A Document Retrieval Model Based on Digital Signal Filtering

ALBERTO COSTA, National University of Singapore and ETH Zurich,
Future Resilient Systems Program
EMANUELE DI BUCCIO and MASSIMO MELUCCI, University of Padua, Italy

Information retrieval (IR) systems are designed, in general, to satisfy the information need of a user who expresses it by means of a query, by providing him with a subset of documents selected from a collection and ordered by decreasing relevance to the query. Such systems are based on IR models, which define how to represent the documents and the query, as well as how to determine the relevance of a document for a query. In this article, we present a new IR model based on concepts taken from both IR and digital signal processing (like Fourier analysis of signals and filtering). This allows the whole IR process to be seen as a physical phenomenon, where the query corresponds to a signal, the documents correspond to filters, and the determination of the relevant documents to the query is done by filtering that signal. Tests showed that the quality of the results provided by this IR model is comparable with the state-of-the-art.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Retrieval Models; H.3.4 [**Systems and Software**]: Performance Evaluation (Efficiency and Effectiveness); F.2.1 [**Numerical Algorithms and Problems**]: Computation of Transforms (e.g., Fast Fourier Transform)

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Retrieval models, digital filtering, discrete Fourier transform, DFT

ACM Reference Format:
Alberto Costa, Emanuele Di Buccio, and Massimo Melucci. 2015. A document retrieval model based on digital signal filtering. ACM Trans. Inf. Syst. 34, 1, Article 6 (September 2015), 37 pages.
DOI: http://dx.doi.org/10.1145/2809787

1. INTRODUCTION

The aim of an information retrieval (IR) system is to predict which documents are relevant to the user information need. In ad hoc settings, the input are a query, given by a user, and a set of documents, called *collection*. The goal is to select from the collection the documents that are supposed to answer to the query and to provide them (ordered by decreasing relevance) to the user. Indeed, this is a rough simplification of an Information Retrieval (IR) ad hoc task, which includes many technical and theoretical

This article extends the conference paper [Costa and Melucci 2010]. A. Costa has been partially supported by SUTD-MIT IDC grant IDG21300102. His research was partially conducted at the Future Resilient Systems at the Singapore-ETH Centre (SEC). The SEC was established as a collaboration between ETH Zurich and the National Research Foundation (NRF) Singapore (FI 370074011) under the auspices of the NRF's Campus for Research Excellence and Technological Enterprise (CREATE) program. The work of M. Melucci has partially been funded by the EU 7th Framework Programme Marie Curie IRSES project 247590 "QONTEXT."
Authors' addresses: A. Costa, National University of Singapore and ETH Zurich, Future Resilient Systems Program, 1 CREATE Way #06-01, CREATE Tower, 138602, Singapore; email: costa@lix.polytechnique.fr; E. Di Buccio and M. Melucci, Department of Information Engineering, University of Padua, I-35131, Padua, Italy; emails: {dibuccio, melo}@dei.unipd.it.

DOI: http://dx.doi.org/10.1145/2809787

issues, but this is sufficient to give an intuitive idea about the kind of problems with which IR deals.

IR researchers are constantly searching for theoretical frameworks that help them to make a major breakthrough in the overall effectiveness of retrieval systems. In searching for these frameworks, researchers sometimes use a "metaphor" to introduce a new IR model, at least at the beginning. In the Boolean model, query terms are sets of documents combined by logical operations (i.e., AND, OR, NOT), and the result is the subset of documents of the collection for which this logical expression is true. In the vector space model (VSM), documents and terms are represented by basis vectors combined by linear combinations, and the result is a list of documents such that a document is ranked by a measure that is higher the closer the query vector is to the document vector. In the probabilistic model, terms are random variables mapping the sample space of documents, which are elementary events, to the multivariate real space, and the documents are ranked by the probability for being relevant to the query. This probability is related to the risk of choosing a nonrelevant document and the risk of missing a relevant document [Robertson 1977].

This article describes a new framework that is based on discrete Fourier transform (DFT) and is called *least spectral power ranking* (LSPR). The reason for this name will be more clear in Section 4. Now we provide an intuitive description of the main idea of LSPR by means of a metaphor.

1.1. Intuitive Description

To give a first overview of this model, a metaphor for LSPR is now provided. Suppose that a spacecraft has to bring an object to a destination. To this end, the spacecraft can use a path. Now suppose that this path is shot by bursts of radiation that differ in intensity and frequency. The path can then be related to the radiation absorbed by the spacecraft. Each burst of radiation can be expressed in terms of intensity and frequency, and in our model it is associated with a discrete time sinusoidal signal. Furthermore, suppose that the spacecraft has a number of different shields at its disposal. Each shield has been designed to protect the spacecraft from the radiation bursts, and to this end the shield has different filters that block radiation bursts of varying intensity and frequency. A filter, which is characterized by a frequency and a strength, can absorb radiation with a given frequency and attenuate radiation with a frequency close to the given frequency depending on the strength. The crew of the spacecraft aims at finding the shields that absorb the most amount of radiation possible to minimize the risk for their health. The spacecraft is the system, the path is the query, and the bursts of radiation are the terms of the query. The shields are the documents, and each filter corresponds to a document term weighted by strength. Thus, finding good shields for a given path corresponds to finding good documents for a given query.

The rest of the article is organized as follows. Section 2 introduces the related work, whereas Section 3 presents a brief description of the theory and the methods of DFT and digital filters, which is required to understand the rest of the article. Then, Section 4 describes LSPR in detail, and Section 5 discusses the relationships between LSPR and the principal IR models. After that, Section 6 presents the results obtained with some test collections. Finally, Section 7 presents the conclusions.

2. RELATED WORK

The work reported in this article is related to the line of research that exploits physics-inspired metaphors or mathematical formalism in domains other than physics. Examples are works carried out in recommender systems such as those explained by Lü et al. [2012]—for example, the adoption of heat diffusion equation described by Zhang et al. [2007] or mean-field approximation described by Yeung et al. [2011]. Another example

is clustering, such as approaches based on Newton's equation of motion illustrated by Blekas and Lagaris [2007] or the potential function of the Schrödinger equation explained by Weinstein and Horn [2009]. Physics-inspired retrieval models have been proposed in IR. For instance, Shi et al. [2005] proposed a retrieval model inspired by Newton's theory of gravitation. Related to this is the literature on IR system design inspired by quantum mechanics (QM) surveyed by Melucci and van Rijsbergen [2011]. The idea of using the mathematical formalism of QM in IR was introduced in van Rijsbergen [2004], in which the author mentions Fourier analysis to support the idea of using complex numbers in IR.

The most related paper to ours is that of Park et al. [2005], who proposed "spectral-based IR methods ... able to utilize many different levels of document resolution by examining the term patterns that occur in the documents." To this end, they combine the idea of multiresolution analysis properties of the wavelet transform with the idea of multilevel document resolution: under the hypothesis that a document is more relevant if the query terms are found within a small proximity to each other, it is crucial to observe the term patterns of the documents. This can be done efficiently by mapping term positions in another domain and then using the wavelet transform. However, the high precision achieved by Park et al. [2005] in comparison to the vector space and proximity retrieval methods can also be explained by the use of multilevel document resolution. This is basically passage retrieval, a methodology aimed at identifying and combining textual document passages to compute better rankings than whole textual document rankings. Passage retrieval has been found to be very effective in IR since the 1990s; a passage retrieval method identifies textual passages (e.g. phrases or textual windows) relevant to the user's information need as illustrated by numerous works (e.g., Callan [1994], Hearst [1997], Kaszkiel and Zobel [1997], Kaszkiel et al. [1999], Knaus et al. [1995], Liu and Croft [2002], Melucci [1998], Mittendorf and Schäuble [1994], and Salton et al. [1993, 1996]). Notice that our work is different from that of Park et al. [2005]: we do not move to the frequency domain to represent the term patterns of the documents. Instead, we compute a frequency representation of the query to perform the filtering operations efficiently, the latter being essential to estimate the relevance of a document.

3. BACKGROUND

This section provides the background on the analysis of digital signals useful for understanding the LSPR model presented in Section 4. Notice that we need some concepts of digital signal processing (DSP) to justify the implementation choices (sometimes based on empirical observations) of the LSPR. These concepts will be described with a level of detail that represents a trade-off between a rigorous presentation and an explanation easy to understand by a broad audience with backgrounds possibly not related to DSP. The reader interested in knowing more about this topic can consult Oppenheim et al. [1996, 1999] and Mitra [2006].

3.1. Digital Signal Processing

DSP refers to representing analog signals using sequences of numbers in the discrete domain (e.g., time, frequency, or other domains) and processing signals using algorithms that take these sequences as inputs and output numbers. Output can be either parameters of input sequences or other sequences that are transformations of the input. For example, DSP is concerned with computing the parameters of spoken text or separating noise from "true" signal. In most cases, DSP can be either one dimensional (e.g., spoken text), two dimensional (e.g., images), or three dimensional (e.g., video).

Mathematically, a signal is a function of variables and provides information about a system. This information consists of variations of variables observed from domains

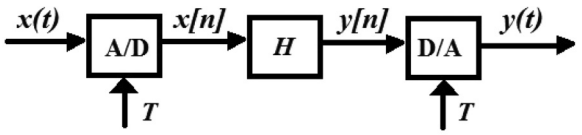


Fig. 1. Main components of DSP.

such as time and frequency. For example, speech is a continuous function of time, whereas image brightness is a function of spatial coordinates.

In this article, we are interested in discrete time signals. These signals are often obtained by sampling a continuous time signal $x(t)$ with sampling period T:

$$x[n] = x(nT), \quad n \in \mathbb{Z}.$$

As notation, in the remainder, $x(t)$ indicates continuous time signals, whereas $x(nT)$ and its square brackets version $x[n]$ represent discrete time signals. In general, DSP considers situations where a continuous time signal $x(t)$ is sampled by an analog-to-digital converter (A/D), and the output $x[n]$ is processed by a system H. The output $y[n]$ can be reconverted into a continuous time signal $y(t)$ by means of a digital-to-analog converter (D/A). This process is represented in Figure 1. Indeed, after the sampling, some information about the original signal $x(t)$ could be lost. However, under some assumptions, it is possible to reconstruct $x(t)$ from its samples. This is formalized by the following theorem (as notation, $F_s = \frac{1}{T}$ is the sampling frequency, i.e., the number of samples per unit time):[1]

THEOREM 3.1 (NYQUIST-SHANNON SAMPLING THEOREM). *Let $x(t)$ be a signal with no frequencies higher than f_M. Then $x(t)$ can be perfectly reconstructed by its samples $x(nT)$ if $F_s \geq 2f_M$.*

Very often, the systems are linear time invariant (LTI). Consider the signal $x[n] = \sum_i A_i x_i[n]$, where A_i are coefficients (complex in general). Moreover, let $y_i[n]$ be the output of the system when the input is $x_i[n]$. A system is linear if, given $x[n]$ as input, the output is $y[n] = \sum_i A_i y_i[n]$. Consider now a system where, given as input $x[n]$, the output is $y[n]$, and let $n_0 \in \mathbb{Z}$. If given as input the signal $x[n - n_0]$ the output is $y[n - n_0]$ (i.e., both input and output experience the same time shifting), the system is time invariant. For LTI systems, the relationship between input and output can be expressed in a very convenient way by operating in the frequency domain rather than in the time domain.

As the signals are processed on digital computers, a natural tendency is to develop more and more sophisticated signal processing algorithms. Often these algorithms require the signal to be transformed from its natural domain (e.g., time) to the frequency domain, and to this aim the Fourier transform can be employed [Oppenheim et al. 1996; Mitra 2006]. Consider, for example, the compression of speech: this requires the explicit computation of the inverse Fourier transform of the logarithm of the Fourier transform of the input. However, this transformation is computationally expensive, and its employment in DSP was at the beginning rather slow. The development of DSP was accelerated by the production of ad hoc hardware circuits and the discovery of an efficient algorithm for calculating Fourier transforms in discrete time. This class of algorithms, known as fast Fourier transform (FFT), drastically reduced the calculation time of the Fourier transform.

[1]This theorem actually provides a sufficient condition. If the signal $x(t)$ is *sparse*, it could be reconstructed even though the condition $F_s \geq 2f_M$ does not hold. For more information about this topic, called *compressed sensing*, see Candès et al. [2006] and Donoho [2006].

3.2. Discrete Time Sinusoidal Signals

In this article, we are interested in linear combinations of discrete time signals having the following form:

$$A_0 \sin(2\pi f_0 nT) = A_0 \sin(\omega_0 n), \quad A_0 \in \mathbb{R},\ n \in \mathbb{Z}, \tag{1}$$

which can be seen as the signal obtained by sampling (with sampling period T) a continuous time sinusoidal signal, and where f_0 is usually called *fundamental frequency*. However, the latter is periodic with period equal to 2π over its frequency, whereas for the discrete time counterpart this generally is not true. In fact, a discrete time signal is periodic if some values or sets of values constantly repeat themselves at each fixed period of length N, that is

$$x[n] = x[n+N], \quad N \in \mathbb{N}. \tag{2}$$

Applying condition (2) to (1), we get

$$\sin(\omega_0 n) = \sin(\omega_0(n+N)) = \sin(\omega_0 n + \omega_0 N),$$

and thanks to the properties of the sinusoidal signals (i.e., $\sin(x) = \sin(x + 2k\pi),\ k \in \mathbb{Z}$), it is equivalent to the condition

$$\omega_0 N = 2\pi k, \quad k \in \mathbb{Z}. \tag{3}$$

Hence, a discrete time sinusoidal signal is periodic with period $\frac{2\pi k}{\omega_0}$ only if condition (3) holds.

3.3. Discrete Fourier Transform

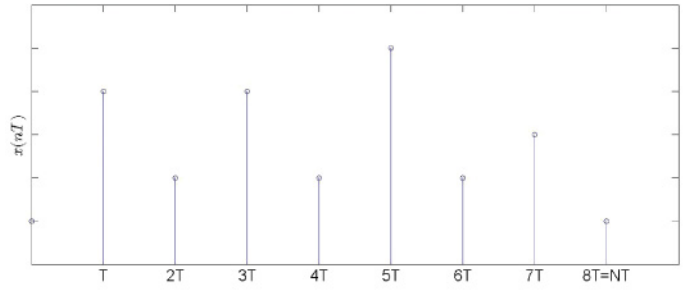
The Fourier transform is widely used in signal processing because it provides the frequency domain representation of a given signal, and this is crucial for several applications. For discrete time signals, the Fourier transform is called discrete time Fourier transform (DTFT). Its main issues are that it is a function of a continuous variable representing the frequency and has infinite duration. In fact, a transform can be carried out numerically by a computer only if it is both discrete and finite. To fix these issues, the DFT, which can be interpreted as a sampling of the DTFT, has been introduced. Let i be the symbol representing the imaginary unit $\sqrt{-1}$, and let $x[n]$ be a finite duration discrete signal such that $x[n] = 0$ for $n \notin \{1, \ldots, N\}$. This signal can be expressed using the *inverse* DFT in this way:

$$x[n] = \begin{cases} \dfrac{1}{N} \displaystyle\sum_{k=1}^{N} X[k] e^{\mathrm{i}k\frac{2\pi}{N}n} & n \in \{1, \ldots, N\}, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

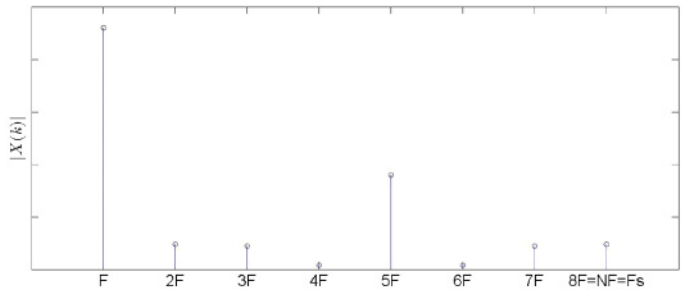
where $\frac{2\pi}{N}$ is called *fundamental angular frequency*. The DFT of $x[n]$ is defined as

$$X[k] = \begin{cases} \displaystyle\sum_{n=1}^{N} x[n] e^{-\mathrm{i}k\frac{2\pi}{N}n} & k \in \{1, \ldots, N\}, \\ 0 & \text{otherwise}. \end{cases} \tag{5}$$

The plot of the module of the DFT (i.e., $|X[k]|$), which is called *magnitude spectrum* (or simply *spectrum*), provides a representation of the frequency composition of the signal (in general, the DFT samples are complex numbers; computing the module of the DFT means substituting each complex sample with its module). An example of a signal and its spectrum, where $N = 8$, is given in Figure 2, whereas the relationships between their parameters are summarized in Table I.



(a) $x(nT)$



(b) $|X(k)|$

Fig. 2. Example of a signal $x(nT)$ and module of its DFT $|X(k)|$.

Table I. Parameters of $x[n]$ and $|X[k]|$

Feature	$x[n]$	$\|X[k]\|$
Number of samples	N	N
Distance between each sample	T	$F = \frac{1}{NT}$
Length	NT	$NF = \frac{1}{T} = F_s$

A good feature of the DFT is that it can be computed efficiently. A straightforward algorithm that computes DFT has complexity in $O(N^2)$. There exists a more efficient algorithm (or better a class of algorithms) to compute the DFT—that is, Fast Fourier Transform (FFT)—which is a "divide and conquer" algorithm with time complexity in $O(N \log N)$ for an input sequence of length N and was proposed by Cooley and Tuckey in 1965.[2] Details about its implementation can be found in the books by Cormen et al. [2000] and Press et al. [1992]. The only important detail for our implementation is that FFT requires N to be a power of two.

3.4. Spectral Leakage

Spectral leakage consists of obtaining a DFT whose spectrum does not reflect the real frequency domain representation of the signal given by the theoretical DTFT. This topic is not easy to explain, as it would require the formal definition of DTFT and the relationships between the different transforms, and this is not the goal of this work. Hence, we try to summarize it to provide a simple overview. As stated earlier, the DTFT has infinite duration because the input signal has generally infinite duration.

[2]Actually, the first who spoke of this algorithm was Gauss in 1805.

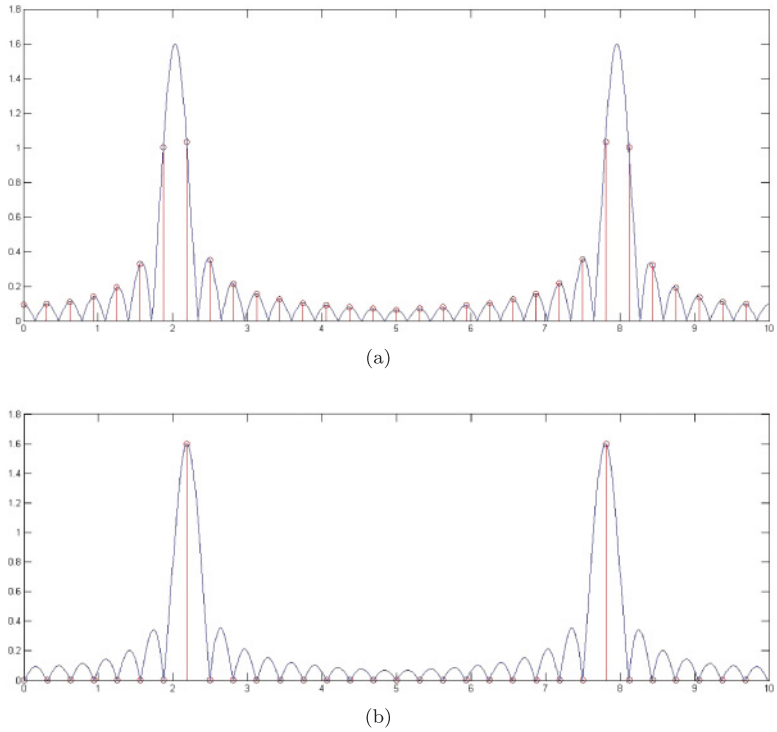


Fig. 3. Modules of *DTFT* after windowing (solid) and *DFT* (circles) with (a) and without (b) spectral leakage for a sinusoidal signal.

To represent the DTFT, only a portion of the input signal is considered. More precisely, the input signal is multiplied by a so-called window. There are many windows, the simplest of them being the rectangular one, which is a function that assumes value 1 in a given interval and 0 otherwise. A multiplication in the time domain is a convolution in the frequency domain, and hence the resulting DTFT is not the theoretical one. Considering a sinusoidal signal, its DTFT should be a function having two peaks: one corresponding to the frequency of the signal and another obtained by symmetry. Instead, the DTFT after windowing is as the solid function represented in Figure 3 (i.e., the convolution between the original DTFT and the cardinal sine function, or *sinc*, which is the frequency domain representation of the rectangular window). Using the DFT, which can be seen as a sampling of the DTFT, it is actually possible to obtain the theoretical DTFT. Consider the signal $\sin(2\pi f_0 nT) = \sin(\omega_0 n)$. Using the relationships presented in Table I, we can write

$$\omega_0 = 2\pi f_0 T = \frac{2\pi f_0}{F_s} = \frac{2\pi f_0}{NF}. \tag{6}$$

Using condition (3), we have

$$\frac{2\pi f_0 N}{NF} = 2\pi k, \quad k \in \mathbb{Z} \qquad \Rightarrow \qquad \frac{f_0}{F} = k, \quad k \in \mathbb{Z}. \tag{7}$$

In other words, if f_0 is a multiple of F, on the one hand the two nonzero peaks of the original DTFT are correctly represented, and on the other hand we eliminate the effect of the presence of the sinc function, whose value for frequencies multiple of F is zero, by construction. If this is not true, a distorted spectrum is obtained because

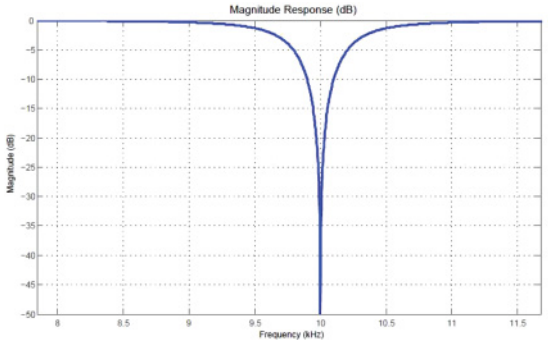


Fig. 4. Module of the transform (in logarithmic scale) for a band-rejection filter with cut-off frequency of 10kHz.

there will be DFT samples corresponding to the components introduced in the DTFT by the windowing that are not in the theoretical DTFT. This is shown in Figure 3. For more information about spectral leakage, the reader can consult Harris [1978]. It is interesting to note from Figure 3 the two features of the DFT of a sinusoidal signal, which will be used in the next section to define the LSPR model. First, it is symmetric with respect to $N/2$. Second, it presents a peak in correspondence to the frequency of the discrete time signal (if there is no spectral leakage, then the peak is the only visible component; otherwise, there are other components that decrease in amplitude when moving away from the sample corresponding to the peak).

3.5. Digital Filtering

One of the most important subjects in DSP is digital filter design. An LTI digital filter is a component (represented by H in Figure 1) that transforms an input signal $x[n]$ to an output signal $y[n]$ through the convolution of the input by an impulse response $h[n]$, which characterizes the filter. We do not provide details about convolution and how to obtain the impulse response of a filter. For our purposes, it is only important to know that convolution is quite expensive to compute, but the DFT can simplify this operation. Let $X[k]$, $Y[k]$, and $H[k]$ be the DFTs of the input signal, the output signal, and the impulse response of the filter, respectively. The relationship between these transforms can be written as

$$Y[k] = H[k] \cdot X[k], \quad k \in \{1, \ldots, N\}. \tag{8}$$

Equation (8) performs filtering with a simple multiplication between samples of the spectra. One could then use the inverse DFT to obtain $y[n]$ from $Y[k]$, and thanks to the FFT, this is more efficient than the computation of $y[n]$ using the convolution. Actually, in this article, we are not interested in finding $y[n]$, but $Y[k]$, and hence we do not need to compute the inverse DFT.

There exist many types of filters that can be used depending on the needs, such as low-pass filters (only the low-frequency components of a signal are preserved), high-pass filters (the high-frequency components of a signal are preserved), band-rejection filters (to remove a limited portion of frequency components of a signal near a given frequency called *cut-off* while preserving the rest of the spectrum almost unchanged; Figure 4). For more details about filtering, the reader can consult Oppenheim et al. [1999].

4. METHODOLOGY

4.1. Overview

According to LSPR, an IR system is viewed as an LTI DSP system that takes as input linear combinations of basic signals having the form of Equation (1), and it possesses two main properties:

(1) A basic signal represents a query term and can be used to construct an input signal that represents a query by linear combination.
(2) The response of the system should be simple enough in structure to allow us an efficient and effective implementation in terms of retrieval performance.

Let us consider the first point. Each query term is associated with a discrete time sinusoidal signal $A_i \sin(2\pi f_i nT)$, where A_i is the weight of the term i (the intensity of a burst of radiation in the metaphor) and f_i is the signal frequency (not related to the number of occurrences of the term). How the weight and the signal frequency are chosen for each term is explained in Section 4.2 (in particular, the frequency is chosen to obtain the spectral leakage). Thus, the query becomes a sum of sinusoidal signals (i.e., the input signal $x[n]$ of the IR system) for which the DFT (i.e., $X[k]$) can be computed. After the absolute value of the DFT of the query $|X[k]|$ has been computed, the spectrum (i.e., the graphical representation of $|X[k]|$) looks like a curve that has peaks of amplitude proportional to A_i around the frequency f_i and lower values for every other frequency in the neighborhood of f_i, due to spectral leakage, as depicted in Figure 3(a). The behavior of the spectrum is crucial because the utilization of a sinusoidal signal combined with the variation in the amplitude of the spectrum causes variations in retrieval effectiveness.

Now consider the second point. The system is designed in such a way that a document is viewed as a set of band-rejection filters, with one filter for each document term, and each filter should have a spectrum similar to that of Figure 4. The position of the filter is set so that if a query has a term associated with a peak in the frequency f_i, and that term occurs in a document, the corresponding filter is centered in f_i, whereas the width of the filter depends on the weight of the term in the document. Details of the parameters of the filters are described in Section 4.3.

The position of a document in the ranking list depends on a score—the power—that represents how much the filter set associated with the document is able to filter the input signal associated with the query. Hence, it is calculated after the signal has been filtered. At this point, there are two possibilities for computing the power: (i) one could either work in the discrete time domain and try to characterize the filter using the so-called impulse response $h[n]$, compute the convolution between the input signal $x[n]$ and $h[n]$ to obtain $y[n]$, and then compute the power from $y[n]$, or (ii) one could work in the frequency domain using the DFT that is obtaining the power from $Y[k]$, which can be computed as a product, sample by sample, between $X[k]$ and $H[k]$, according to Equation (8). There are two advantages of the latter method: first, we do not need to compute a convolution; second, it is easy to obtain $H[k]$, unlike $h[n]$. Moreover, to further ease the procedure, we do not use a real band-rejection filter having a spectrum like that of Figure 4, but a simplified version with triangular shape whose spectrum can be viewed in Figure 7, and hence $H[k]$ is even more simple to define. Using the frequency domain analysis, the power can be computed as the sum of the components of the filtered spectrum. Note that in DSP, the word *power* is usually related to the sum of the squares of the samples of the spectrum. We performed some experiments in order to investigate the effect of the two ways to compute the power on the effectiveness of LSPR. The obtained results, reported in Table XI of Appendix A, show that for our application the latter definition of power is less effective for estimating the relevance

of a document, due to the large number of elements in $Y[k]$ whose magnitude is lower than 1. If a document is associated with a low power, it is considered highly relevant by the system; according to the metaphor introduced in Section 1, this means that the shield represented by the document is able to attenuate the radiation effectively. Hence, the documents are ordered by increasing power. This is why the model is referred to as LSPR.

LSPR consists of three main steps:

(1) transformation of a query into a signal,
(2) transformation of a document into a band-rejection filter set, and
(3) document ranking.

Some preprocessing operations are executed before initializing LSPR. Preprocessing includes all operations usually done by every IR system, such as stemming (i.e., reduction of each term to its root, e.g., the terms *computer*, *computation*, and *computing* can be transformed in "comput") and stop-words removal (i.e., the removal of all terms of no interest in a document, e.g., prepositions, conjunctions, articles).

4.2. Query to Signal Transformation

As stated earlier, the query is associated with a signal having this form:

$$x[n] = \sum_{i=1}^{|Q|} A_i \sin(2\pi f_i n T), \quad n \in \{1, \ldots, N\}, \tag{9}$$

where $|Q|$ is the number of distinct terms in the query. Since the DFT of $x[n]$ has to be computed, the first step is to set the parameters of the DFT presented in Table I—that is, F and N, as well as the frequencies of the signals f_i and the weights A_i. We choose to set $F = 2$ Hz and to set every frequency f_i of the signals as an odd number: in this way, it is not possible for a frequency to be an integer multiple of F, and this guarantees the presence of spectral leakage since the condition of Equation (7) is not fulfilled.[3] Whereas spectral leakage should be avoided in DSP since it complicates the recognition of the correct frequencies of the signals, spectral leakage is kept in the LSPR model because the presence of some components with reduced amplitude at the frequencies close to f_i makes it possible to better estimate the relevance of a document for a given query. In fact, in LSPR, the decrease in importance for a term of the query follows the decrease in amplitude of the spectrum, so the filtering step will be more significant when affecting the frequency f_i rather than the frequencies far away from f_i: this is an important hypothesis, and the efficacy of the presented model is based on it. As previously explained, if no spectral leakage were present, the only sample with a nonzero amplitude would correspond to f_i (and the symmetric one in $NF - f_i$), thus the effect of the filtering operation would be independent of the amplitude of the filter. Instead, due to the spectral leakage of LSPR, the spectrum around f_i looks like that of Figure 6.

The choice of $F = 2$Hz allows us to write each sinusoidal signal in another way. Recalling that $T = \frac{1}{NF}$, we obtain

$$A_i \sin(2\pi f_i n T) = A_i \sin\left(\frac{\pi f_i}{N} n\right). \tag{10}$$

Regarding f_i, we associate 300 samples of the spectrum to each query, and for each term we assign to f_i a value close to the 200-th sample. These values are chosen because

[3]Usually times are expressed in seconds and frequencies in s^{-1}—that is, Hertz.

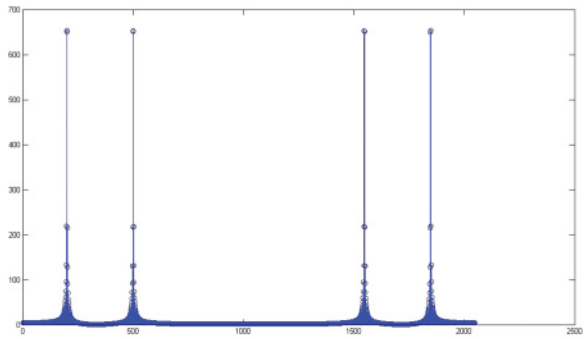


Fig. 5. Spectrum for the query of the example.

we performed some tests showing that the amplitude of the spectrum was less than 1% of the peak value (i.e., an attenuation of more than –20dB) 100 samples after and before the peak (see Figure 6), and the use of 300 samples ensures that each sinusoidal signal is related to a disjoint interval of the spectrum. In practice, to have the peak of the sinusoidal signal near to the sample 200 of the relative 300-sample interval and to guarantee spectral leakage, we set

$$f_i = F \cdot (300(i-1) + 200) + 1, \quad \forall i \in \{1, \ldots, |Q|\}. \tag{11}$$

Concerning the amplitude, A_i can be computed as a normalized version of the inverse document frequency (IDF) [Spärck Jones 1972].

Finally, consider the parameter N. The DFT of (9) is computed using N samples, where N is both the number of samples of the sinusoidal signals used as input for the DFT and the number of samples of the spectrum. As stated earlier, we need at least 300 samples for each query term. The FFT algorithm requires that N is rounded at the closest power of two, and this value has to be doubled due to the symmetry of the DFT; the latter operation also has the advantage of fulfilling the conditions of Sampling Theorem 3.1.

Now we can show with an example how this method works. Suppose that there is a query with two terms. For the sake of clarity, suppose also that the amplitudes of the two original terms (i.e., their Inverse Document Frequency (IDF) weights) are equal, and we call this value A. Since there are two terms, we need $2 \cdot 300 = 600$ samples. Then 600 is rounded to 1,024, and finally $N = 1024 \cdot 2 = 2048$. Using formula (11), the frequencies of the two signals are set to 401Hz and 1,001Hz, respectively. The sampling frequency is equal to $F_c = NF = 2048 \cdot 2 = 4096$Hz, and the maximum frequency f_M among the sinusoidal signals is 1,001Hz, so the condition of Theorem 3.1 ($F_c \geq 2 f_M$) is verified. According to Equations (9) and (10), the input signal is

$$x[n] = A\sin\left(\frac{401\pi}{2048}n\right) + A\sin\left(\frac{1001\pi}{2048}n\right), \quad n \in \{1, \ldots, 2048\}. \tag{12}$$

The FFT algorithm returns the DFT of $x[n]$, and the spectrum looks like that of Figure 5. In this figure, we can see that the spectrum is symmetric with respect to $N/2$ (sample 1,024, which corresponds to the frequency 2,048Hz; in fact, due to the symmetry, we can consider only half of the spectrum), and the peaks are near samples 200 and 500. Figure 6 shows the spectrum more clearly near sample 200.

4.3. Document to Filter Transformation

Each document is transformed into a filter set (one for each query term q_i that is also in the document). As stated before, in LSPR, each filter is similar to a "band-rejection"

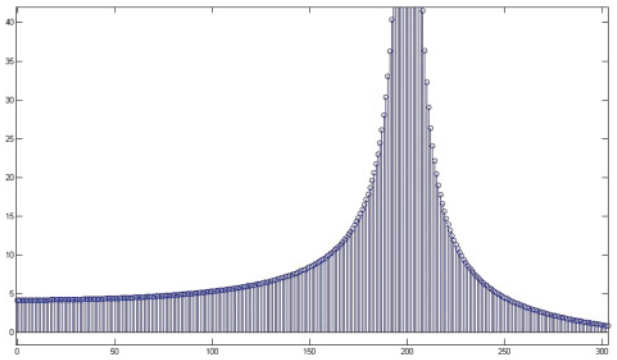


Fig. 6. Detail of the spectrum near sample 200.

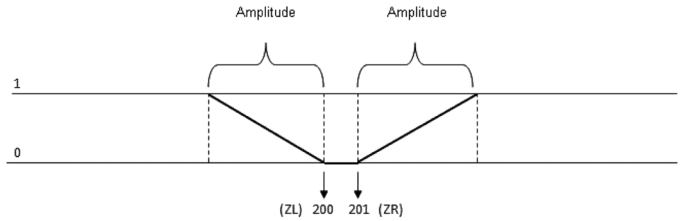


Fig. 7. Example of a filter.

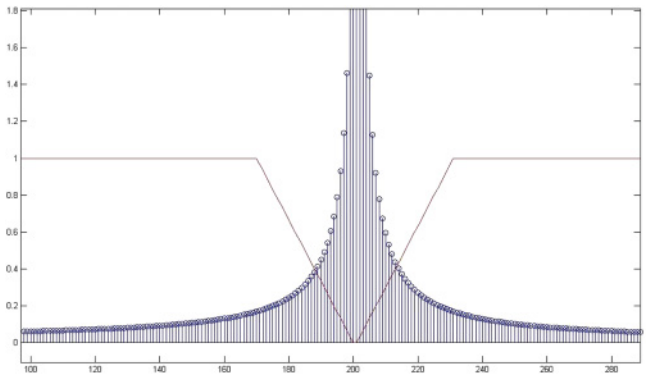


Fig. 8. A fragment of a spectrum with a filter.

filter with triangular form. To be precise, for each q_i, there are two points, Z_{L_i} (*Zero Left*) and Z_{R_i} (*Zero Right*), where the module is 0, and there is a parameter called *amplitude* such that the value of the filter before $Z_{L_i}-$amplitude and after $Z_{R_i}+$amplitude is 1. Finally, to connect Z_{L_i} (Z_{R_i}) with $Z_{L_i}-$amplitude ($Z_{R_i}+$amplitude), there is a linear function. An example of this filter is represented in Figure 7, whereas Figures 8 and 9 respectively show a fragment of a spectrum with a filter and the spectrum after filtering.

Two issues need further discussion: the selection of the amplitude (which represents the "strength" of the filter) and the position of the filter. The former is the weight of the document term multiplied by a constant called *selectivity* and then rounded to the closest integer value. As for the weight, we have employed the product of a normalized version of the IDF and the saturated term frequency (TF) in the document as adopted

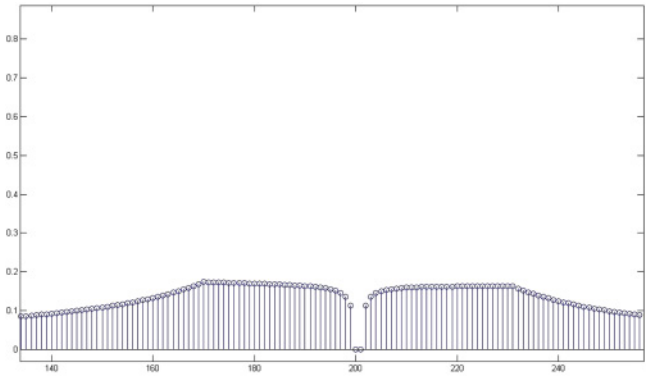


Fig. 9. The resulting filtered spectrum.

in Best Match 25 (BM25), which is one of the most effective term weighting schemes (a short description of BM25 is given in Section 6.2), but other weighting schemes can be used. In the first version of LSPR presented by Costa and Melucci [2010], we used a normalized version of the TF-IDF weighting scheme, where for each term of a given document the IDF score is multiplied by TF—that is, the number of occurrences of the term in that document. With regard to selectivity, experiments reported by Costa and Melucci [2010] identified 24 as a good value. The impact of selectivity for the efficacy of LSPR is discussed in Section 6.

Concerning the position of the filter, this is computed as follows: if the term with index i was occurring in the query, Z_{L_i} and Z_{R_i} correspond to samples 200 and 201 of the relative interval. For example, if this term is the second of the query, $Z_{L_2} = 500$ and $Z_{R_2} = 501$, whereas $Z_{L_1} = 200$ and $Z_{R_1} = 201$ for the first query term, as in Figure 7. Otherwise, the filter is not created. The choice of the values Z_{L_i} and Z_{R_i} depends on the way the frequencies of the signals are chosen. As an example, consider the first term of a query. According to Equation (11), its frequency is set to 401Hz. In the spectrum, this frequency lies between samples 200 (i.e., frequency $200F = 400Hz$) and 201 (i.e., frequency $201F = 402$Hz). Hence, the highest peaks (which should be removed by the filter) are located in correspondence to samples 200 and 201. A similar phenomenon can be seen in Figure 3(a). This can be extended to the other terms of the query: given a document, the value of Z_{L_i} for the filter corresponding to the term q_i of the query (Z_{R_i} is just computed as $Z_{L_i} + 1$) is as follows:

$$Z_{L_i} = \begin{cases} 300(i-1) + 200 & \text{if } q_i \text{ appears in the document,} \\ \text{the filter is not created} & \text{otherwise.} \end{cases} \tag{13}$$

4.4. Document Ranking

Suppose that a query ($x[n]$) is given as input to the system by the user, and consider the set of documents retrieved by the system (i.e., those with at least one term in common with the query). They are ranked by a score computed after the filtering step. In fact, for each document, the spectrum obtained by the query to signal transformation ($|X[k]|$) is filtered by the filters obtained after the document to filter step ($H[k]$), and the result is $|Y[k]|$. Using (8), we can compute $|Y[k]|$ as

$$|Y[k]| = H[k] \cdot |X[k]|, \quad k \in \{1, \ldots, N/2\}, \tag{14}$$

where $H[k] = |H[k]|$ by construction (i.e., the values $H(k)$ are nonnegative real numbers), and we consider k up to $N/2$ due to the symmetry of the DFT. Finally, the score obtained by each document is the sum of the samples of the spectrum after filtering

ALGORITHM 1: An algorithm to summarize LSPR

Input: query Q, document collection C, selectivity value.
Output: ranking list.
foreach $q_i \in Q$ **do**
    $f_i \leftarrow 2 \cdot (300(i-1)+200)+1$;
end
Compute the parameter N as explained in Section 4.2 and the input signal $x[n]$
foreach $n \in \{1,\ldots,N\}$ **do**
    $x[n] \leftarrow \sum_{i=1}^{|Q|} A_i \sin\left(\frac{\pi f_i}{N} n\right)$;
end
Compute the DFT $X[k]$ of $x[n]$, and consider for the spectrum only $k \in \{1,\ldots,N/2\}$ due to the symmetry of the DFT
spectrum $\leftarrow |X[k]|$, $k \in \{1,\ldots,N/2\}$;
foreach $q_i \in Q$ **do**
    Retrieve the posting list $L[i]$ of q_i
    $L[i] \leftarrow \{\text{documents } D_j \in C \mid D_j \bigcap q_i \neq \emptyset\}$;
    foreach $D_j \in L[i]$ **do**
        if it is the first time that the document D_j is considered **then**
            filtered_spectrum$_j \leftarrow$ spectrum;
        end
        Create a filter (equivalent to $H[k]$) for the term q_i, as described in Section 4.3, and perform filtering on $|X[k]|$
        $Z_{L_i} = 300(i-1)+200$;
        amplitude $\leftarrow$ Round(selectivity $\cdot$ weight($q_i \in D_j$));
        filter(filtered_spectrum$_j$, Z_{L_i}, amplitude);
    end
end
foreach document D_j which was processed **do**
    power$[j] \leftarrow \sum_{k=1}^{N/2}$ filtered_spectrum$_j[k]$;
end
Order the documents by increasing power and save the list in result
return result;

(the "power"), and hence every document is associated with a score. The documents are ordered by increasing power, because the more the filters decrease the power of the spectrum, the more the system considers the document and the query related. Algorithm 1 summarizes LSPR (the text delimited by $*$ is a comment).

4.5. Complexity

The complexity of the algorithm mainly depends on three operations: (i) transformation of the query into spectrum, (ii) transformation of documents into filters, and (iii) computation of filtered power and creation of the ranking list.

Concerning (i), if the query contains $|Q|$ terms, then the signal $x[n]$ can be computed in $O(N|Q|)$ steps (see Equation (9)), where N is computed as described in Section 4.2. The computation of the spectrum can be obtained in $O(N \log N)$ using the FFT algorithm. Note that from the discussion in that section, we have $|Q| < \log_2 N$. Hence, the complexity for (i) is $O(N \log N)$.

Concerning (ii), let $L[i]$ be the list of documents matching query term i. For each document in $L[i]$, a filter is created. This can be done in $O(1)$, as it is equivalent to computing Z_{L_i} and the linear function of Figure 7. Hence, filter computation requires $O(|L[i]|)$ steps for each query term q_i and (ii) requires $O(\sum_{q_i \in Q} |L[i]|)$ steps.

Table II. Normalized TF-IDF Index Matrix and IDF Weights for Query

	D1	D2	D3	Q
author	0.663	0	0	0
book	0.663	0	0	0
computer	0	0.596	0	0
data	0.245	0.220	0	0
information	0	0.440	0.136	0.585
MAP	0	0	0.367	0
precision	0	0	0.367	0
recall	0	0	0.367	0
relevance	0	0	0.735	1.585
retrieval	0.245	0	0.136	0.585
storage	0	0.596	0	0
system	0	0.220	0.136	0

Now consider (iii). Each call to the function "filter" in the algorithm is $O(1)$, because it is computed as the multiplication of the components of a filter times those of the corresponding interval in the spectrum. Since there are $O(\sum_{q_i \in Q} |L[i]|)$ calls to this function, the complexity for the filtering is $O(\sum_{q_i \in Q} |L[i]|)$. After that, the power is computed as the sum of $N/2$ values, and thus this operation is $O(N)$ for each one of the $|\bigcup_{q_i \in Q}\{L[i]\}|$ documents processed. Therefore, this operation can be done in $O(N|\bigcup_{q_i \in Q}\{L[i]\}|)$. Sorting the documents by increasing power can be done in $O(|\bigcup_{q_i \in Q}\{L[i]\}| \log(|\bigcup_{q_i \in Q}\{L[i]\}|))$. Since, in general, $|\log(|\bigcup_{q_i \in Q}\{L[i]\}|) < N$, it follows that (iii) presents a complexity of $O(\sum_{q_i \in Q} |L[i]| + N|\bigcup_{q_i \in Q}\{L[i]\}|)$.

The total complexity of the algorithm (without taking into account indexing, stemming, and stop-words removal) is $O(N \log N + \sum_{q_i \in Q} |L[i]| + N|\bigcup_{q_i \in Q}\{L[i]\}|)$. Note that some data of the filters that correspond to the documents can be precomputed at indexing time and stored in the posting lists, thus reducing the actual computation time. Moreover, some optimizations of the index operations can limit the total number of postings to a maximum (e.g., 1,000) number m of documents retrieved, thus obtaining that the total complexity is $O(N \log N + |Q|m + N|\bigcup_{q_i \in Q}\{L[i]\}|)$.

Note that this is the complexity when considering the basic operations (e.g., sum, multiplication) performed in the central memory. Considering the external complexity, the LSPR requires access to the index $O(|Q|)$ times for retrieving the posting lists.

4.6. Toy Example

To give a global idea about how LSPR works, we present an example with a collection of three documents and a query. Suppose that we have the following documents in the collection (for the sake of clarity, we consider them after stop-words removal and without stemming):

—*D1*: (retrieval, data, author, book),
—*D2*: (information, computer, system, storage, information, data), and
—*D3*: (information, retrieval, system, relevance, MAP, precision, recall, relevance).

Suppose also that the query is *Q:* (information, retrieval, relevance). Table II reports the index of the collection with the normalized TF-IDF weighting scheme (even though, as stated earlier, we actually use the BM25 in our implementation) and the query with the IDF weighting scheme. At this point, we show how LSPR creates the spectrum of the query, as described in Section 4.2. First, the frequencies of the terms *information*, *retrieval*, and *relevance* are respectively set to 401Hz, 1,001Hz, and 1,601Hz, whereas

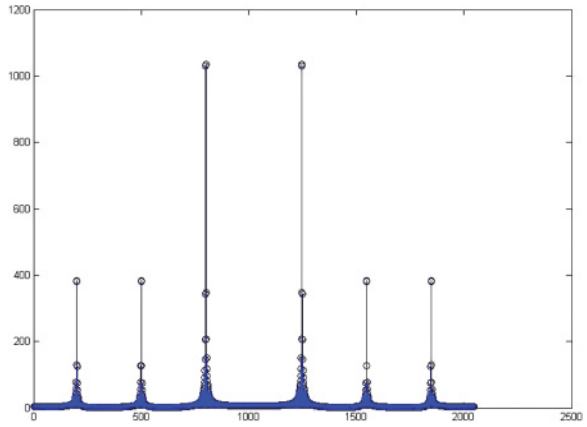
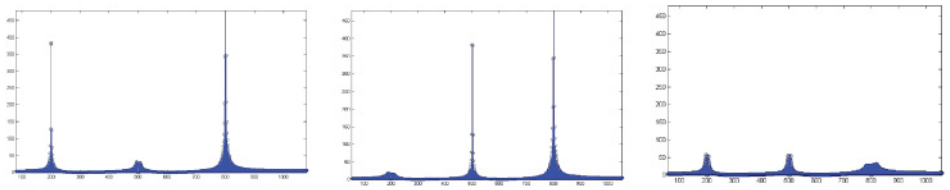


Fig. 10. Spectrum for the query.



(a) D1 (b) D2 (c) D3

Fig. 11. Spectra after the filtering operations performed on the three documents.

N is 2,048. Now, $x[n]$ is computed as

$$x[n] = 0.585 \sin\left(\frac{401\pi}{2048}n\right) + 0.585 \sin\left(\frac{1001\pi}{2048}n\right) + 1.585 \sin\left(\frac{1601\pi}{2048}n\right), \quad n \in \{1, \ldots, 2048\}. \tag{15}$$

The signal $x[n]$ is given as input to the FFT algorithm, and we obtain the spectrum shown in Figure 10 (as stated earlier, due to the symmetry, we actually consider only half of the spectrum). Now the documents are transformed into filters, as explained in Section 4.3. We adopt a value of selectivity of 24 in this example. Since D1 contains only the second term of the query (i.e., retrieval), with weight 0.245, it is transformed into a filter with $Z_{L_2} = 500$ and with amplitude = Round $(24 \cdot 0.245) = 6$. Similarly, D2 corresponds to a filter (information) with $Z_{L_1} = 200$ and amplitude = Round $(24 \cdot 0.440) = 11$. Finally, D3 is transformed into a set of three filters; the first one (information) has $Z_{L_1} = 200$ and amplitude = Round $(24 \cdot 0.136) = 3$, the second one (retrieval) has $Z_{L_2} = 500$ and amplitude = Round $(24 \cdot 0.136) = 3$, and the third one has $Z_{L_3} = 800$ and amplitude = Round $(24 \cdot 0.735) = 18$. At the end, after the filtering operations on the spectrum represented in Figure 10, we obtain the filtered spectra of Figure 11.

The power of the spectrum of Figure 10 is 13007.091, whereas the powers of the spectra of Figure 11 are 11836.613, 11649.498, and 6919.414, respectively. Thus, according to the ranking rule of LSPR, the first document in the ranking list is D3, the second is D2, and the third is D1.

5. RELATIONSHIPS WITH THE MAIN IR MODELS

The first part of this section explains the differences and the second part explains the connections between the LSPR and other models (i.e., the VSM, the binary retrieval model (BRM), and the query-likelihood language model (QLM)).

5.1. Differences between LSPR and Other Models

With regard to the differences between the models, consider the innermost cycle of Algorithm 1. When iterating over the postings of the list retrieved for query term i, a filter is created for each document of the posting list as explained in Section 4.3. The creation of a filter depends on some parameters and design decisions. These parameters and design decisions permit the implementation of different filters, which can then yield different results. Note that filtering consists of a product between the components of the filter (associated to the document) and those of the spectrum of the signal computed by DFT and associated to the query. It is the implementation of a filter that can make filtering different from other IR models. These models are indeed based on computing the sum of the products between the query term weights and the document term weights—that is, sum of real numbers provided by weighting schemes that in turn are derived from probability theory, linear algebra, or information theory. In the LSPR model, the real numbers are not directly provided by these weighting schemes—they instead are provided by techniques made available by the theory and the methodology of DSP. Techniques other than those used in the article might be used to implement the filters. Such a "degree of freedom" might make the LSPR model potentially advantageous, as it would allow a researcher to design and experiment alternative filter designs and other mechanisms to define the signal $x[n]$ of a query. This can be facilitated by the connections between the LSPR and other IR models highlighted in the rest of the section—for example, the correspondence between signal and probability distribution that is highlighted in the following might suggest using probability functions to define the frequency components $X[k]$.

In Fang et al. [2011], an axiomatic framework was introduced to analyze the causes of success and failure of an IR model. This framework states some reasonable constraints on some observables (e.g., TF in a collection) that make a retrieval function of a classical model effective only if they are valid; these constraints should indeed be necessary conditions according to Fang et al. [2011]. Other frameworks were proposed in IR literature to compare retrieval models; an example is the framework illustrated in Roelleke's work [Roelleke 2013].

In the following, we argue that the axiomatic framework cannot straightforwardly be applied to the LSPR model and that the weights commonly used in the classical models have more complex effects. In particular, the weight of a term of the query has an impact on parts of the filtered spectrum related to other terms. To this end, we describe and analyze an example of the LSPR model in the axiomatic framework. For $k \in \{1, \ldots, N/2\}$, filtering is performed by

$$Y[k] = H[k] \cdot X[k].$$

Note that we do not consider the computation of the module, as it would make the description more difficult. To compute $H[k]$ for a generic query term q, we have two cases: $Z_{Lq} - \text{amplitude} \leq k \leq Z_{Lq}$ and $Z_{Rq} \leq k \leq Z_{Rq} + \text{amplitude}$ (out of these bounds $H[k] = 1$, as shown in Figure 7). Consider the first one (a similar result hold for the other case). When $Z_{Lq} - \text{amplitude} \leq k \leq Z_{Lq}$, the filter associated to a query term q is a line $H_{Lq}[k] = z_1 k + z_0$ described by

$$z_1 = \frac{1 - 0}{(Z_{Lq} - \text{amplitude}) - Z_{Lq}} = -\frac{1}{\text{amplitude}},$$

$$z_0 = H_{Lq}[k] - z_1 k = 0 - \left(-\frac{1}{\text{amplitude}}\right) = Z_{Lq} = \frac{Z_{Lq}}{\text{amplitude}},$$

where z_1 was obtained by using the points $(Z_{Lq} - \text{amplitude}, 1)$ and $(Z_{Lq}, 0)$ and z_0 using the resulting z_1 and the point $(Z_{Lq}, 0)$. Therefore, for a generic point $k : Z_{Lq} - \text{amplitude} \leq k \leq Z_{Lq}$ (i.e., k is on the left part of the filter), we have

$$H_{Lq}[k] = -\frac{1}{\text{amplitude}}k + \frac{Z_{Lq}}{\text{amplitude}} = \frac{Z_{Lq} - k}{\text{amplitude}}.$$

Let us consider a query with two terms: $Q = \{q_1, q_2\}$. Then

$$x[n] = A_1 \sin\left(\frac{f_1\pi}{N}n\right) + A_2 \sin\left(\frac{f_2\pi}{N}n\right),$$

where A_t is the weight associated to the term q_t. Once the DFT has been applied, we obtain

$$\begin{aligned}
X[k] &= \sum_{n=1}^{N} x[n] e^{-ik\frac{2\pi}{N}n} \qquad k \in \{1, \ldots, N\} \\
&= \sum_{n=1}^{N} \left[A_1 \sin\left(\frac{f_1\pi}{N}n\right) + A_2 \sin\left(\frac{f_2\pi}{N}n\right)\right] e^{-ik\frac{2\pi}{N}n} \\
&= \sum_{n=1}^{N} \left[A_1 \left(\frac{e^{if_1\frac{\pi}{N}n} - e^{-if_1\frac{\pi}{N}n}}{2i}\right) + A_2 \left(\frac{e^{if_2\frac{\pi}{N}n} - e^{-if_2\frac{\pi}{N}n}}{2i}\right)\right] e^{-ik\frac{2\pi}{N}n} \\
&= \frac{1}{2i}\sum_{n=1}^{N} \left[A_1 e^{if_1\frac{\pi}{N}n} e^{-ik\frac{2\pi}{N}n} - A_1 e^{-if_1\frac{\pi}{N}n} e^{-ik\frac{2\pi}{N}n} + A_2 e^{if_2\frac{\pi}{N}n} e^{-ik\frac{2\pi}{N}n} - A_2 e^{-if_2\frac{\pi}{N}n} e^{-ik\frac{2\pi}{N}n}\right] \\
&= \frac{1}{2i}\sum_{n=1}^{N} \left[A_1 e^{-i(2k-f_1)\frac{\pi}{N}n} - A_1 e^{-i(2k+f_1)\frac{\pi}{N}n} + A_2 e^{-i(2k-f_2)\frac{\pi}{N}n} - A_2 e^{-i(2k+f_2)\frac{\pi}{N}n}\right].
\end{aligned}$$

Now suppose that we want to compute $Y[k]$ for a point $k : Z_{Lq_1} - \text{amplitude} \leq k \leq Z_{Lq_1}$ (i.e., the value of the filtered spectrum for a position k on the left side of the peak corresponding to the first query term but whose distance from the peak is smaller than or equal to the amplitude):

$$\begin{aligned}
Y[k] = H[k] \cdot X[k] &= H_{Lq_1}[k] \cdot X[k] \\
&= \frac{1}{2i}\sum_{n=1}^{N} \left[\left(\frac{Z_{Lq_1} - k}{\text{amplitude}_1}\right) A_1 e^{-i(2k-f_1)\frac{\pi}{N}n} \right. \\
&\quad - \left(\frac{Z_{Lq_1} - k}{\text{amplitude}_1}\right) A_1 e^{-i(2k+f_1)\frac{\pi}{N}n} \\
&\quad + \left(\frac{Z_{Lq_1} - k}{\text{amplitude}_1}\right) A_2 e^{-i(2k-f_2)\frac{\pi}{N}n} \\
&\quad \left. - \left(\frac{Z_{Lq_1} - k}{\text{amplitude}_1}\right) A_2 e^{-i(2k+f_2)\frac{\pi}{N}n}\right].
\end{aligned}$$

Recall that the amplitude is the product of a weight (e.g., BM25) by a selectivity σ. The amplitude for the filter associated to the first filter can be expressed as

$$\text{amplitude}_1 = \sigma \text{TF}_{1j} A_1,$$

where TF_{1j} is the saturated document TF of the first query term in document j and A_1 is the IDF of the first query term. More precisely, it is the IDF (normalized over the maximum IDF value) used in BM25 or in TF-IDF. We can now rewrite the previous expression as

$$\begin{aligned} Y[k] \;=\; \frac{1}{2i} \sum_{n=1}^{N} \Bigg[& \left(\frac{Z_{Lq_1} - k}{\sigma \text{TF}_{1j} A_1} \right) A_1 e^{-i(2k - f_1)\frac{\pi}{N} n} \\ & - \left(\frac{Z_{Lq_1} - k}{\sigma \text{TF}_{1j} A_1} \right) A_1 e^{-i(2k + f_1)\frac{\pi}{N} n} \\ & + \left(\frac{Z_{Lq_1} - k}{\sigma \text{TF}_{1j} A_1} \right) A_2 e^{-i(2k - f_2)\frac{\pi}{N} n} \\ & - \left(\frac{Z_{Lq_1} - k}{\sigma \text{TF}_{1j} A_1} \right) A_2 e^{-i(2k + f_2)\frac{\pi}{N} n} \Bigg]. \end{aligned}$$

Let us denote

$$a_1 = \frac{Z_{Lq_1} - k}{\sigma \text{TF}_{1j}}.$$

Now we obtain the following expression:

$$Y[k] \;=\; \frac{1}{2i} \sum_{n=1}^{N} \left[a_1 e^{-i(2k - f_1)\frac{\pi}{N} n} - a_1 e^{-i(2k + f_1)\frac{\pi}{N} n} + \frac{a_1 A_2}{A_1} e^{-i(2k - f_2)\frac{\pi}{N} n} - \frac{a_1 A_2}{A_1} e^{-i(2k + f_2)\frac{\pi}{N} n} \right].$$

We can rewrite this equation as follows:

$$Y[k] = \frac{a_1}{2i} \sum_{n=1}^{N} \left[e^{-i(2k - f_1)\frac{\pi}{N} n} - e^{-i(2k + f_1)\frac{\pi}{N} n} + \frac{A_2}{A_1} \left(e^{-i(2k - f_2)\frac{\pi}{N} n} - e^{-i(2k + f_2)\frac{\pi}{N} n} \right) \right]. \tag{16}$$

It follows that the constraints of the axiomatic framework cannot straightforwardly be applied to the LSPR model. Indeed, an increment of TF_{1j} decrements the term a_1, thus decrementing the power in line with the axiomatic framework stating that an increment of a within-document frequency should promote a document in a ranked document list. The effects of the variations of σ should be similar to those caused by TF_{1j}; however, the experiments reported in Section 6 suggest that these effects are less obvious than those suggested by the mathematical analysis. The behavior of the IDF weighting is less obvious. On the one hand, an increase of A_1 would decrease the power and consequently push the document toward higher positions in the ranking list. On the other hand, an increase of A_2 would have the opposite effect. In addition, to complicate the matter, the effect of this variation depends on the frequencies f_i associated to the terms, and the computation of the module of $Y[k]$ (which is needed by LSPR to calculate the power) makes the expression (16) even more complicated. Moreover, this effect depends on the section of the spectrum considered, and a query consisting of more terms would complicate the analysis even more.

5.2. Connections between LSPR and Other Models

The formulation of the DFT provided in Section 3 allows us to make a connection with three major IR models: the Vector Space Model (VSM), the probabilistic BRM, and the Query-likelihood Language Model (QLM).

First, note that a discrete time signal can be expressed using the inverse DFT, which can be also rewritten in a vectorial form, thus Equation (4) can be compacted as follows:

$$\mathbf{x} = \begin{pmatrix} x[1] \\ \vdots \\ x[N] \end{pmatrix}. \tag{17}$$

This vector can be expressed as a linear combination of basis vectors as follows:

$$\mathbf{x} = \frac{1}{N} \sum_{k=1}^{N} X[k] \mathbf{W}_{Nk}, \tag{18}$$

where

$$\mathbf{W}_{Nk} = \begin{pmatrix} e^{\mathrm{i}k\frac{2\pi}{N}} \\ \vdots \\ e^{\mathrm{i}k\frac{2\pi}{N}n} \\ \vdots \\ e^{\mathrm{i}k2\pi} \end{pmatrix}. \tag{19}$$

Now consider the VSM. In this IR model, each weight is a measure of the importance of an index term in a document or a query, respectively. The index term weights are computed on the basis of the frequency of the index terms in the document, the query, or the collection. At retrieval time, the documents are ranked by a function of the inner product (e.g., the cosine of the angle) between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine (see Melucci [2009] for an example). In mathematical terms, the VSM can be expressed as follows:

$$\mathbf{x} = \sum_{k=1}^{N} x_k \mathbf{e}_k, \tag{20}$$

where $\mathbf{x}$ is a vector of N real numbers that represents a query, a document, or a passage; x_k is the weight of term k; and $\mathbf{e}_k$ is a basis vector that represents a term. In practice, the basis is the canonical basis of a vector space—that is,

$$e_{k,j} = \begin{cases} 1 & k = j, \\ 0 & k \neq j, \end{cases}$$

where $e_{k,j}$ is the j-th element of $\mathbf{e}_k$. In other words, the k-th canonical basis vector has null elements except the k-th element, which is 1. It follows that when $\mathbf{x}$ is the query vector and $\mathbf{h}$ is a document vector,

$$\mathbf{x} = (x_1, \ldots, x_N)^\top, \qquad \mathbf{h} = (h_1, \ldots, h_N)^\top, \qquad \mathbf{x} \cdot \mathbf{h} = \sum_{k=1}^{N} x_k h_k. \tag{21}$$

The relationship with LSPR can now be pointed out. In VSM, the query and the documents can be seen as vectors having the form of (20), whereas in LSPR, the query and the documents can be represented using the inverse DFT, whose vectorial form is as in (18). Note the strong relationships between these two equations: the term x_k in the VSM plays the role of $X[k]$ in LSPR. The ranking in the VSM is performed by computing the cosine of the angle between the vectors of the query and the document, respectively, according to Equation (21), and then ordering the documents by decreasing value of the cosine. In LSPR, the corresponding operation is the computation of the power. As a matter of fact, the product $x_k h_k$ in the VSM is similar to the product $|X[k]|H[k]$ of Equation (14) in the LSPR (where we use the module because in LSPR we have in general complex numbers, unlike VSM), and the sum over k of this quantity gives the power (for the symmetry, we can consider the sum up to $N/2$, but we would get the same ranking if considering the sum up to N).

As for the BRM, consider the notion of characteristic function of a probability function associated with a sample space. For the sake of clarity, consider a sample space of N binary random variables such that a random variable corresponds to the occurrence of an index term; this space is common in IR. Suppose that $p(\mathbf{z}) = P(\mathbf{Z} = \mathbf{z})$ is the probability that the binary random vector $\mathbf{z}$ has been observed such that $z_k = 1$ when term k occurs and 0 otherwise. The characteristic function of $\mathbf{Z}$ with probability function $p(\mathbf{z})$ is defined as follows:

$$\Phi[n] = \sum_{z_1 \in \{0,1\}} \cdots \sum_{z_N \in \{0,1\}} e^{\mathrm{i}\mathbf{z}n} p(\mathbf{z}), \tag{22}$$

which is actually the expectation of $e^{\mathrm{i}\mathbf{z}n}$ computed over $p(\mathbf{z})$. Comparing (22) to (18), the correspondence with LSPR is established by $\Phi[n]$, which can be viewed as a signal; $e^{\mathrm{i}\mathbf{z}n}$, which corresponds to $\mathbf{W}_{Nk}$; and $p(\mathbf{z})$, which plays the role of the frequency components $X[k]$.

As for the QLM, the notion of characteristic function is applied to a sample space where the random variables are no longer binary and are in contrast term frequencies. In this model, the probability that a query term occurs is given by

$$\Pr(Q|B) = p_B(w_{(1)} \dots w_{(N)}) \tag{23}$$

$$= p_B(w_{(1)}) p_B(w_{(2)}|w_{(1)}) \cdots p_B(w_{(N)} \mid w_{(N-1)} \cdots w_{(1)}), \tag{24}$$

where B indicates a source of evidence or context used for estimation purposes. Indeed, query term probability estimation is implemented by

$$P(Z_i = 1) = \frac{f(w_{(i)}, B)}{\sum_{i=1}^{N} f(w_{(i)}, B)}, \tag{25}$$

where the f's refer to frequencies within B, or by

$$P(Z_i = 1) = (1 - \lambda) \frac{f(w_{(i)}, B)}{\sum_{i=1}^{N} f(w_{(i)}, B)} + \lambda \frac{f(w_{(i)}, \mathcal{V})}{\sum_{i=1}^{N} f(w_{(i)}, \mathcal{V})}, \tag{26}$$

where B has been implemented as a document and integrated by the collection vocabulary $\mathcal{V}$. Using this estimation, we have that

$$p(\mathbf{z}) = P(Z_1 = z_1) \cdots P(Z_N = z_N).$$

The characteristic function is then computed using (22), and the correspondence with LSPR is thus established.

There are some connections between LSPR and the information-based models and term burst-based models for ad hoc IR. With regard to the connections with the information-based models, the reader can refer to the Clinchant and Gaussier [2010]. As for the divergence from randomness (DFR) model [Amati and van Rijsbergen 2002], these probabilistic models, which rank documents by a measure of probability or a function thereof, can be connected to LSPR on the basis of the connection between the characteristic function of a probability distribution and the DFT of a signal, as explained in this section.

6. EXPERIMENT

6.1. Research Questions

The experiments reported in this section are a proof of concept to demonstrate that the LSPR retrieval model can be used in ad hoc settings—that is, LSPR is comparable to the state-of-the-art. Therefore, the main research question is:

i. When considering an ad hoc setting, what is the effectiveness of the LSPR retrieval model when compared to state-of-the-art retrieval models?

To implement the LSPR retrieval model, we need to specify how to compute the amplitude of the peaks in the spectrum, how to compute the amplitude of the filter associated to a document, and how to set the selectivity value. Section 6.2 describes how the first two components were computed in the experiments by using estimates adopted by state-of-the-art retrieval models. With regard to selectivity, we considered a range of values to address the following research questions:

ii. What is the effect of the selectivity values on retrieval effectiveness?
iii. Is there a maximum value for the function that relates selectivity to retrieval measures?
iv. Is the effect of the selectivity value dependent on the adopted test collection?

Section 6.2 describes the experimental methodology, and Section 6.3 describes the test collections and the retrieval measures adopted to investigate the preceding research questions.

6.2. Experimental Methodology

To gain some insight into the first research question, we compared the LSPR effectiveness with that of one of the state-of-the-art retrieval models. The baseline adopted in our experiments is the BM25 weighting scheme described in Robertson and Zaragoza [2009]. A brief description is reported in the following.

6.2.1. Baseline. Let V_D be the set of terms appearing in document D; the weight w_i assigned to the term $t_i \in V_D$ is

$$w_i = \frac{TF'_i}{k_1 + TF'_i} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right), \tag{27}$$

where N is the total number of documents in the collection, n_i is the number of documents in the collection where the term t_i appears, and k_1 is a hyperparameter. The quantity TF'_i is defined as $TF'_i = TF_i/B$, where TF_i is the TF of t_i, and

$$B = (1 - b) + b\frac{dl}{avdl}, \tag{28}$$

where b is an hyperparameter, $dl = \sum_{t_i \in V_D} TF_i$ is the document length and $avdl$ is the average document length in the collection.

We optimized the hyperparameters of BM25 (i.e., b and k_1) using two different methods for the parameter optimization:

—overfitting for each test collection over the entire topic set—that will constitute an *upperbound* for the Best Match 25 (BM25) effectiveness for the specific test collection; and
—for a given test collection, the entire topic set is split in two subsets at random; the first split is adopted for optimizing the hyperparameters, and the second split is adopted to test the BM25 with the most effective hyperparameter values obtained in the training set for a specific effectiveness measure—the measures adopted are listed in Section 6.3.

For both the methods, we performed exhaustive search in the two-dimensional hyperparameter space considering the range $0 \leq b \leq 1$ with a step of 0.01 and the range $0 \leq k_1 \leq 10$ with a step of 0.1; the range for k_1 was chosen according to the approach adopted in Taylor et al. [2006].

6.2.2. LSPR Implementation. The LSPR retrieval model implementation adopted in the experiments exploits the IDF of a query term q_i to estimate the amplitude of the peak in the spectrum corresponding to that term—that is,

$$A_i = \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right). \tag{29}$$

The amplitude of the filter is computed as

$$\text{amplitude}_i = \text{selectivity} \cdot \frac{TF'_i}{k_1 + TF'_i} \cdot \frac{IDF_i}{\max_{q_j} IDF_j}. \tag{30}$$

Therefore, the current implementation involves three hyperparameters: b and k_1, which are shared with the BM25, and the selectivity value. The values of b and k_1 that maximize the effectiveness of BM25 have been used to evaluate and compare LSPR. The "best" values of b and k_1 for BM25 do not immediately correspond to the best values of b and k_1 for LSPR since the latter has a third parameter—selectivity. Finding the best triple (b, k_1, selectivity) requires an exhaustive search of the parameter space. The problem is that an exhaustive search of the parameter space is extremely expensive. The parameter space is spanned by

—b, which ranges from [0.00, 1.00] and has 100 values;
—k_1, which ranges from [0.0, 10.0] and has 100 values;
—selectivity, which ranges from [1, 200] and has 200 values;
—two training/test methods (random split and upperbound);
—two optimization measures (i.e., mean average precision (MAP) and normalized discounted cumulative (NDCG); see Section 6.3);
—two test collections (see Section 6.3).

The total number of points is thus $2^4 10^6$. As each point corresponds to one run, the total time needed to perform an exhaustive search is prohibitive.[4] Therefore, we had to set (b, k_1) to the best values for BM25 and to tune selectivity. The selectivity value was optimized by performing exhaustive search search in the range [1, 200]; the upper bound 200 was chosen to avoid the filter associated with a specific term of the document

[4]If 1 minute were taken to perform one run, $2^4 10^6$ would be about 267,000 hours (i.e., more than 11,000 days). The number of points can be reduced by a factor of 2 by running the models only for the "upperbound" case and then computing the optimal parameter values for the "random split" case by postretrieval analysis. In this case, the number of points will be $2^3 10^6$, about 134,000 hours (i.e., more than 5,500 days).

Table III. Description of the Test Collections Adopted in the Experiments

Test Collection	Corpus	Number of Documents	Topic Sets
robust04		528,155	301–450; 600–700
	Financial Times (FT)	210,158	
	Federal Register (FR)	55,630	
	Foreign Broadcast Information Service (FBIS)	130,471	
	LA Times (LATIMES)	131,896	
wt00-01	WT10g	1,692,096	451–550

to attenuate components of the spectrum corresponding to other terms; as mentioned in Section 4.2, we observed that the amplitude of the spectrum was less than 1% of the peak value 100 samples after and before the peak.

The diverse weighting schemes adopted in this article have been implemented on the top of the open source library Apache Lucene.[5] For both BM25 and LSPR, Porter stemming was adopted and we performed stop-words removal by the default Lucene stop set.

6.3. Test Collections and Effectiveness Measures

The experiments were carried out on two test collections: the TREC Web 2000–2001 test collection and the TREC Robust 2004. Information on the adopted test collection is reported in Table III. The first two columns respectively report the name of the test collection and its constituting document corpora; for each document corpus, the number of documents is reported in the third column. The fourth column reports the set of topics adopted in each test collection. The number of documents m per run is that requested by the track organizers for computing the effectiveness measures; for both tracks, $m = 1{,}000$.

To measure the effectiveness of the LSPR model to rank highly relevant documents at high-rank positions, we adopted the Normalized Discounted Cumulative Gain (NDCG) measure illustrated by Järvelin and Kekäläinen [2002], specifically the variant adopted in the TREC2010 Web Track: NDCG@k = DCG@k/IDCG@k, where

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{(2^{rel_i} - 1)}{\log_2(i+1)}$$

and IDCG@k is the DCG@k computed on the top k relevant documents in the pool, ranked by their (descending) degree of relevance; rel_i denotes the relevance grade of documents at the i-th position. This variant was adopted because it puts a stronger emphasis than the original definition on retrieving highly relevant documents at high-rank positions, as Croft et al. [2009] explained. In the remainder of this article, we will denote with NDCG the case when the value of k is set to the run size—that is, $k = m = 1{,}000$.

To gain insight into the effectiveness of the LSPR model when a recall-oriented measure is adopted, we utilized the Mean Average Precision (MAP) (see Croft et al. [2009] for a definition). The reason for considering a recall-oriented measure is that if for a given number m of documents retrieved LSPR is able to provide more relevant documents than BM25, the former can be a more effective weighting scheme to adopt in the first stage of learning to rank models that operate in a two-stage fashion [Liu 2009]. The first stage usually involves a bag-of-words method and aims at retrieving a first set of documents that are good candidates for satisfying the user information

[5]The actual version adopted is the 4.7.2; the BM25 was reimplemented with a custom similarity that saves document lengths as integer to avoid possible effect due to the way Apache Lucene handles the document lengths by default—an approximation of the lengths is stored in the norm of the document field.

Table IV. Comparison in Terms of MAP between BM25 and LSPR

				BM25		LSPR		
Test Collection	Hyperparamer Tuning	b	k_1	training	test	test	selectivity	p-value
robust04	random training/test (d)	0.75	1.2	0.2209	0.2568	0.2626	17	0.346
	random training/test	0.30	0.7	0.2407	0.2749	0.2668	27	0.199
	upperbound (d)	0.75	1.2	—	0.2353	0.2405	14	0.160
	upperbound	0.30	0.7	—	0.2544	0.2474	64	**0.001**
wt00-01	random training/test (d)	0.75	1.2	0.1573	0.1853	0.1872	25	0.873
	random training/test	0.35	1.2	0.1797	0.2133	0.2068	19	0.631
	upperbound (d)	0.75	1.2	—	0.1716	0.1803	90	0.245
	upperbound	0.33	0.9	—	0.1990	0.1863	21	**0.045**

Table V. Comparison in Terms of NDCG between BM25 and LSPR

				BM25		LSPR		
Test Collection	Hyperparameter Tuning	b	k_1	training	test	test	selectivity	p-value
robust04	random training/test (d)	0.75	1.2	0.4917	0.5146	0.5207	21	0.297
	random training/test	0.22	1.1	0.5101	0.5351	0.5234	70	**0.016**
	upperbound (d)	0.75	1.2	—	0.5009	0.4988	21	0.590
	upperbound	0.22	1.1	—	0.5201	0.5071	25	**<0.001**
wt00-01	random training/test (d)	0.75	1.2	0.4082	0.4521	0.4353	16	0.306
	random training/test	0.29	1.9	0.4450	0.4813	0.4488	8	**0.017**
	upperbound (d)	0.75	1.2	—	0.4306	0.4256	30	0.627
	upperbound	0.40	1.1	—	0.4667	0.4377	17	**0.003**

need. The second stage involves more computationally intensive retrieval algorithms that exploit a high number of features to re-rank the documents to present the user with highly relevant documents at high-rank positions.

As mentioned in Section 6.2, since both retrieval models required hyperparameter optimization, we considered two different settings. In the first setting, in the remainder of this article denoted by *random training/test*, the topic set was split in two parts: we consider a 60%/40% (training/test) split for the *robust04* test collection and a 50%/50% split for the *wt00-01* test collection, following the approach adopted in Metzler and Croft [2007], but considering a random split instead of a sequential split.[6] In the second setting, in the remainder of this article denoted by *upperbound*, we used the entire topic set to perform training, and the optimal hyperparameter values (for LSPR the optimal selectivity value) were then selected. The optimization was performed for both the selected effectiveness measures (i.e., NDCG and MAP).

6.4. Discussion of the Results

The obtained results are reported in Table IV for MAP and in Table V for NDCG. The first column reports the label of the adopted test collection. The second column reports the method adopted for hyperparameter optimization. The third and the fourth columns report the most effective values of b and k_1 observed for the BM25 in the training set and then adopted in the test set. The fifth and the sixth columns refer to BM25 and respectively report the value of the considered effectiveness measure in the training set and in the test set when the optimal values of b and k_1 were adopted; in the *upperbound* setting, being training and test set the same, the values obtained in the training set are not reported. The next two columns refer to LSPR: the seventh column reports the value of the considered effectiveness measure in the test set, when

[6]In Metzler and Croft [2007], when considering a x%/(100-x)% split, the first x% topics ordered by topic identifier were adopted for training and the remaining topics were adopted for test.

Table VI. MAP for LSPR Instantiation with Selectivity Set to 100

Test Collection	Hyperparameter Tuning	b	k_1	BM25	$\text{LSPR}_{\text{BM25sOpt}}$	$\text{LSPR}_{\text{BM25s100}}$
robust04	random training/test (d)	0.75	1.2	0.2568	0.2626	0.2625
	random training/test	0.30	0.7	0.2749	0.2668	0.2688
	upperbound (d)	0.75	1.2	0.2353	0.2405	0.2379
	upperbound	0.30	0.7	0.2544	0.2474	0.2463
wt00-01	random training/test (d)	0.75	1.2	0.1853	0.1872	0.1977
	random training/test	0.35	1.2	0.2133	0.2068	0.2073
	upperbound (d)	0.75	1.2	0.1716	0.1803	0.1774
	upperbound	0.33	0.9	0.1990	0.1863	0.1846

the optimal b and k_1 values for the BM25 are used, and for the most effective selectivity value; selectivity is optimized in the training set, and the optimal value is reported in the eighth column. To verify if the differences are statically significant, we performed a *paired t-test* with a 95% confidence interval. The obtained p-value is reported in the ninth column.

With regard to the first research question, the LSPR retrieval model is less effective than a fully optimized BM25, although not always to a statistically significant degree. We report also the results of BM25 and LSPR when the default hyperparameter values for BM25 are adopted—results are those denoted with (*d*). In this case, LSPR is able to outperform BM25 in terms of MAP even if the results are not statistically significant.

With regard to the other research questions—the effect of selectivity—the results reported in Tables IV and V show that the most effective selectivity value depends on the test collection and therefore that the conjecture discussed in Costa and Melucci [2010] on a maximum value that relates selectivity to retrieval measure is not true. We plotted the effectiveness measures against selectivity. Plots are reported in Figures 12 through 15 and show that the relationship between selectivity and effectiveness is quite variable for low selectivity values when the *wt00-01* test collection is adopted; however, for all configurations, the effectiveness does not change a lot for high values of selectivity.

To reduce the number of hyperparameters of LSPR, thus making it comparable in terms of complexity to BM25, we investigated a fixed value for selectivity. In the current instantiation of LSPR, selectivity and a document-specific weight (in our case, a normalized version of BM25) determine the amplitude of the filter—that is, which bandwidth interval the filter is going to remove. The maximum allowed value for the amplitude is 200. Since the considered document-specific weight ranges from $[0, 1]$,[7] selectivity determines how many samples of the query spectrum the filter is going to remove. We set 100 as the default parameter to remove the most significant part of the spectrum related to the corresponding query term as well (see remarks reported in Section 4.2 on the amplitude of the spectrum before and after 100 samples). The results are reported in Tables VI and VII. $\text{LSPR}_{\text{BM25sOpt}}$ refers to the LSPR instantiation with the best selectivity value, whereas $\text{LSPR}_{\text{BM25s100}}$ refers to the instantiation where selectivity was set to 100. The results show that the two instantiations are comparable in terms of retrieval effectiveness. Looking at the *random training/test* hyperparameter tuning configurations, we can observe an increment in terms of effectiveness when adopting $\text{LSPR}_{\text{BM25s100}}$, particularly in terms of MAP and NDCG for *wt00-01*. This results suggest that using an efficient parameter optimization method is possible to further improve LSPR effectiveness.

[7]The saturated TF tends to 1 for TF that tends to $+\infty$ and the IDF was normalized over its maximum values, and therefore its maximum is 1.

Table VII. NDCG for LSPR Instantiation with Selectivity Set to 100

Test Collection	Hyperparameter Tuning	b	k_1	BM25	$\text{LSPR}_{\text{BM25sOpt}}$	$\text{LSPR}_{\text{BM25s100}}$
robust04	random training/test (d)	0.75	1.2	0.5146	0.5207	0.5183
	random training/test	0.22	1.1	0.5351	0.5234	0.5230
	upperbound (d)	0.75	1.2	0.5009	0.4988	0.4957
	upperbound	0.22	1.1	0.5201	0.5071	0.5051
wt00-01	random training/test (d)	0.75	1.2	0.4521	0.4353	0.4442
	random training/test	0.29	1.9	0.4813	0.4488	0.4620
	upperbound (d)	0.75	1.2	0.4306	0.4256	0.4194
	upperbound	0.40	1.1	0.4667	0.4377	0.4297

Table VIII. MAP for LSPR Instantiation with Saturated Frequency (satTF) for Filter Amplitude Estimation and Selectivity Set to 100

Test Collection	Hyperparameter Tuning	b	k_1	$\text{LSPR}_{\text{BM25sOpt}}$	$\text{LSPR}_{\text{BM25s100}}$	$\text{LSPR}_{\text{satTFs100}}$
robust04	random training/test (d)	0.75	1.2	0.2626	0.2625	0.2621
	random training/test	0.30	0.7	0.2668	0.2688	0.2667
	upperbound (d)	0.75	1.2	0.2405	0.2379	0.2372
	upperbound	0.30	0.7	0.2474	0.2463	0.2448
wt00-01	random training/test (d)	0.75	1.2	0.1872	0.1977	0.1916
	random training/test	0.35	1.2	0.2068	0.2073	0.2007
	upperbound (d)	0.75	1.2	0.1803	0.1774	0.1727
	upperbound	0.33	0.9	0.1863	0.1846	0.1752

Table IX. NDCG for LSPR Instantiation with Saturated Frequency (satTF) for Filter Amplitude Estimation and Selectivity Set to 100

Test Collection	Hyperparameter Tuning	b	k_1	$\text{LSPR}_{\text{BM25sOpt}}$	$\text{LSPR}_{\text{BM25s100}}$	$\text{LSPR}_{\text{satTFs100}}$
robust04	random training/test (d)	0.75	1.2	0.5207	0.5183	0.5183
	random training/test	0.22	1.1	0.5234	0.5230	0.5206
	upperbound (d)	0.75	1.2	0.4988	0.4957	0.4952
	upperbound	0.22	1.1	0.5071	0.5051	0.5037
wt00-01	random training/test (d)	0.75	1.2	0.4353	0.4442	0.4365
	random training/test	0.29	1.9	0.4488	0.4620	0.4549
	upperbound (d)	0.75	1.2	0.4256	0.4194	0.4128
	upperbound	0.40	1.1	0.4377	0.4297	0.4226

As mentioned previously, the instantiation of LSPR adopted in the experiments relies on the BM25 weight to estimate the amplitude of the filter. A further research question is related to the effect of this choice: is the LSPR effectiveness due to the adoption of BM25 for the filter amplitude estimation? To provide a first answer to this question, we investigated an LSPR instantiation where the document weight used for the filter amplitude estimation solely relies on the saturated TF—the amplitude of the peak associated to query terms is still estimated using IDF. Results are reported in Tables VIII and IX in terms of MAP and NDCG, respectively. The results show that the two instantiations with ($\text{LSPR}_{\text{BM25s100}}$) and without ($\text{LSPR}_{\text{satTFs100}}$) IDF are comparable in terms of retrieval effectiveness. The results obtained for the diverse instantiations suggest investigating further approaches to estimate the amplitude of the peak associated to query terms and the amplitude of the filter associated to documents. Techniques adopted in DSP could be useful to suggest these estimations.

6.5. Failure Analysis

To identify some hypotheses about under what conditions LSPR performs better than BM25, we carried out an analysis on the runs on the test collections where BM25 was fully optimized:

Table X. Features Adopted for the Failure Analysis

Feature	Description
ql	$N_L(q)$, which denotes the query length, (i.e., the number of locations in the query)
min_qTF	$\min_{t \in q} n_L(t, q)$, where $n_L(t, q)$ is the number of locations in the query where the term t occurs
max_qTF	$\max_{t \in q} n_L(t, q)$
avg_qTF	$\frac{1}{N_L(q)} \sum_{t \in q} n_L(t, q)$
min_IDF	$\min_{t \in q} IDF(t)$
max_IDF	$\max_{t \in q} IDF(t)$
avg_IDF	$\text{avg}_{t \in q} IDF(t)$
dl	$N_L(d)$, which denotes the document length
relevance	binary/graded assessment provided on the document with regard to the topic
query_doc_terms	number of query terms present both in the document and the query
w_{BM25}	BM25 weight of the document for the considered topic
w_{LSPR}	LSPR weight of the document for the considered topic
r_{BM25}	rank of the document in the ranked list obtained by BM25 for the considered topic
r_{LSPR}	rank of the document in the ranked list obtained by LSPR for the considered topic
min_TF	$\min_{t \in q} n_L(t, d)$, where $n_L(t, d)$ is the number of locations at which the term t occurs in the document d, namely the TF of t in d
max_TF	$\max_{t \in q} n_L(t, d)$
sum_TF	$\sum_{t \in q} n_L(t, d)$
avg_TF	$\text{avg}_{t \in q} n_L(t, d)$

—robust04 random training/test,
—robust04 upperbound,
—wt00-01 random training/test, and
—wt00-01 upperbound,

using the best-performing hyperparameter setting in terms of MAP reported in Table IV. We focused only on the MAP since NDCG explicitly takes into account diverse relevance degrees, thus introducing an additional variable in the failure analysis. We selected a set of features to describe each topic-document pair. The eighteen selected features and their description are reported in Table X; the description exploits the notation proposed in Roelleke [2013].

For each test collection, we selected a number of "good" topics and a number of "bad" topics. Good topics were those for which LSPR performed better than BM25; we selected the top 10 topics ranked by decreasing percentage change

$$\Delta_1 = \frac{\text{AP}_{\text{LSPR}} - \text{AP}_{\text{BM25}}}{\text{AP}_{\text{BM25}}}$$

and the top 10 topics ranked by decreasing absolute change

$$\Delta_2 = \text{AP}_{\text{LSPR}} - \text{AP}_{\text{BM25}}.$$

The final list of good topics for each test collection was obtained by merging the two obtained topic sets. For each topic, we selected the relevant document present both in the BM25 and LSPR result lists and with the highest difference in terms of rank position: $r_{\text{BM25}} - r_{\text{LSPR}}$.

Bad topics were those for which BM25 performed better than LSPR; we selected the top 10 topics ranked by increasing percentage change Δ_1 and the top 10 topics ranked by increasing absolute change Δ_2. The final list of bad topics for each test collection was obtained by merging the two obtained topic sets. For each topic, we selected the relevant document present both in the BM25 and LSPR result lists and with the lowest difference in terms of rank position: $r_{\text{BM25}} - r_{\text{LSPR}}$.

Each of the selected documents was represented as a feature vector $o \in \mathbb{R}^{18}$—that is, on the basis of the features reported in Table X. All entries of all four considered test collections were considered together to perform the analysis.

To consider the difference in terms of rank between BM25 and LSPR and to provide more weight to the differences at high-rank positions, we defined the following variable:

$$y = \frac{r_{\text{BM25}} - r_{\text{LSPR}}}{\max\{r_{\text{BM25}}, r_{\text{LSPR}}\}}. \tag{31}$$

On inspection, the data showed a relationship between y and the TF-type features—that is, min_TF, max_TF, sum_TF, and avg_TF. In particular, y tends to be negative (i.e., BM25 tends to be superior to LSPR) when min_TF increases; the same pattern can be observed with regard to sum_TF, max_TF, and avg_TF, although to a lesser degree. Note that there is a strong correlation between avg_TF, sum_TF, and max_TF as expected—two of these features will be excluded from the model—whereas there is no significant correlation between min_TF and the other TF-type features.

With regard to the IDF-type features (i.e., min_IDF, max_IDF, avg_IDF), no significant relationships with y were observed, although the correlation index would suggest the opposite. Indeed, a graphical inspection shows that the correlation is only caused by a few outliers (i.e., a few high IDFs associated with a few high y's). Moreover, no significant relationships were observed between dl, query_doc_terms, the qTF-like features, and y.

To better understand the relationship between the TF-type features and y, a linear model between y and those features was estimated. However, we had to apply a logarithmic transformations to the features and to y to obtain a good fit of the linear model expressed by the following equation:

$$y' = 1.32 + 0.78 \cdot \log(\text{avg\_TF}) - 0.68 \cdot \log(\text{min\_TF}) - 1.06 \cdot \log(\text{max\_TF}),$$

where $y' = \log(y - \min(y) + 0.0001)$. This model shows that y increases (i.e., LSPR becomes superior to BM25) when min_TF and max_TF decreases; the latter features were the most significant in the model (p-value < 0.0005), whereas avg_TF was little significant (p-value < 0.02).

To resume, it seems that LSPR outperforms BM25 when the query terms are not very frequent in the relevant documents. Considering how LSPR works, the results of the analysis can be explained this way. If the query terms are very frequent in a relevant document, this means that the set of filters associated with that document will have a large amplitude, and very likely almost all peaks of the spectrum of the query will be removed. In other words, the power of the spectrum after the filtering will be determined mostly by the small-amplitude components of the spectrum that are far from the peaks (e.g., see Figure 11(c)). Moreover, since almost all peaks are removed, the impact of the IDF weighting of the query is lower. In this situation, if many of the relevant documents are associated with this type of filtered spectra, it is not easy to rank them correctly. In other words, if we remove too much from the spectrum from many relevant documents, we might lose the ability to effectively predict their relevance.

7. CONCLUSIONS

This article proposes LSPR, a new IR model based on concepts taken from DSP. Experiments discussed in Section 6 show that the quality of the results is comparable to the state-of-the-art. More precisely, the failure analysis of Section 6.5 showed that LSPR performs better than BM25 when the query terms do not frequently occur in the relevant documents.

An interesting feature of LSPR is that the IR process has a physical interpretation. In fact, the query is viewed as a signal, and each document is represented by a set of filters; the retrieval process corresponds to filtering the signal with the filters, and thus each document provides a different filtered signal. The ranking is then made in such a way that the more the signal has been filtered by the filter set, the more the document associated with that filter set is considered relevant. This suggests a possible implementation for IR systems, especially taking into account the advent of the Internet of Things (IoT)—an introduction to the IoT is given by Atzori et al. [2010]. Indeed, traditional IR systems usually operate in digital format documents, yet it is likely that the IoT will require an alternative organization of the information stored in the interconnected physical objects (e.g., radio-frequency identification (RFID) tags, sensors, actuators, mobile phones) interacting with each other through uniform addressing schemes. As a possible scenario, consider a library where one wants to find books that can provide the information needed. Each book could be associated to an Radio-Frequency IDentification (RFID) tag or other hardware implementing a filter. If the user wants to know if a book is relevant for his search, his query could be transformed into a signal by means of a software application installed on his smartphone, and that signal would be sent to the filter associated to the book, which returns the filtered signal. The smartphone could then analyze the signal and return to the user an indication about the relevance of the book. Indeed, the LSPR as it has been presented in this work probably could not be directly employed in this scenario due to physical constraints, and it should be adapted. However, the important point is that our work tries to make a connection between the classical IR models (as explained in Section 5) and the physical world. In fact, the spread of IoT and distributed systems is likely to provide, in the near future, the conditions for the implementation of systems that interact more and more with users, and hence it is important to think about how traditional IR systems can be adapted to this new scenario.

The future work deals with three aspects. First, it will offer a more in-depth study of the possibility of implementing physically an IR system, as discussed previously. Second, will explore an analysis of the parameters of LSPR that could be tuned to improve the performances, such as the setting of the frequencies of the signal associated with the query, or the number of samples where the spectrum of the filter has the value 0 (i.e., the values of Z_{L_i} and Z_{R_i}). In fact, in this article, we proposed a simple approach, where the frequencies are determined depending on the position of the term in the query. A more sophisticated method can take advantages of clustering (that has been shown to be important for IR already by van Rijsbergen [1979] and Manning et al. [2008]), to assign close frequencies to terms belonging to the same cluster. In general, one might consider the mutual relationship between terms to dynamically set their frequencies. This way, the sections of the spectrum of the query associated with two related terms could overlap, and a filter affecting one term would also affect the other one. In terms of implementation this would not require major changes (the creation of the spectrum, of the filters, and the filtering operation would be the same after updating the frequencies), but it could lead to a possible increase in performance. In addition, a deeper investigation of the theory and techniques of DSP could potentially provide insights on how to increase the performances of LSPR, using approaches that might not be part of the standard IR literature.

Third, the optimization of the parameters of LSPR should be taken into account. As discussed in Section 6.2.2, LSPR employs the optimal parameters setting of BM25, and only the selectivity value has been optimized. Hence, the performance of LSPR is potentially suboptimal. Unfortunately, an application of grid search for the complete optimization of the parameters of LSPR would be too time consuming, as explained

in Section 6.2.2. To gain some insight on the improvement of the performance when the parameters of LSPR are better optimized, we implemented the line search method presented in Taylor et al. [2006]. We performed some experiments with the *wt00-01* collection in the upperbound setting, and we considered the MAP. We used the default values as a starting point. Using grid search for BM25, as shown in Table IV, yields a MAP of 0.1990, whereas LSPR (using the BM25 best setting and optimizing only the selectivity) provides a value of 0.1863. Using LSPR with the parameters setting found by line search provides a value of 0.2057 (with $b = 0.32$, $k_1 = 5.9$, and selectivity value 24). Just for reference, line search applied to BM25 yields a MAP of 0.1989. Note that the results of line search could be worse than that of grid search (as for BM25, even though slightly, in the test performed). In fact, the exploration of the domain in line search is driven, at each step, by the position of the best solution found in the neighborhood of the current best solution. On the contrary, grid search explores the whole domain, thus avoiding the issue of being trapped in a local optimum that could affect line search. Another drawback of line search is that the solution found depends on the starting point.

These results suggest that LSPR could provide better MAP values than BM25. However, given the aforementioned considerations, it is not straightforward to perform a full optimization of LSPR. A complete investigation of the question (taking into account other collections and the NDCG measure as well), focused on efficient optimization of the parameters, will be addressed in future work.

APPENDIXES

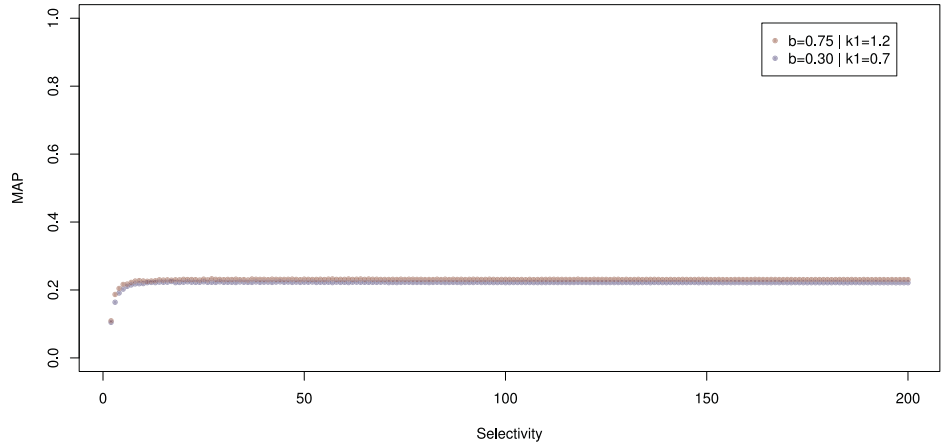
A. EFFECT OF THE SPECTRUM AND THE RESIDUAL POWER COMPUTATION METHOD

Table XI. MAP Values for Different Methods of Spectrum Computation

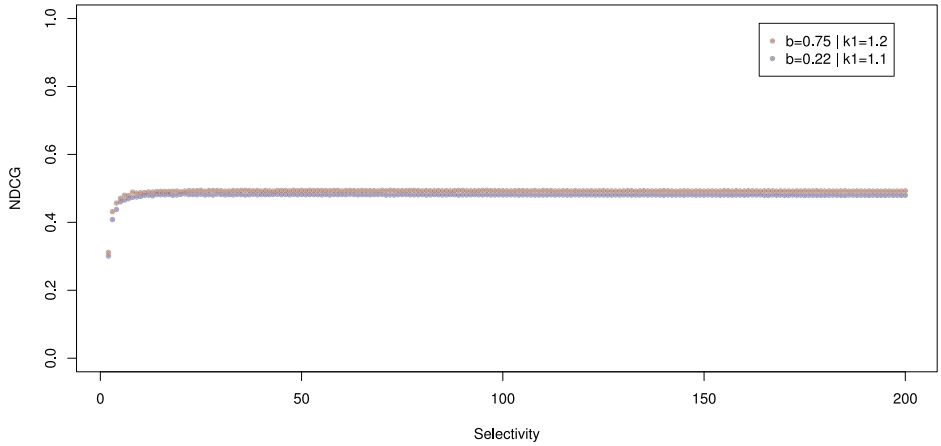
Test Collection	Hyperparameter Tuning	b	k_1	BM25	LSPR_{sum}	LSPR_{square_sum}
robust04	upperbound (d)	0.75	1.2	0.2353	0.2405	0.2337
	upperbound	0.30	0.7	0.2544	0.2474	0.2437
wt00-01	upperbound (d)	0.75	1.2	0.1716	0.1803	0.1533
	upperbound	0.33	0.9	0.1990	0.1863	0.1672

Note: LSPR_{sum} refers to the LSPR instantiation where the spectrum and the residual power computation is performed as the sum of the components. LSPR_{sum_square} refers to the LSPR instantiation where the spectrum and the residual power computation is performed as the sum of the squares of the components.

B. EFFECT OF THE SELECTIVITY VALUE IN TERMS OF RETRIEVAL EFFECTIVENESS

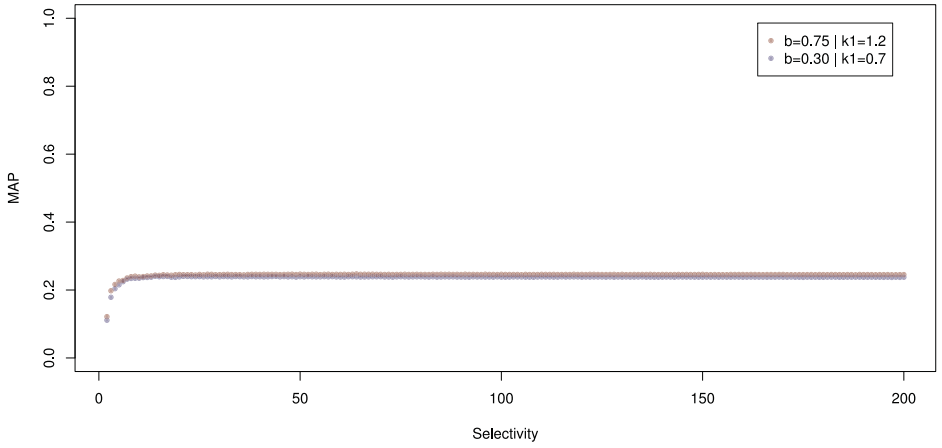


(a) MAP

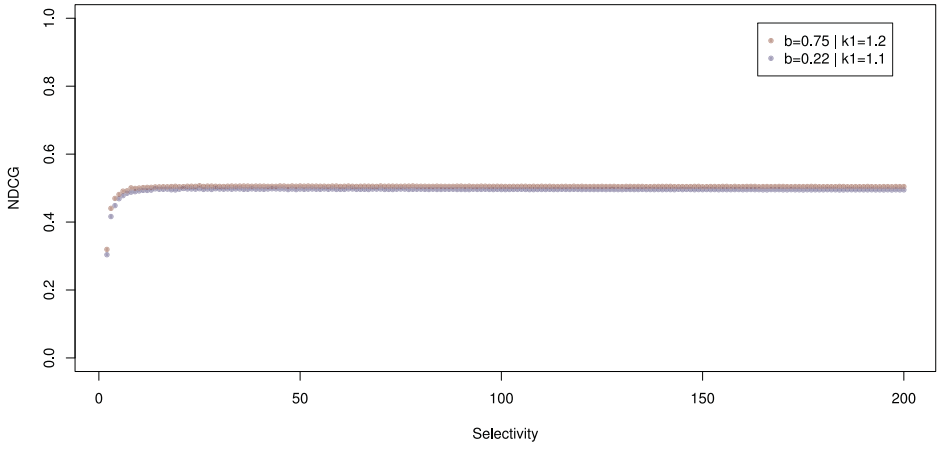


(b) NDCG

Fig. 12. Effect of the selectivity value in terms of MAP and NDCG on robust04—random split.

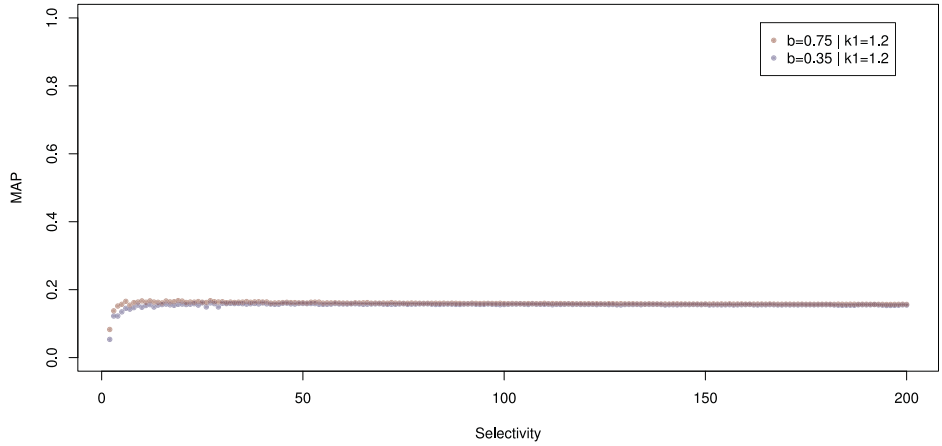


(a) MAP

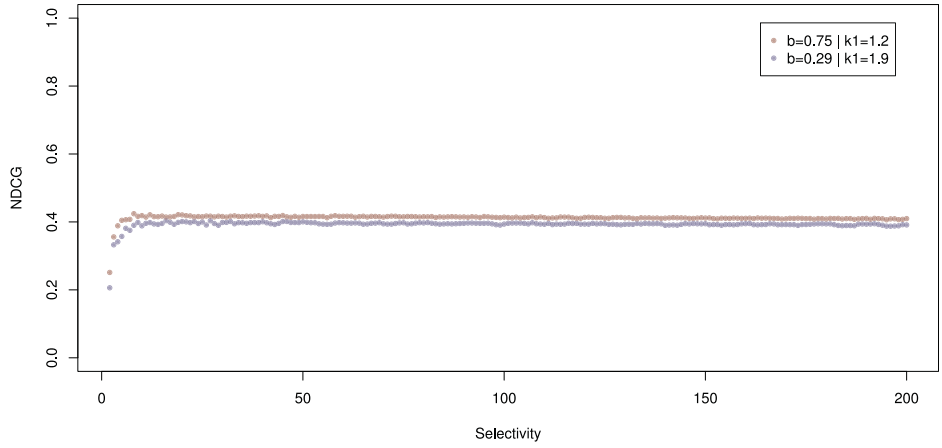


(b) NDCG

Fig. 13. Effect of the selectivity value in terms of MAP and NDCG on robust04—upperbound.

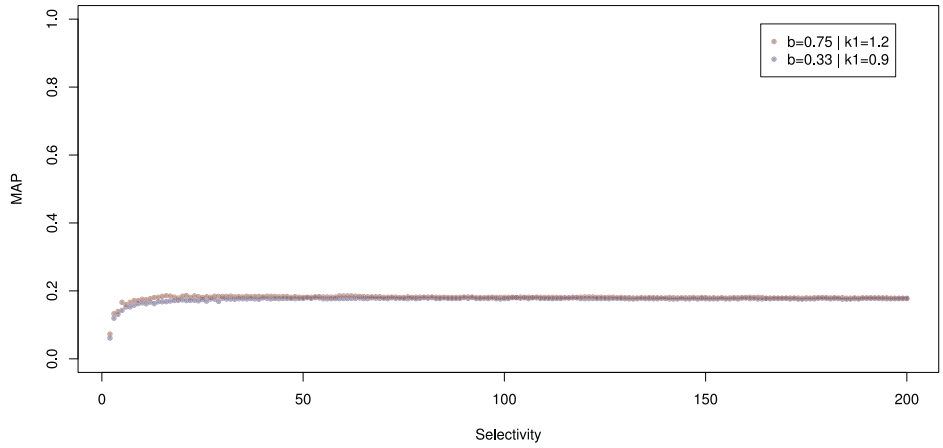


(a) MAP

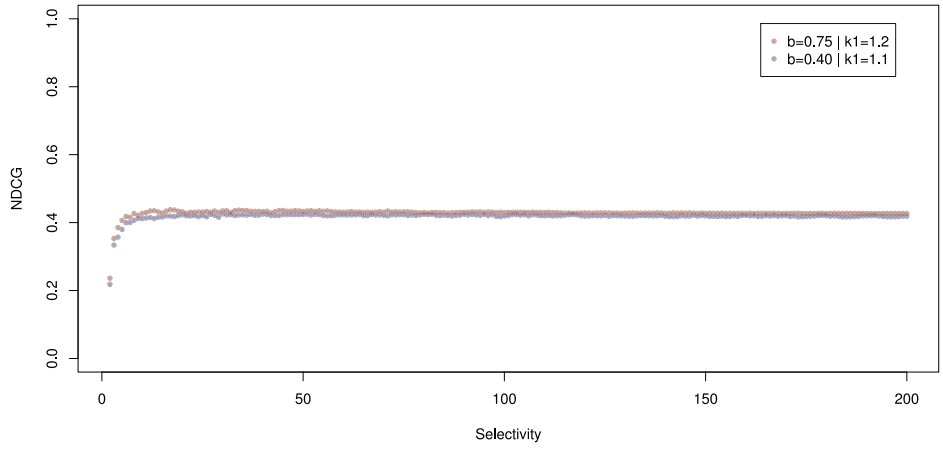


(b) NDCG

Fig. 14. Effect of the selectivity value in terms of MAP and NDCG on web0001—random split.



(a) MAP



(b) NDCG

Fig. 15. Effect of the selectivity value in terms of MAP and NDCG on web0001—upperbound.

ACKNOWLEDGMENTS

The authors would like to thank Marco Maso, Tony Quek, and Matthias Wildemeersch for their suggestions and comments on DSP. The authors also thank the associate editor and the anonymous referees for their comments and suggestions, which helped to improve the quality of the article significantly.

REFERENCES

G. Amati and C. J. van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems* 20, 4, 357–389.

L. Atzori, A. Iera, and G. Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15, 2787–2805.

K. Blekas and I. E. Lagaris. 2007. Newtonian clustering: An approach based on molecular dynamics and global optimization. *Pattern Recognition* 40, 6, 1734–1744.

J. P. Callan. 1994. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*. 302–310.

E. J. Candès, J. K. Romberg, and T. Tao. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics* 59, 8, 1207–1223.

S. Clinchant and E. Gaussier. 2010. Information-based models for ad hoc IR. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*. ACM, New York, NY, 234–241.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2000. *Introduction to Algorithms* (2nd ed.). MIT Press, Cambridge, MA.

A. Costa and M. Melucci. 2010. An information retrieval model based on discrete Fourier transform. In *Advances in Multidisciplinary Retrieval*. Lecture Notes in Computer Science, Vol. 6107. Springer, 84–99.

W. B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice.* Addison-Wesley.

D. L. Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52, 4, 1289–1306.

H. Fang, T. Tao, and C. Zhai. 2011. Diagnostic evaluation of information retrieval models. *ACM Transactions on Information Systems* 29, 2, Article No. 7.

F. J. Harris. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE* 66, 1, 51–83.

M. Hearst. 1997. Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics* 23, 1, 33–64.

K. Järvelin and J. Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4, 422–446.

M. Kaszkiel and J. Zobel. 1997. Passage retrieval revisited. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*. 178–185.

M. Kaszkiel, J. Zobel, and R. Sacks-Davis. 1999. Efficient passage ranking for document databases. *ACM Transactions on Information Systems* 17, 4, 406–439.

D. Knaus, E. Mittendorf, and P. Schäuble. 1995. Improving a basic retrieval method by links and passage level evidence. In *Proceedings of the 3rd Text Retrieval Conference (TREC-3)*. 241–246.

T.-Y. Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3, 225–331.

X. Liu and W. B. Croft. 2002. Passage retrieval based on language models. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM'02)*. ACM, New York, NY, 375–382.

L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. 2012. Recommender systems. *Physics Reports* 519, 1, 1–49.

C. Manning, P. Raghavan, and H. Schütze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press, Cambridge, MA.

M. Melucci. 1998. Passage retrieval: A probabilistic technique. *Information Processing and Management* 34, 1, 43–67.

M. Melucci. 2009. Vector-space model. *Encyclopedia on Database Systems*, L. Liu and M. Tamer Ozsu (Eds.). Springer, 3259–3263.

M. Melucci and C. J. van Rijsbergen. 2011. *Quantum Mechanics and Information Retrieval*. Springer, Berlin, 125–155.

D. Metzler and W. B. Croft. 2007. Latent concept expansion using Markov random fields. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. ACM Press, New York, NY, 311.

S. K. Mitra. 2006. *Digital Signal Processing: A Computer-Based Approach* (3rd ed.). McGraw-Hill, New York, NY.

E. Mittendorf and P. Schäuble. 1994. Document and passage retrieval based on hidden Markov model. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*. 318–327.

A. V. Oppenheim, R. W. Schafer, and J. R. Buck. 1999. *Discrete-Time Signal Processing* (2nd ed.). Prentice Hall, Upper Saddle River, NJ.

A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. 1996. *Signals & Systems* (2nd ed.). Prentice Hall, Upper Saddle River, NJ.

L. A. F. Park, K. Ramamohanarao, and M. Palaniswami. 2005. A novel document retrieval method using the discrete wavelet transform. *ACM Transactions on Information Systems* 23, 3, 267–298.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing* (2nd ed.). Cambridge University Press, Cambridge, MA.

S. E. Robertson. 1977. The probability ranking principle in IR. *Journal of Documentation* 33, 4, 294–304.

S. E. Robertson and H. Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4, 333–389.

T. Roelleke. 2013. *Information Retrieval Models: Foundations and Relationships*. Morgan and Claypool.

G. Salton, J. Allan, and C. Buckley. 1993. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*. 49–58.

G. Salton, A. Singhal, C. Buckley, and M. Mitra. 1996. Automatic text decomposition using text segments and text themes. In *Proceedings of the ACM Conference on Hypertexts (Autonomous Hypertext Systems and Link Discovery)*. 53–65.

S. Shi, J.-R. Wen, Q. Yu, R. Song, and W.-Y. Ma. 2005. Gravitation-based model for information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*. ACM, New York, NY, 488–495.

K. Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28, 1, 11–21.

M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. 2006. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM'06)*. ACM, New York, NY, 585–593.

C. J. van Rijsbergen. 1979. *Information Retrieval* (2nd ed.). Butterworths, London.

C. J. van Rijsbergen. 2004. *The Geometry of Information Retrieval*. Cambridge University Press, Cambridge, MA.

M. Weinstein and D. Horn. 2009. Dynamic quantum clustering: A method for visual exploration of structures in data. *Physical Review E* 80, 6, 066117.

C. H. Yeung, G. Cimini, and C.-H. Jin. 2011. Dynamics of movie competition and popularity spreading in recommender systems. *Physical Review E* 83, 1, 016105.

Y.-C. Zhang, M. Blattner, and Y.-K. Yu. 2007. Heat conduction process on community networks as a recommendation model. *Physical Review Letters* 99, 15, 154301.

Received November 2013; revised April 2015; accepted July 2015