1 Text Representations for Ranking

According to the Probability Ranking Principle [Robertson 1977], under certain assumptions, for a given user's query, documents in a collection should be ranked in order of the (decreasing) probability of relevance w.r.t. the query, to maximise the overall effectiveness of a retrieval system for the user. The task of *ad-hoc ranking* is, for each query, to compute an ordering of the documents equivalent or most similar to the optimal ordering based on the probability of relevance. It is common to limit the documents to be ordered to just the top k documents in the optimal ordering.

Let $\mathcal{D}$ denote a collection of (text) documents, and $\mathcal{Q}$ denote a log of (text) queries. Queries and documents share the same vocabulary $\mathcal{V}$ of terms. A *ranking function*, also known as *scoring function*, $s : \mathcal{Q} \times \mathcal{D} \to \mathbb{R}$ computes a real-valued score for the documents in the collection $\mathcal{D}$ w.r.t. the queries in the log $\mathcal{Q}$. Given a query q and a document d, we call the value $s(q, d)$ *relevance score* of the document w.r.t. the query. For a given query, the scores of the documents in the collection can be used to induce an ordering of the documents, in reverse value of score. The closer this induced ordering is to the optimal ordering, the more effective an IR system based on the scoring function is.

Without loss of generality, the scoring function $s(q, d)$ can be further decomposed as:

$$s(q, d) = f\big(\phi(q), \psi(d), \eta(q, d)\big), \tag{1}$$

where $\phi : \mathcal{Q} \to V_1$, $\psi : \mathcal{D} \to V_2$, and $\eta : \mathcal{Q} \times \mathcal{D} \to V_3$ are three *representation functions*, mapping queries, documents, and query-document pairs into the *latent representation spaces* V_1, V_2, and V_3, respectively [Guo et al. 2020]. These functions build abstract mathematical representations of the text sequences of documents and queries amenable for computations. The elements of these vectors represent the features used to describe the corresponding objects, and the *aggregation function* $f : V_1 \times V_2 \times V_3 \to \mathbb{R}$ computes the relevance score of the document representation w.r.t. the query representation.

The representation functions ϕ, ψ and η, and the aggregation function f can be designed by hand, leveraging some axioms or heuristics, or computed through machine learning algorithms. In the following, we will overview the representation functions adopted in classical IR (Section 1.1), in LTR scenarios (Section 1.2) and the recently proposed word embeddings (Section 1.3).

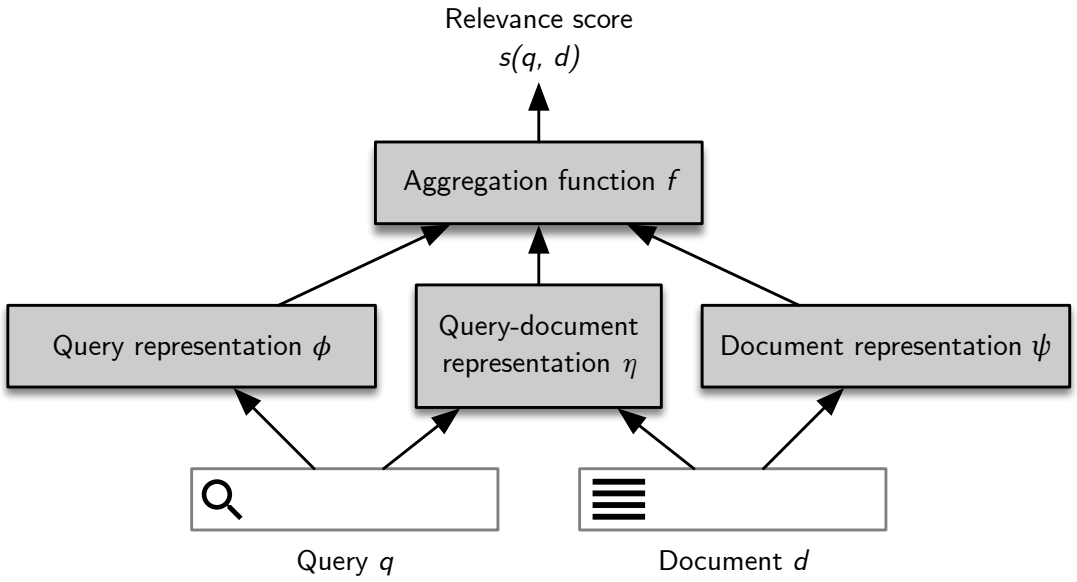

Figure 1: Representation-based decomposition of a ranking function. Adapted from [Guo et al. 2020].

1.1 BOW Encodings

In classical IR, both representation and aggregation functions are designed manually, incorporating some lexical statistics such as number of occurrences of terms in a document or in the whole collection. Classical IR ranking models, e.g., vector space models [Salton et al. 1975], probabilistic models [Robertson and Zaragoza 2009] and statistical language models [Ponte and Croft 1998], are based on the *bag of words* (BOW) model, where queries and documents are represented as a set of terms from the vocabulary $\mathcal{V}$ together with the number of occurrences of the corresponding tokens in the text. More formally, queries and documents are represented as vectors $\phi(q)$ and $\psi(d)$ in $\mathbb{N}^{|\mathcal{V}|}$, called *BOW encodings*, where the i-th component of both representations encodes the number of occurrences of the term $w_i \in \mathcal{V}$ in the corresponding text. The query-document representation function η is not present in these ranking functions. The aggregation function f over these representations is an explicit formula taking into account the components of the query and document representations, i.e., the in-query and in-document term frequencies, together with other document normalisation operations. These representations are referred to as *sparse representations*, since most of their components are equal to 0 because they correspond to tokens not appearing in the query/document. Sparse representations can be trivially computed and efficiently stored in specialised data structures called *inverted indexes*, which represent the backbone of commercial Web search engine [Cambazoglu and Baeza-Yates 2015]; see [Büttcher et al. 2010, Manning et al. 2008, Tonellotto et al. 2018] for more

details on inverted indexes and classical IR ranking models.

1.2 LTR Features

With the advent of the Web, new sources of relevance information about the documents have been made available. The importance of a Web page, e.g., PageRank, additional document statistics, e.g., term frequencies in the title or anchors text, and search engine interactions, e.g., clicks, can be exploited as relevance signals. Moreover, collaborative and social platforms such as Wikipedia, Twitter and Facebook represent new sources of relevance signals. These relevance signals have been exploited to build richer query and document representations in LTR. The relevance signals extracted from queries and/or documents are called *features*. There are various classes of these features [Bendersky et al. 2011, Macdonald et al. 2012], such as:

- *query-only* features, i.e., components of $\phi(q)$: query features with the same value for each document, such as query type, query length, and query performance predictors;
- *query-independent* features, i.e., components of $\psi(d)$: document features with the same value for each query, such as importance score, URL length, and spam score;
- *query-dependent* features, i.e., components of $\eta(q, d)$: document features that depend on the query, such as different term weighting models on different fields.

In LTR, the representation functions are hand-crafted: exploiting the relevance signals from heterogeneous information sources, the different components of query and document representations are computed with feature-specific algorithms. Hence, the representations $\phi(q)$, $\psi(d)$, and $\eta(q, d)$ are elements of vector spaces over $\mathbb{R}$, but whose dimensions depend on the number of hand-crafted query-only, query-independent, and query-dependent features, respectively. Moreover the different components of these vectors are heterogeneous, and do not carry any specific semantic meaning. Using these representations, in LTR the aggregation function f is machine-learned, for example using logistic regression [Gey 1994], gradient-boosted regression trees [Burges 2010] or neural networks [Burges et al. 2005]; see [Liu 2009] for a detailed survey.

1.3 Word Embeddings

Both BOW encodings and LTR features are widely adopted in commercial search engines, but they suffer from several limitations. On the one hand, semantically-related

terms end up having completely different BOW encodings. Although the two terms `catalogue` and `directory` can be considered synonyms, their BOW encodings are completely different, with the single 1 appearing in different components. Similarly, two different documents on a same topic can end up having two unrelated BOW encodings. On the other hand, LTR features create text representations by hand via feature engineering, with heterogeneous components and no explicit concept of similarity.

In the 1950s, many linguists formulated the *distributional hypothesis*: words that occur in the same contexts tend to have similar meanings [Harris 1954]. According to this hypothesis, the meaning of words can be inferred by their usage together with other words in existing texts. Hence, by leveraging the large text collections available, it is possible to learn useful representations of terms, and devise new methods to use these representations to build up more complex representations for queries and documents. These learned representations are vectors in $\mathbb{R}^n$, with $n \ll |\mathcal{V}|$, called *distributional representations* or *word embeddings*. The number of dimensions n ranges approximatively from 50 to 1000 components, instead of the vocabulary size $|\mathcal{V}|$. Moreover, the components of word embeddings are rarely 0: they are real numbers, and can also have negative values. Hence, word embeddings are also referred to as *dense representations*.

Among the different techniques to compute these representations, there are algorithms to compute global representations of the words, i.e., a single fixed embedding for each term in the vocabulary, called *static word embeddings*, and algorithms to compute local representations of the terms, which depend on the other tokens used together with a given term, i.e., its context, called *contextualised word embeddings*. Static word embeddings used in neural IR are learned from real-world text with no explicit training labels: the text itself is used in a self-supervised fashion to compute word representations. There are different kinds of static word embeddings, for different languages, such as word2vec [Mikolov et al. 2013], fasttext [Joulin et al. 2017] and GloVe [Pennington et al. 2014]. Static word embeddings map terms with multiple senses into an average or most common sense representation based on the training data used to compute the vectors; each term in the vocabulary is associated with a single vector. Contextualised word embeddings map tokens used in a particular context to a specific vector; each term in the vocabulary is associated with a different vector every time it appears in a document, depending on the surrounding tokens. The most popular contextualised word embeddings are learned with deep neural networks such as the Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2019], the Robustly Optimized BERT Approach (RoBERTa) [Liu et al. 2019], and the Generative Pre-Training models (GPT) [Radford and Narasimhan 2018].

In neural IR, word embeddings are used to compute the representation functions ϕ,