over negative feedback.

One round of relevance feedback has been found to be quite useful, and two rounds are marginally useful. As an alternative to relevance feedback, the user manually revises and resubmits the initial query. Web search engines do not use relevance feedback due to computational cost as well as users' unwillingness to provide explicit feedback. Though pseudo-relevance obviates the need for explicit feedback and user involvement, several iterations may lead to *query drift* (Zighelnic and Kurland, 2008).

14 IR LIBRARIES, FRAMEWORKS, AND TEST COLLECTIONS

Apache Lucene™ is perhaps the best known open-source Java IR library (McCandless et al., 2010). Apache Lucene project has three related/independent components: Lucene Core, PyLucene, and Solr™. Lucene Core functions include text analysis/tokenization and indexing; search based on Vector Space, probabilistic, and language modeling-based IR models; tolerant retrieval/spell checking, and results highlighting. PyLucene is a Python port of the Lucene Core.

14.1 Solr and Elasticsearch

Solr is a high-performance open-source search server built using the Lucene Core (Grainger and Potter, 2015). Client applications communicate with Solr using XML/HTTP. Language bindings are provided for JSON, Python, and Ruby.

Elasticsearch is another open-source search engine based on Lucene Core (Gheorghe et al., 2015). Turnbull and Berryman (2016) view Solr and Elasticsearch as programmable relevance frameworks and illustrate how to improve search relevance. They also discuss how to achieve personalization of search, and incorporate secondary data sources, taxonomies, and text analytics for improving relevance.

14.2 Lucene Image Retrieval

Lucene Image Retrieval (LIRE) is an open-source Lucene-based library for Content-Based Image Retrieval (CBIR). LIRE's functions enable organizing, annotating, indexing, and retrieving information from large, unstructured repositories of images and videos (Lux and Marques, 2013).

14.3 Apache UIMA®

Unstructured Information Management Architecture (UIMA®) is an Apache-licensed open-source implementation of the UIMA specification (Apache Software Foundation, 2018). UIMA defines a common, standard interface to enable text analytics components from multiple vendors to interoperate. UIMA is the only standard recognized by the Organization for the Advancement of Structured Information Standards (OASIS) for semantic search and content analytics. It goes beyond text analytics and includes analysis of audio and video. IBM Content Analytics is a UIMA-compliant analytics engine and platform for developing and deploying text analytic applications (IBM, 2018).

14.4 Lemur and Wumpus

The Lemur project developed the Lemur toolkit, which is an open-source software framework for building language modeling and information retrieval applications (University of Massachusetts Amherst, 2018a). The Lemur project also developed an open-source search engine called Indri, using the toolkit. In addition to building search engines, the toolkit can also be used for developing text analysis tools and IR data resources.

Wumpus is an IR system/search engine developed at the University of Waterloo (The University of Waterloo, 2018). It is implemented using C++, and is freely available under the terms of the GNU General Public License (GPL). Wumpus supports a number of ranking functions/retrieval models including proximity queries, probabilistic IR (BM25), Language Model, and Divergence from Randomness. It has no notion of *document* in the conventional sense, and treats the entire text collection as a single document. This enables treating every part of the text collection as a potential unit for retrieval.

14.5 NLP/IR Tools

As there is a significant overlap between NLP and IR, some NLP tools and libraries are also useful for IR. Such NLP tools include NLTK (Bird et al., 2018), GATE (The University of Sheffield, 2015), Open NLP (The Apache Software Foundation, 2018a), Stanford CoreNLP (Manning et al., 2014; The Apache Software Foundation, 2018b), MALLET (University of Massachusetts Amherst, 2018b), Gensim (Řehůřek, 2018), and RapidMiner (RAPIDMINER, 2018).

Meta is an academic toolkit for applying IR and text mining techniques to real-world text data (Zhai and Massung, 2016). Weka is Java machine learning library for data mining tasks (The University of Waikato, 2018).

14.6 Test Collections

Standard test collections are essential for evaluating IR systems and comparing their retrieval effectiveness with one another (Harman, 2011). Such collections provide three things: a fixed number of documents, a set of information needs/topics (aka queries), and a set of known relevant documents for each of the information needs/topics.

The Cranfield collection is the pioneering test collection developed in the late 1950s. It contains 1398 abstracts of aerodynamics journal articles, a set of 225 queries, and expert-provided relevance judgments for each query. For example, "what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft" is an information need. Relevance of a document to a query is specified at five levels (see Table 15): the document is a complete answer to the information need (level 1), the relevance of the document is high (level 2), the document is useful (level 3), the document is of minimum relevance/interest (level 4), and the document is of no interest (level 5). Expert-provided relevance judgments are available for every query-document pair.

Reuters Corpus Volume I (RCV1) is a freely available archive of over 800,000 newswire stories for IR and machine learning research (Lewis et al., 2004). RCV1 specifically targets text categorization research. Each story is manually assigned a topic code, which denotes the major subject of the story. Topic codes are manually organized into four hierarchical groups—CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets).

OHSUMED is a test collection of 348,566 documents from the MEDLINE database (Hersh et al., 1994). These documents originated from 270 medical journals over a 5-year period (1987–1991). Each document is represented by six fields—author, title, abstract, source, publication type, and one or more categories of Medical Subject Headings (MeSH) of cardiovascular diseases group. For document classification research, the MeSH categories of a document represent its class labels.

14.7 TREC Collections

Text REtrieval Conference (TREC) collections are produced by the annual TREC conference, which is being held since 1991 (National Institute of Standards and Technology (NIST), 2018). TREC collections build on the pioneering Cranfield approach, and extended the methodology as needed (Voorhees and Harman, 2005). The initial TREC collection was created in 1991 to evaluate the results of DARPA's TIPSTER project (Harman, 2011). Since then many large collections have been created along with uniform scoring procedures. TREC provides an excellent forum for IR research groups and product vendors to compare their approach/products' effectiveness on the same test collection. TREC events feature a set of IR tasks known as *tracks*.

For example, TREC 2018 tracks include Complex Answer Retrieval, Incidence Streams, Precision Medicine, Real-Time Summarization, among others.

15 FACETS OF IR RESEARCH

Current and emerging areas of IR research are listed in this section.

15.1 Vocabulary, Faceted, and Exploratory Search

Controlled vocabulary is a means for organizing knowledge in a domain to facilitate effective retrieval through browsing and searching. The Library of Congress Subject Headings (LCSH) is an example of controlled vocabulary. It is essentially a classification scheme, often manually constructed, and is widely used in library cataloging systems. Faceted Application of Subject Terminology (FAST) project is an application of LCSH, whose goal is to simplify LCSH syntax and make LCSH easier to understand, maintain, and use.

Hierarchical structure of concepts in a domain is generally known as a *taxonomy*. Hierarchical organization of a controlled vocabulary for classification purpose is referred to as *faceted classification*. Shiyali Ramamrita Ranganathan, a mathematician and considered by many as the father of modern library science, introduced the concept of faceted vocabularies (Hlava, 2014b).

Faceted vocabulary and classification is central to new ways of seeking information—exploratory search (White and Roth, 2009) and faceted search (Tunkelang, 2009). The search enabled by web search engines is referred to as *known-item search* as the user knows what she is looking for. In contrast, *exploratory search* is an information seeking paradigm of users who do not have a known target document and may not even have a well-defined information need.

Faceted search enables users to explore document collections by applying multiple filters. Each document/information element is classified along multiple explicit dimensions—facets. For example, a document may be faceted along the following dimensions—author name, keywords, journal name, publisher name, and year of publication. Faceted search is also known as faceted navigation or faceted browsing.

15.2 Information Architecture

Information Architecture is defined as the structural design of shared information environments (Ding et al., 2017). Taxonomies and metadata are central to information architecture. The interrelationships among taxonomies, meta data, and information architecture are discussed in Hlava (2014c). This work also discusses the differences among various types of controlled vocabularies including taxonomies, thesauri, authority files, and ontologies. Integrating taxonomies into the workflow of an organization is discussed in Hlava (2014a).

Practical applications of taxonomies such as guiding a search user, taxonomy-enabled search tools such as search engines, crawlers, and spiders are also discussed in this work.

15.3 Search Interfaces and User Modeling

Aspects related to search user interfaces—how humans search for information, good interface design, aesthetically pleasing interfaces, analyzing and evaluating search user interfaces—are discussed in Wilson (2011). Given that measuring user engagement is a complex and multifaceted activity, Lalmas et al. (2014) discuss various ways to measure user engagement. Knowledge of context in search and its impact on user behavior are discussed in Agarwal (2017). They propose theoretical frameworks to map the boundaries, elements, and variables of context. *User modeling* is related to user interfaces, and involves capturing users' search behavior. Chuklin et al. (2015) summarize the advances in modeling user click behavior on a web search engine result page including how click models compare to each other, approaches to evaluation of click models, and click model applications.

15.4 Personal Information Management

Jones (2015) defines *personal information management* (PIM) as "the art of getting things done in our lives through information." In this three-part series, Jones discusses a broad range of PIM topics including novel modalities for input and output; technologies for structuring, storing, searching, transforming, and analyzing personal information. The other two parts in this series are Jones (2012, 2013). This is an emerging IR area with many unsolved problems.

15.5 Neural Network Approaches to IR

Neural network-based approaches to adaptive IR began in late 1980s (Belew, 1989). With recent advances in deep learning (Goodfellow et al., 2016), there has been tremendous interest in applying neural learning to Natural Language Processing (NLP) (Goldberg, 2017) and IR applications such as question-answering, image retrieval (ACM SIGIR, 2018; Lam et al., 2015). Furthermore, the overlap between tools and approaches used for NLP and IR is increasing.

15.6 Query Difficulty

IR systems generally suffer from substantial variance in retrieval effectiveness for certain queries. Estimating the *query difficulty* will help to address these situations appropriately (Carmel and Yom-Tov, 2010).

15.7 Information Extraction

Information Extraction (IE) addresses the issues involved in automatically extracting information from unstructured sources (Sarawagi, 2008). IE enables querying, analyzing, and organizing unstructured information by utilizing the query processing functions of databases.

15.8 Text Simplification

Automatic text simplification is process of transforming text into a simpler form from a language perspective, yet retaining the original message (Saggion, 2017). Text simplification has many applications including IR. For example, an IR system can simplify the language of retrieved documents for users with poor literacy, or those with cognitive or linguistic impairment.

15.9 Machine Translation-Based Approaches to IR

A statistical machine translation-based approach to IR is proposed in Berger and Lafferty (1999). The relevance of a document to a user query is based on the probability that the query would have been generated as a *translation* of the document. Berger and Lafferty developed a model for the document-to-query translation process and an algorithm for learning the model parameters using unsupervised learning. Based on experiments on TREC data, the authors conclude that the model performs well in comparison to conventional IR models.

15.10 XML Retrieval

IR models predominantly focused on retrieval of plain text documents that have no internal structure. In contrast, XML documents are hierarchically organized and have both content and structure. XML retrieval provides a focused access to documents—query returns parts of a document rather than the entire document (Lalmas, 2009). Lalmas describes query languages, indexing strategies, ranking algorithms, and presentation schemes for XML retrieval.

15.11 Dynamic Information Retrieval

Dynamic information retrieval modeling is based on the premise that an IR system should evolve in response to dynamic behavior of documents, relevance, users, and their tasks (Yang et al., 2016). Yang et al. (2016) describe a range of approaches to dynamic information retrieval from classical relevance feedback to partially observable Markov decision processes (POMDPs).

15.12 Metasearch Engines

A metasearch engine acts as a unified interface to several underlying (web) search engines. It passes the user query to the underlying engines, aggregates the results, and returns a single ranked list. Meng and Yu (2011) discuss the functional components that underlie large-scale metasearch engines.

15.13 Scholarly Collaboration Using Academic Social Web Platforms

He and Jeng review the rapid transformation of scholarly collaboration on various academic social web platforms such as CiteULike, Mendeley, Academia.edu, and ResearchGate (He and Jeng, 2016). They examine scholarly collaboration behaviors—sharing of academic resources, exchanging opinions, and keeping up with current research trends.

15.14 Access Control

IR systems have generally sidestepped the problem of securely managing document repositories. A trustworthy repository should provide information assurance in the form of secure document access, event logs, and audit trails (Moore et al., 2016).

15.15 Multimedia and Cross-Language Retrieval

The problem of annotating, indexing, organizing, and retrieving visual information from large unstructured repositories is addressed in Lux and Marques (2013). They illustrate retrieving images by content using open-source Java libraries—Lucene and LIRE (Lucene Image Retrieval). Learning semantic relations between the latent semantic structure of linguistic and visual data using deep neural network architectures is an area of immense research activity.

Cross-language information retrieval (CLIR) enables the process of finding documents written in one natural language for the queries expressed in other languages (Nie, 2010). Translation is the key problem in CLIR—either the query or the documents are translated from a language to another.

15.16 Long-Range IR Challenges and Opportunities

During a 3-day workshop in February 2012, 45 IR researchers met to discuss long-range challenges and opportunities in the field (Allan et al., 2012). The result of the workshop is a diverse set of research directions, project ideas, and challenge areas. This report describes the workshop format, provides summaries of broad themes that emerged, includes brief descriptions of all the ideas, and provides detailed discussion of six proposals that were voted

"most interesting" by the participants. Key themes include the need to: move beyond ranked lists of documents to support richer dialog and presentation, represent the context of search and searchers, provide richer support for information seeking, enable retrieval of a wide range of structured and unstructured content, and develop new evaluation methodologies.

16 ADDITIONAL READING

This section provides a list of resources for further exploration of the IR field.

16.1 Earliest Books

The earliest IR book is Salton (1968). Subsequent Salton's IR books include Salton (1983, 1988). In fact, this chapter's senior author first learned IR from Salton (1983). In addition to foundational knowledge including natural language processing, this work describes SMART retrieval system, which is the first to implement the Vector Space model. In contrast, Salton (1988) focuses on practical aspects of IR systems—file access, editing and formatting, data compression and encryption, indexing, abstracting, spell checking, syntax, and style checking.

Rijsbergen (1979) is the earliest book which has dedicated a complete chapter to probabilistic IR. A definitive theoretical resource and a practical guide to text indexing and compression is Witten et al. (1999). Grossman and Frieder (2004) is still a relevant IR reference. It provides an exposition of IR models, tools, cross-language IR, parallel IR, and integrating text with structured data. Belew (2001) offers a cognitive science perspective to the study of information as a computer science discipline using the notion *Finding Out About*.

16.2 Recent Books

Manning et al. (2008) offer a more recent and comprehensive treatment of various IR topics: index construction and compression, term weighting, IR models (Vector Space, Probabilistic, and Language Models), XML retrieval, query processing and retrieval evaluation, text classification and clustering, and web search. Büttcher et al. (2010) is a succinct, rigorous, and comprehensive treatise on modern IR. Authors of this book have developed an academic search engine called *Wumpus* and use this engine to obtain all performance figures presented in the book. Baeza-Yates and Ribeiro-Neto (2011) is another comprehensive introduction to modern IR. Various authors have contributed chapters to this edited work.

Croft et al. (2010) is a very readable introduction to IR and web search engines. Though the focus of this book is on web search engines, it provides an excellent introduction to IR concepts and models. Lastly, Zhai and

Massung (2016) is a recent book which focuses on text data mining and IR techniques that are needed to build text information systems such as search engines and recommender systems. MeTA is an open-source software that accompanies this book, which is intended for enabling readers to quickly run controlled experiments.

The Answer Machine is a nontechnical guide to search and content analytics (Feldman, 2012). This book describes the search evolution, and provides an overview of search engines, clustering, classification, content analytics, and visualization. It also discusses IBM Watson's DeepQA technology and how it was used to answer Jeopardy game questions.

Though IR systems are expected to retrieve *relevant documents*, the notion of *relevance* is not defined explicitly. It is assumed that the IR system users know what relevance means. Saracevic (2016) traces the evolution of relevance in information science from a human point of view. It provides detailed answers to questions such as what is relevance, its properties and manifestations, and factors that affect relevance assessments.

In IR applications such as web and enterprise search, efficiently identifying *near-duplicates* in massive document collections is an important task. Manasse (2012) provides a detailed account on efficient algorithms for detecting closely related web pages.

16.3 Machine Learning and NLP

Machine Learning (ML) and Natural Language Processing (NLP) play critical roles in IR. Barber (2012), Murphy (2012), Manning and Schütze (1999), and Goodfellow et al. (2016) are good resources for ML and NLP. Ingersoll et al. (2013) provide a basic introduction to text processing using IR and NLP open-source software tools.

16.4 Journals and Conferences

Given the rapid advances in the IR and web search fields, books should be supplemented by conference papers and journal articles. Morgan & Claypool Publishers offer Synthesis Digital Library of Engineering and Computer Science. The basic components of the library are short electronic books that synthesize important research or development topics. These books are intended to offer unique value to the reader by providing more synthesis, analysis, and depth compared to typical research journal articles. They are organized by *series*, and the following series are relevant to IR: Information Concepts, Retrieval, and Services; Human Language Technologies; Information Security, Privacy, & Trust; and Data Management. Similarly, *now* Publishers offer recent research through Foundations and Trends® *series*. The series that are relevant to IR include Information Retrieval; Web Science; Databases; and Machine Learning. IR journals include ACM Transactions on Information

Systems, Elsevier's Information Processing & Management, and Springer's Information Retrieval.

The Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) is the premier IR conference. Other conferences include the ACM International Conference on Web Search and Data Mining (ACM WSDM), Text REtrieval Conference (TREC), SPIRE: String Processing and Information Retrieval; European Conference on Information Retrieval (ECIR); ACM Conference on Information and Knowledge Management (CIKM); and Knowledge Discovery and Data Mining (KDD).

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation IUSE/PFE:RED award #1730568. The authors are grateful to Oliver Chen for his help with some illustrations.

REFERENCES

Aalbersberg, I.J., 1992. Incremental relevance feedback. In: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '92. ACM, New York, NY, pp. 11–22.

ACM, SIGIR, 2018. Neu-IR: Workshop on Neural Information Retrieval. https://neu-ir.weebly.com/. (Retrieved: 2018-06-03).

Agarwal, N.K., 2017. Exploring context in information behavior: seeker, situation, surroundings, and shared identities. Synth. Lect. Inf. Concepts Retr. Serv. 9 (7), i–163. https://doi.org/10.2200/S00807ED1V01Y201710ICR061.

Allan, J., Croft, B., Moffat, A., Sanderson, M. (Eds.), 2012. Frontiers, Challenges, and Opportunities for Information Retrieval, In: The Second Strategic Workshop on Information Retrieval in Lorne.

Amigó, E., Fang, H., Mizzaro, S., Zhai, C., 2018. Report on the SIGIR 2017 workshop on axiomatic thinking for information retrieval and related tasks (ATIR). SIGIR Forum 51 (3), 99–106. https://doi.org/10.1145/3190580.3190596.

Apache Software Foundation, 2018. Apache UIMA Project. https://uima.apache.org/. (Retrieved: 2018-03-31).

Baeza-Yates, R., Ribeiro-Neto, B., 2011. Modern Information Retrieval: The Concepts and Technology Behind Search, second ed. ACM Press Books/Addison-Wesley Professional, Boston, MA. ISBN 978-0321416919.

Barber, D., 2012. Bayesian Reasoning and Machine Learning. Cambridge University Press, Cambridge, United Kingdom. ISBN 978-0521518147.

Belew, R.K., 1989. Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents. In: Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '89. ACM, New York, NY, pp. 11–20.

Belew, R.K., 2001. Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW. Cambridge University Press, New York, NY. ISBN 978-0521630283.

Berger, A., Lafferty, J., 1999. Information retrieval as statistical translation. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99. ACM, New York, NY, pp. 222–229.

Bhuyan, J.N., Deogun, J.S., Raghavan, V.V., 1997. Algorithms for the boundary selection problem. Algorithmica 17, 133–161.

Bing, L., Niu, Z.Y., Li, P., Lam, W., Wang, H., 2018. Learning a unified embedding space of web search from large-scale query log. Knowl.-Based Syst. 150, 38–48. ISSN 0950-7051. https://doi.org/10.1016/j.knosys.2018.02.037.

Bird, S., Klein, E., Loper, E., 2018. Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit. http://www.nltk.org/book/. (Retrieved: 2018-03-31).

Büttcher, S., Clarke, C., Cormack, G., 2010. Information Retrieval: Implementing and Evaluating Search Engines. The MIT Press, Cambridge. ISBN 978-0262026512.

Cao, G., Nie, J.Y., Gao, J., Robertson, S., 2008. Selecting good expansion terms for pseudo-relevance feedback. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08. ACM, New York, NY, pp. 243–250.

Carmel, D., Yom-Tov, E., 2010. Estimating the query difficulty for information retrieval. Synth. Lect. Inf. Concepts Retr. Serv. 2 (1), 1–89. https://doi.org/10.2200/S00235ED1V01Y201004ICR015.

Chamberlin, D.D., Boyce, R.F., 1974. SEQUEL: a structured English query language. In: Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control, SIGFIDET '74. ACM, New York, NY, pp. 249–264.

Chuklin, A., Markov, I., de Rijke, M., 2015. Click models for web search. Synth. Lect. Inf. Concepts Retr. Serv. 7 (3), 1–115. https://doi.org/10.2200/S00654ED1V01Y201507ICR043.

Codd, E.F., 1970. A relational model of data for large shared data banks. Commun. ACM 13 (6), 377–387.

Croft, W.B., Metzler, D., Strohman, T., 2010. Search Engines: Information Retrieval in Practice. Addison-Wesley, Boston, MA. ISBN 978-0136072249.

Cui, H., Wen, J.R., Nie, J.Y., Ma, W.Y., 2002. Probabilistic query expansion using query logs. In: Proceedings of the 11th International Conference on World Wide Web, WWW '02. ACM, New York, NY, pp. 325–332.

Deogun, J.S., Raghavan, V.V., Rhee, P., 1989. Formulation of the term refinement problem for user-oriented information retrieval. In: Proceedings of the Annual AI Systems in Government Conference. Washington, DC, pp. 72–78.

Ding, W., Lin, X., Zarro, M., 2017. Information architecture: the design and integration of information spaces. Synth. Lect. Inf. Concepts Retr. Serv. 9 (2), 1–152. https://doi.org/10.2200/S00755ED2V01Y201701ICR056.

Duarte Torres, S., Hiemstra, D., Serdyukov, P., 2010. Query log analysis in the context of information retrieval for children. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10. ACM, New York, NY, pp. 847–848.

Efthimiadis, E.N., 1995. User choices: a new yardstick for the evaluation of ranking algorithms for interactive query expansion. Inf. Process. Manag. 31 (4), 605–620. https://doi.org/10.1016/0306-4573(95)00070-W.

Fang, H., Tao, T., Zhai, C., 2004. A formal study of information retrieval heuristics. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '04. ACM, New York, NY, pp. 49–56.

Feldman, S.E., 2012. The answer machine. Synth. Lect. Inf. Concepts Retr. Serv. 4 (3), 1–137. https://doi.org/10.2200/S00442ED1V01Y201208ICR023.

Gheorghe, R., Hinman, M.L., Russo, R., 2015. Elasticsearch in Action. Manning, Shelter Island, NY. ISBN 978-1617291623.

Goldberg, Y., 2017. Neural network methods for natural language processing. Synth. Lect. Hum. Lang. Technol. 10 (1), 1–309. https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. The MIT Press, Cambridge. ISBN 978-0262035613.

Grainger, T., Potter, T., 2015. Solr in Action. Manning, Shelter Island, NY. ISBN 978-1617291029.

Grefenstette, G., Wilber, L., 2010. Search-based applications: at the confluence of search and database technologies. Synth. Lect. Inf. Concepts Retr. Serv. 2 (1), 1–141. https://doi.org/10.2200/S00320ED1V01Y201012ICR017.

Grossman, D.A., Frieder, O., 2004. Information Retrieval: Algorithms and Heuristics, second ed. The Information Retrieval Series. Springer, New York, NY. ISBN 978-1402030048.

Gudivada, V.N., Rao, D., Raghavan, V.V., 2014. NoSQL systems for big data management. IEEE World Congress Serv. 190–197. http://doi.ieeecomputersociety.org/10.1109/SERVICES.2014.42.

Gudivada, V., Rao, D., Raghavan, V., 2015. Big data driven natural language processing research and applications. In: Govindaraju, V., Raghavan, V., Rao, C.R. (Eds.), Big Data Analytics, Handbook of Statistics, vol. 33. Elsevier, New York, NY, pp. 203–238.

Gudivada, V., Rao, D., Raghavan, V., 2016. Renaissance in database management: navigating the landscape of candidate systems. IEEE Comput. 49 (4), 31–42.

Haines, D., Croft, W.B., 1993. Relevance feedback and inference networks. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93. ACM, New York, NY, pp. 2–11.

Harman, D., 1992. Relevance feedback revisited. In: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '92. ACM, New York, NY, pp. 1–10.

Harman, D., 2011. Information retrieval evaluation. Synth. Lect. Inf. Concepts Retr. Serv. 3 (2), 1–119. https://doi.org/10.2200/S00368ED1V01Y201105ICR019.

He, D., Jeng, W., 2016. Scholarly collaboration on the academic social web. Synth. Lect. Inf. Concepts Retr. Serv. 8 (1), 1–106. https://doi.org/10.2200/S00698ED1V01Y201601ICR047.

Hersh, W., Buckley, C., Leone, T.J., Hickam, D., 1994. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94. Springer-Verlag, New York, NY, pp. 192–201.

Hiemstra, D., 2001. Using language models for information retrieval. Ph.D. thesis, University of Twente, Enschede, The Netherlands.

Hlava, M.M.K., 2014a. The taxobook: applications, implementation, and integration in search: part 3 of a 3-part series. Synth. Lect. Inf. Concepts Retr. Serv. 6 (4), 1–156. https://doi.org/10.2200/S00604ED1V03Y201410ICR037.

Hlava, M.M.K., 2014b. The taxobook: history, theories, and concepts of knowledge organization, part 1 of a 3-part series. Synth. Lect. Inf. Concepts Retr. Serv. 6 (3), 1–80. https://doi.org/10.2200/S00602ED1V01Y201410ICR035.

Hlava, M.M.K., 2014c. The taxobook: principles and practices of building taxonomies, part 2 of a 3-part series. Synth. Lect. Inf. Concepts Retr. Serv. 6 (4), 1–164. https://doi.org/10.2200/S00603ED1V02Y201410ICR036.

IBM, 2018. Content Analytics With Enterprise Search: UIMA Framework. https://www-01.ibm.com/software/ecm/content-analytics/uima.html. (Retrieved: 2018-04-02).

Ingersoll, G.S., Morton, T.S., Farris, A.L., 2013. Taming text: how to find, organize, and manipulate it. Manning, Shelter Island, NY. ISBN 978-1933988382.

Jones, W., 2012. The future of personal information management, part I: our information, always and forever. Synth. Lect. Inf. Concepts Retr. Serv. 4 (1), 1–125. https://doi.org/10.2200/S00411ED1V01Y201203ICR021.

Jones, W., 2013. Transforming technologies to manage our information: the future of personal information management, part II. Synth. Lect. Inf. Concepts Retr. Serv. 5 (4), 1–179. https://doi.org/10.2200/S00532ED1V01Y201308ICR028.

Jones, W., 2015. Building a better world with our information: the future of personal information management, part 3. Synth. Lect. Inf. Concepts Retr. Serv. 7 (4), 1–203. https://doi.org/10.2200/S00653ED1V01Y201506ICR042.

Kraft, D.H., Colvin, E., 2017. Fuzzy information retrieval. Synth. Lect. Inf. Concepts Retr. Serv. 9 (1), 1–63. https://doi.org/10.2200/S00752ED1V01Y201701ICR055.

Lafferty, J., Zhai, C., 2001. Document language models, query models, and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01. ACM, New York, NY, pp. 111–119.

Lalmas, M., 2009. XML retrieval. Synth. Lect. Inf. Concepts Retr. Serv. 1 (1), 1–111. https://doi.org/10.2200/S00203ED1V01Y200907ICR007.

Lalmas, M., O'Brien, H., Yom-Tov, E., 2014. Measuring user engagement. Synth. Lect. Inf. Concepts Retr. Serv. 6 (4), 1–132. https://doi.org/10.2200/S00605ED1V01Y201410ICR038.

Lam, A.N., Nguyen, A.T., Nguyen, H.A., Nguyen, T.N., 2015. Combining deep learning with information retrieval to localize buggy files for bug reports. In: 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 476–481.

Lavrenko, V., Croft, W.B., 2001. Relevance based language models. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01. ACM, New York, NY, pp. 120–127.

Lewis, D.D., Yang, Y., Rose, T.G., Li, F., 2004. RCV1: a new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397. ISSN 1532-4435.

Li, H., 2014. Learning to Rank for Information Retrieval and Natural Language Processing, Synthesis Lectures on Human Language Technologies, vol. 7, 3. Morgan & Claypool, San Rafael, CA, pp. 1–121. https://doi.org/10.2200/S00607ED2V01Y201410HLT026.

Lin, J., Dyer, C., 2010. Data-Intensive Text Processing With MapReduce. Morgan & Claypool, San Rafael, CA. ISBN 9781608453436. https://doi.org/10.2200/S00274ED1V01Y201006HLT007.

Lux, M., Marques, O., 2013. Visual information retrieval using Java and LIRE. Synth. Lect. Inf. Concepts Retr. Serv. 5 (1), 1–112. https://doi.org/10.2200/S00468ED1V01Y201301ICR025.

Manasse, M.S., 2012. On the efficient determination of most near neighbors: horseshoes, hand grenades, web search and other situations when close is close enough. Synth. Lect. Inf. Concepts Retr. Serv. 4 (4), 1–88. https://doi.org/10.2200/S00444ED1V01Y201208ICR024.

Manning, C., Schütze, H., 1999. Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge. ISBN 978-0262133609.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY. ISBN 978-0521865715.

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D., 2014. The Stanford CoreNLP Natural Language Processing Toolkit. Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

McCandless, M., Hatcher, E., Gospodnetic, O., 2010. Lucene in Action, second ed. Manning, Shelter Island, NY. ISBN 978-1933988177.

Meng, W., Yu, C.T., 2011. Advanced Metasearch Engine Technology. Synthesis Lectures on Data Management. Morgan & Claypool, San Rafael, CA. https://doi.org/10.2200/S00307ED1V01Y201011DTM011.

Mitra, M., Singhal, A., Buckley, C., 1998. Improving automatic query expansion. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98. ACM, New York, NY, pp. 206–214.

Moore, R.W., Xu, H., Conway, M., Rajasekar, A., Crabtree, J., Tibbo, H., 2016. Trustworthy policies for distributed repositories. Synth. Lect. Inf. Concepts Retr. Serv. 8 (3), 1–133. https://doi.org/10.2200/S00732ED1V01Y201609ICR051.

Murphy, K., 2012. Machine Learning: A Probabilistic Perspective, Adaptive Computation and Machine Learning. The MIT Press, Cambridge University Press. ISBN 978-0262018029.

National Institute of Standards and Technology (NIST), 2018. Text REtrieval Conference (TREC). https://trec.nist.gov/. (Retrieved: 2018-05-10).

Nie, J.Y., 2010. Cross-Language Information Retrieval. Morgan & Claypool, San Rafael, CA. ISBN 9781598298642.

Ponte, J.M., Croft, W.B., 1998. A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98. ACM, New York, NY, pp. 275–281.

Project Gutenberg, 2018. Offers Over 56,000 Free eBooks. http://www.gutenberg.org/. (Retrieved: 2018-04-02).

Qiu, Y., Frei, H.P., 1993. Concept based query expansion. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93. ACM, New York, NY, pp. 160–169.

Radecki, T., 1979. Fuzzy set theoretical approach to document retrieval. Inf. Process. Manag. 15 (5), 247–259. https://doi.org/10.1016/0306-4573(79)90031-1.

Raghavan, V.V., Wong, S.K.M., 1986. A critical analysis of vector space model for information retrieval. J. Am. Soc. Inf. Sci. 37 (5), 279–287. https://doi.org/10.1002/(SICI)1097-4571(198609)37:5¡279::AID-ASI1¿3.0.CO;2-Q.

RAPIDMINER, 2018. Lightning Fast Unified Data Science Platform. https://rapidminer.com/. (Retrieved: 2018-03-31).

Rijsbergen, C.J.V., 1979. Information Retrieval, second ed. Butterworth-Heinemann, Oxford, UK. ISBN 978-0408709293.

Robertson, S.E., 1997. Overview of the Okapi projects. J. Doc. 53 (1), 3–7. https://doi.org/10.1108/EUM0000000007186.

Rocchio, J.J., Salton, G., 1965. Information search optimization and interactive retrieval techniques. In: Proceedings of the November 30-December 1, 1965, Fall Joint Computer Conference, Part I, AFIPS '65 (Fall, Part I). ACM, New York, NY, pp. 293–305.

Roelleke, T., Wang, J., 2008. TF-IDF uncovered: a study of theories and probabilities. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08. ACM, New York, NY, pp. 435–442.

Saggion, H., 2017. Automatic text simplification. Synth. Lect. Hum. Lang. Technol. 10 (1), 1–137. https://doi.org/10.2200/S00700ED1V01Y201602HLT032.

Salton, G., 1968. Automatic Information and Retrieval. McGraw-Hill, New York, NY. ISBN 978-0070544857.

Salton, G., 1983. Introduction to Modern Information Retrieval, McGraw Hill Computer Science Series, McGraw-Hill, New York, NY. ISBN 978-0070544840.

Salton, G., 1988. Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer, Addison-Wesley Series in Computer Science, Addison-Wesley, Boston, MA. ISBN 978-0201122275.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. 24 (5), 513–523.

Saracevic, T., 2016. The notion of relevance in information science: everybody knows what relevance is. But, what is it really? Synth. Lect. Inf. Concepts Retr. Serv. 8 (3), 1–109. https://doi.org/10.2200/S00723ED1V01Y201607ICR050.

Sarawagi, S., 2008. Information extraction. Found. Trends Databases 1 (3), 261–377. https://doi.org/10.1561/1900000003.

Shokouhi, M., Zobel, J., Tahaghoghi, S., Scholer, F., 2007. Using query logs to establish vocabularies in distributed information retrieval. Inf. Process. Manag. 43 (1), 169–180. https://doi.org/10.1016/j.ipm.2006.04.003.

Strohmaier, M., All, M.K., 2012. Acquiring knowledge about human goals from Search Query Logs. Inf. Process. Manag. 48 (1), 63–82. ISSN 0306-4573. https://doi.org/10.1016/j.ipm.2011.03.010.

The Apache Software Foundation, 2018a. openNLP. http://opennlp.apache.org/. (Retrieved: 2018-04-02).

The Apache Software Foundation, 2018b. Stanford CoreNLP: Natural Language Software. https://stanfordnlp.github.io/CoreNLP/. (Retrieved: 2018-04-02).

The University of Sheffield, 2015. GATE. http://gate.ac.uk/. (Retrieved: 2018-04-02).

The University of Waikato, 2018. Weka 3: Data Mining Software in Java. https://www.cs.waikato.ac.nz/ml/weka/index.html. (Retrieved: 2018-05-01).

The University of Waterloo, 2018. Wumpus Search Engine. http://www.wumpus-search.org/. (Retrieved: 2018-03-31).

Tunkelang, D., 2009. Faceted search. Synth. Lect. Inf. Concepts Retr. Serv. 1 (1), 1–80. https://doi.org/10.2200/S00190ED1V01Y200904ICR005.

Turnbull, D., Berryman, J., 2016. Relevant Search: With Applications for Solr and Elasticsearch. Manning, Shelter Island, NY. ISBN 978-1617292774.

Turney, P.D., Pantel, P., 2010. From frequency to meaning: Vector space models of semantics. J. Artif. Int. Res. 37 (1), 141–188. ISSN 1076-9757. http://dl.acm.org/citation.cfm?id=1861751.1861756.

University of Massachusetts Amherst, 2018a. The Lemur Project. https://en.wikipedia.org/wiki/Lemur_Project. (Retrieved: 2018-31-03).

University of Massachusetts Amherst, 2018b. MAchine Learning for LanguagE Toolkit (MALLET). http://mallet.cs.umass.edu/. (Retrieved: 2018-31-03).

van Rijsbergen, C.J., 1979. Information Retrieval, second ed. Butterworths, Boston, MA. ISBN 0-408-70929-4.

Voorhees, E., Harman, D. (Eds.), 2005. TREC: Experiment and Evaluation in Information Retrieval. In: The MIT Press, Cambridge. ISBN 978-0262220736.

Řehůřek, R., 2018. Gensim: Topic Modeling for Humans. https://radimrehurek.com/gensim/. (Retrieved: 2018-06-09).

White, R.W., Roth, R.A., 2009. Exploratory search: beyond the query-response paradigm. Synth. Lect. Inf. Concepts Retr. Serv. 1 (1), 1–98. https://doi.org/10.2200/S00174ED1V01Y200901ICR003.

Wilson, M.L., 2011. Search user interface design. Synth. Lect. Inf. Concepts Retr. Serv. 3 (3), 1–143. https://doi.org/10.2200/S00371ED1V01Y201111ICR020.

Witten, I.H., Moffat, A., Bell, T.C., 1999. Managing Gigabytes: Compressing and Indexing Documents and Images, second ed. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann, Burlington. ISBN 978-1558605701.

Wong, S.K.M., Yao, Y.Y., 1995. On modeling information retrieval with probabilistic inference. ACM Trans. Inf. Syst. 13 (1), 38–68. https://doi.org/10.1145/195705.195713.

Wong, S.K.M., Ziarko, W., Wong, P.C.N., 1985. Generalized vector spaces model in information retrieval. In: Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '85. ACM, New York, NY, pp. 18–25.

Yang, G.H., Sloan, M., Wang, J., 2016. Dynamic information retrieval modeling. Synth. Lect. Inf. Concepts Retr. Serv. 8 (3), 1–144. https://doi.org/10.2200/S00718ED1V01Y201605ICR049.

Zhai, C., 2008. Statistical Language Models for Information Retrieval, Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA. ISBN 9781598295917. https://doi.org/10.2200/S00158ED1V01Y200811HLT001.

Zhai, C., Lafferty, J., 2001. Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, CIKM '01. ACM, New York, NY, pp. 403–410.

Zhai, C., Massung, S., 2016. Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining. ACM Books, New York, NY. ISBN 978-1970001167.

Zighelnic, L., Kurland, O., 2008. Query-drift prevention for robust query expansion. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08. ACM, New York, NY, pp. 825–826.