Olivier Canévet
Élève ingénieur en 3$^{é}$ année à TELECOM Bretagne
olivier.canevet@telecom-bretagne.eu

# Evaluating ranking methods on heterogeneous digital library collections

Master Recherche en Sciences, Technologies, Santé
de Télécom Bretagne en cohabilitation conjointe avec l'Université de Rennes1
Mention "Électronique et Télécommunications"
Spécialité Signal, Image, Systèmes Embarqués, Automatique

April – September 2012

Advisors

Jean-Yves Le Meur
jean-yves.le.meur@cern.ch

Ludmila Marian
ludmila.marian@cern.ch

Thierry Chonavel
thierry.chonavel@telecom-bretagne.eu

# Abstract

In the frame of research in particle physics, CERN has been developing its own web-based software *Invenio* to run the digital library of all the documents related to CERN and fundamental physics. The documents (articles, photos, news, thesis, ...) can be retrieved through a search engine.

The results matching the query of the user can be displayed in several ways: sorted by latest first, author, title and also ranked by word similarity.

The purpose of this project is to study and implement a new ranking method in Invenio: distributed-ranking (D-Rank). This method aims at aggregating several ranking scores coming from different ranking methods into a new score. In addition to query-related scores such as word similarity, the goal of the work is to take into account non-query-related scores such as citations, journal impact factor and in particular scores related to the document access frequency in the database. The idea is that for two equally query-relevant documents, if one has been more downloaded for instance, it should be displayed in front of the other.

The approach that we studied consists in using *logistic regression* as the aggregation process, which is performed through a weighted sum of the scores to be aggregated. Usually, optimal weights can be computed based on the data. In our case, we used the user feedback: the search activity has been recorded for six months (queries made, displayed, downloaded documents,...) and we divided this data set in two: one to estimate the optimal coefficients and the other to test them. The test consisted in reranking the queries made by the users with the optimal coefficients. Then we compared the results with the initial ranking to see if the documents which were clicked at the time were ranked higher.

The optimal coefficients obtained are coherent in the sense that negative attributes for a document got a negative coefficient in the logistic formula. But the order of magnitude between the logistic coefficients were unexpected, as query-relevant score was much lower than the others weights. The re-ranking of the queries showed some improvement for records which had already been downloaded in the database and which were ranked higher.

# Résumé

Dans le contexte de recherche en physique des particles, le CERN développe sa propore application web *Invenio* pour stocker et gérer une bibliothèque numérique de données hétérogènes, concernant principalement la physique des particles et l'actualité du CERN. Les documents sont accessibles via un moteur de recherche.

Les documents correspond à la requête faite par un utilisateur peuvent être affichés de différentes manières: ils peuvent être triés selon leur date (le plus récent en premier), selon l'auteur ou encore le titre, ou bien ils peuvent être hierarchisés suivant leur pertinence par rapport à la requête.

La finalité de ce projet est d'étudier et d'implémenter dans *Invenio* une nouvelle méthode de hiéarchisation des résultats : *distributed-ranking* (D-Rank). Cette méthode consiste à aggréger différents scores provenant de différents classements pour obtenir un nouveau classement. En plus des attributs relatifs au contenu du document lui-même (comme la pertinence par rapport à la requête, le nombre de citations ou le *journal impact factor*), on cherche à prendre en compte également des données statistiques sur l'accès à un document. En effet, même si un document sera moins pertinent qu'un autre par rapport à la requête, le fait d'avoir été plus consulté par les utilisateurs doit lui conférer un certain avantage et ce dernier pourrait donc être mieux classé.

*D-Rank* utilise la régression logistique pour l'aggrégation des scores. Cet outil statistique suppose l'utilisation de poids sur chaque attribut que l'on souhaite aggréger, poids qui peuvent être calculés grâce aux données pour être optimaux. Dans notre cas, nous utilisons les requêtes faites par les utilisateurs pour calculer ces coefficients, la finalité étant de classer en premiers les documents qui ont le plus de chances d'être consultés. Pour cela, nous avons utilisé cinq mois de données de navigation dans la librairie pour calculer ces poids optimaux, puis nous avons utilisé les requêtes du sixième mois, que nous avons classé avec cette nouvelle méthode, pour voir si les documents qui avaient été consultés à l'époque ont été mieux classés.

Les coefficients ont un signe coherent: les attributs péjoratifs pour un document ont un poids négatif dans la régression logistique. Cependant, l'ordre de grandeur entre les coefficients est inattendu: le coefficient de pertinence par rapport à la requête est bien inférieur aux coefficients de la qualité

du document. Le reclassement des résultats du dernier mois a montré une amélioration pour les documents qui avaient déjà été téléchargés. Une explication possible à cet écart dans les ordres de grandeur des coefficients est que les requêtes utilisées pour l'apprentissage étaient triées selon les plus récents document en premier, et non par pertinence.

Pour améliorer les résultats, il serait intéressant de ne retenir que certaines requêtes pour l'apprentissage, celles qui correspondraient à une recherche précise de l'utilisateur et qui pourraient être testées sur des requêtes similaires dans le dernier mois.

# Acknowledgments

First, I would like to thank Ludmila Marian who supervised my work all along those six months, by helping and supporting me patiently, and for all our musings which lead to interesting ideas.

Second, I would like to thank Jean-Yves Le Meur for proposing me to work on this extremely challenging project, which had plenty of open questions from the beginning, which made it perfect for a Master Thesis project.

I would also like to thank Tibor Šimko for his help on technical questions and also for sharing his experience on Linux with me. I thank Jérôme Caffaro for the interest his showed all along in my project and his help on the database. More generally, the Invenio development team is a great team I was very glad to be in and thank everyone.

Last but not least, it is important for me to thank CERN for letting students work within it, and for having great social activities: dance, biketowork, ...

# Contents

# Introduction

Document retrieval is a challenging task especially for digital repositories containing millions of records. One may identify two tasks for this purpose: the first one is to find text information of interest (matching a user query) out of a large collection of documents. The second is to display the results – in a reasonable amount of time – in a satisfactory order so that the user finds easily what (s)he is looking for. The latter task can be achieved by estimating the relevance of a document towards the query or towards its intrinsic content or history, which is then used to rank the results.

In the frame of fundamental research in Physics, CERN publishes lots of various documents which has lead the institution to develop its own digital library software (*Invenio*) which is currently used by CERN (as CDS *CERN document server*) and by other institutions as well. Invenio allows to run an online digital library and handles different types of documents from articles to books and from photos to videos, ...

Currently, in addition to sorting methods based on date, author, title, and other metadata fields several ranking methods are implemented within Invenio such as word similarity (towards the query) or citations (how many times a document is cited by others) which all consist in a specific score for the records. However, it can be interesting to take into account several rankings at the same time and to aggregate several individual scores into one, providing thus a new ranking score. In particular, it would be interesting to take into account the access frequency of a document in the ranking. Indeed, if a record is famous (e.g. downloaded a lot), it means that it is important, and that its fame could contribute positively to its ranking score.

Distributed-ranking has been developed in the frame of CDS to combine quality scores (such as the number of displays, downloads) with the query-relevance of a record. This method involves logistic regression based on a weighted sum, in which the weights directly affect the final aggregated score. The principal concern of this dissertation is in finding the best values for the weights to provide an efficient ranking of the results. For this purpose, the search activities was recorded: for a given query, which records were displayed and which ones were clicked (i.e. which one were of particular interest for the user?)

The rest of the dissertation is organised as follows:

**Context** We briefly put the project into its context of research at CERN, and in the development of Invenio.

**State of the art** We present what ranking is and how different scores can be aggregated. In particular, we present more in details *distributed-ranking* which is the key point of the project.

**Attributes to aggregate** We present the attributes related to the access frequency of a document that are going to be aggregated in the process of ranking.

**Implementation of D-Rank** We describe how the method is implemented within Invenio, and how the access counts to the documents are computed.

**Learning the optimal coefficients from the user feedback** We learn the optimal coefficients on a training set and we test them on a separate set by comparing the rankings of the documents of interest between the original ranking and the new ranking.

# Chapter 1

# Context of the internship

This chapter presents the context of the work done during this internship. Although CERN carries out research in the field of fundamental physics, it is also very active in information technology, and provides tools and facilities for its own needs as well as for other institutions.

## 1.1 CERN

The *European Organization for Nuclear Research* (CERN) [1, 2] is an international research center specialized in fundamental physics. It is located in Geneva on the Franco-Swiss border but has several other sites in the area for the facilities of some experiments. CERN was founded in 1954 by 12 European governments and now counts 20 European member states.

CERN provides the world's largest and most complex instruments and facilities required for high-energy physics (HEP) and more particularly to study the basic constituents of matter. The main instruments used for these research activities are accelerators such as the Large Hadron Collider, Synchrotron boosters, an Antiproton Decelerator (AD) and particle detectors like the Compact Muon Solenoid.

## 1.2 Experiments at CERN

A wide range of experiments are conducted at CERN to understand what the matter is made of as well as anti-matter. The main idea is to accelerate particles to achieve a high energy level in which particles will behave differently and analyse it through detectors. The following briefly describe some experiments.

At a low level of energy, *ISOLDE* [3] is a source of radioactive isotopes and is located at the Proton-Synchrotron Booster. It can produce more than 1000 kinds of isotopes with a view to studying the nucleus, but some are also related to astrophysics or life sciences.

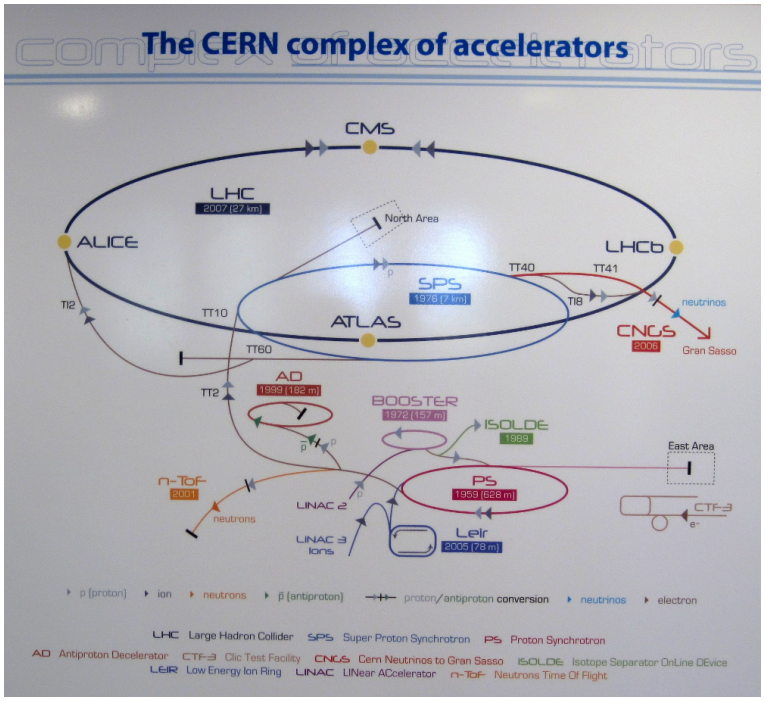



Figure 1.1: CERN accelerators

To produce antimatter, the *proton synchrotron* delivers high energy protons which collides with a block of metal thus producing a proton/antiproton pair. As the antiprotons travel too fast, they cannot be used to create antiatoms yet. The AD's task is to make a directed low-energy beam out of the those antiprotons thanks to focussing magnets. When the antiprotons have reached 10% of the speed of light, they are mixed with positrons to form antiatoms. One experiment involved with AD uses antiprotons for access the suitability in cancer therapy.

The flagship of CERN has been the *Large Hadron Collider* for the past few years. LHC is a huge instrument build 100 m underground North Geneva in the *Pays de Gex*, between the Jura mountains, the Geneva airport and CERN. It consists in a 27 km circle pipe in which travel two hadrons (beams of subatomic particles, protons or lead ions) in opposite direction. The two hadrons collides in different areas of the pipe depending on the experiment. This allows physicists to recreate the conditions right after the Big Bang and reach very high energies. The particles which are created during the collisions are analysed by different type of detectors. The two major experiments of the LHC are ATLAS (*A Toroidal LHC Apparatus* [4]) and CMS (*Compact Muon Solenoid* [5]). Those two experiments have common purposes such as finding the Higgs boson, particles involved in the dark matter or searching for extra dimensions.


Figure 1.2: CCC – The CERN Control Center where the eight accelerators are controlled

The accelerators at CERN were also used to test the module AMS-02 (*Alpha Magnetic Spectrometer Experiment*) embedded in the International Space Station (*ISS*). This module is designed to measure cosmic rays with a view to finding traces of unusual matter. This experiment tries to:

- improve the sensitivity of the antihelium/helium flux ratio in the Universe,
- detect collisions of neutralinos (if they really exist), which may constitute dark matter.

The accelerators at CERN allowed to expose the AMS-02 module to nuclear particle beams before sending it to the ISS.

## 1.3 Achievements

Several major achievements have been made at CERN in those experiments.

In 1983, the two experiments UA1 and UA2 [6] lead to the discovery of the two heavy particles W and Z bosons which were only hypothetic so far. The SPS accelerator was modified into a proton-antiproton collider to reach sufficient level of energy in comparison to the collision of one single beam on a fixed target. The two physicists behind this experiment received the Nobel Price of Physic the next year.


Figure 1.3: AMS building on Prévessin site

In 1995, the PS210 experiment lead to the creation of the first antihydrogen atoms, traveling at the speed of light. And in 2011, the ALPHA Collaboration succeeded in trapping antimatter for 1000 seconds [7]. This amount of time was sufficient enough to change the state of the antiatoms and thus analyse its spectrum.

Most recently [8], the two experiments CMS and ATLAS presented their preliminary results on the search of the Higgs boson: *We observe in our data clear signs of a new particle, at the level of 5 sigma, in the mass region around 126 GeV.*

Besides physic discoveries or achievements, the *World Wide Web* was first proposed at CERN in 1989 as a possible solution to manage general information and tackles the problem of loss of information in an big organization where the turnover of people is high. The original idea was to *link and access the information as a web of nodes in which the user can browse at will.*

Following this idea of information storage and retrieval, CERN has been developing its own digital library for HEP publications and other documents related to life at CERN.

## 1.4 CDS and Invenio

The CERN Document Server (CDS) [9] is a digital library containing over one million records mainly in HEP. Not only does CDS host articles, books and proceedings but also multimedia such as pictures and videos, presentations and talks, which makes it one of the largest heterogeneous

digital library. Due to the complexity of the repository and CERN specific needs such as articles written by hundreds of authors or a volume of data of millions of records, CERN decided to develop its own software called *Invenio*. Moreover, a study lead in the field of particles physics showed that when it comes to finding literature on a subject, users would rather turn to specialized search engines than general purpose ones [10].

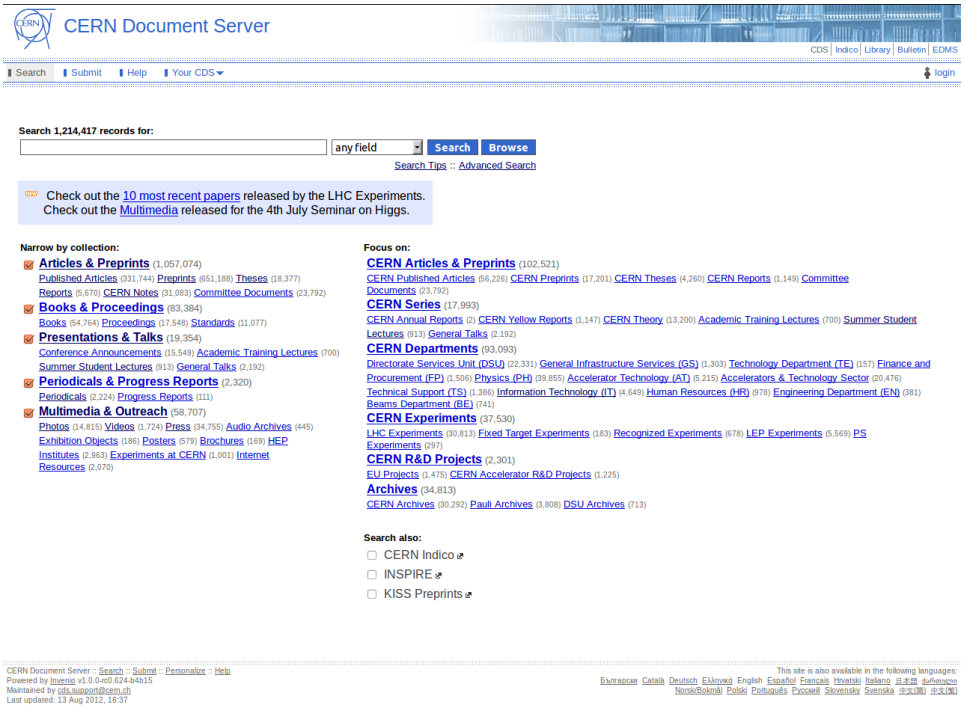



Figure 1.4: The CERN Document Server main page

Invenio [11] is an open source digital library software which provides tools to run a digital library. Invenio is developed by CERN and other contributors among its community like EPFL, DESY or SLAC. It is used by many institutions or projects like INSPIRE (for HEP documents) or others universities (for academic documents).

Invenio allows to manage very large heterogeneous collections of documents by following steps:

**ingestion** adding a document or bulk of documents to the file archive collection,

**classification** assign the document to one or more categories,

**indexing** associate or tag the document with different search terms,

**curation** keep metadata updated regularly,

**dissemination** make information available for other systems or for the users.

Invenio has a powerful Google-like search engine (see picture 1.4) and can combine metadata, reference and fulltext search. Invenio is compliant with standards such as the Open Archives Initiative metadata harvesting protocol (OAI-PMH) for efficient dissemination of content, it uses MARC 21 as its bibliographic format to store and process bibliographic meta-data. Invenio also provides personalization and collaborative features such as alerts, baskets, reviews or comments. It is available in 28 different languages.

Invenio is developed and maintained by the two sections *DLS* and *DLT* (Digital Library Service and Technology) which are part of the *CIS* (Collaboration and Information Services) group of IT department.

## 1.5 Ranking the query results

As stated before, Invenio has a powerful search engine to retrieve the documents. The results can currently been *sorted* by common criteria based on the fields of the document (author, report number, title, latest first, year), by ascending or descending order. Those criteria do not require any off-line precomputation and just correspond to a new arrangement of the records in some ordered sequence.

Invenio also offers the possibility to *rank* the records based on precomputed scores or information such as word similarity or citation: the former requires saving the keywords and their importance inside the document (log entropy of a word in the document) which will be used at search time for ranking while the latter involves building the graph of citations/references of the documents in the database.

Sometimes, the first displayed results are not clicked by the user even if they are the most relevant to the query. As queries are performed, some records become more famous than the others: they are more downloaded and viewed by the users. It is reasonable to think that therefore, those records should be favored upon less famous records in future queries. It would then be interesting to take into account the access frequency of the records in the ranking of the documents in addition to already existing ranking methods (word similarity, citation,...), to rank higher those famous documents and finally get results fitting the clicks to the user. Ideally, the user should find what (s)he is looking for in the first page.

## 1.6 Objectives

A promising aggregation method was proposed to take several ranking scores into account to get a new ranking value: distributed-ranking (D-Rank). Over the past two years, several students worked on analysing the clicks of the users, work which was not integrated to the production version of Invenio.

The objectives of the projects are the following:

- take over the previous work dealing with analysing the user queries,
- adapt the aggregation method to the need of the project,
- allow the user to customize the influence of the scores with one another,
- use the user feedback (what the users searched for and then clicked) to extract the relevant and irrelevant records,
- compute optimal regression coefficients on those data,
- test those coefficients on another set of queries (for which we know which records were clicked) to see if the new ranking method with the optimal coefficients ranks higher the documents of interest, ranking method ranks higher the documents
- write a documentation/user guide on the general workflow of the method, from the query analysis to the computation of the coefficients for ranking.

# Chapter 2

# State of the art

This chapter presents the work related to our research. We present current methods used to rank search results, to aggregate scores and to find the best aggregation functions.

## 2.1 Ranking scientific documents

Different methods can be used to rank records depending on what the user is interested in.

### 2.1.1 Word similarity method

Invenio provides *word similarity* ranking method [12, 13]. Once relevant documents to the query are found, the next step is to rank them based on the words of the query and on how those words appear in the document (title, author, text, ...) To do so, documents and queries are represented as vectors in the term space.

When a document is inserted in the collection, a term list is built for each document (*indexing* phase). This list contains relevant words inside the document (word appearing at least once, and numbers, stopwords and punctuation are removed). Then, a weight is computed for each word of the list based on:

- how many times the word appears in the documents ($L$, local),
- how many times the word appears in the collections ($G$, global),
- a normalized value related to the size of the documents ($N$, normalization),

given by following formula:

$$w = LGN$$

One should therefore note that as the repository gets more documents, the weights of the words change even if the document remains unchanged.

To compute the similarity of a document against a query, the dot product between the two vectors is computed. The following example is taken from [12]: query "infant fetus" on the document of table 2.1 gives following similarity score:

$$\underbrace{0.18 \times 0.6}_{infant} + \underbrace{0.09 \times 0.3}_{fetus} = 0.135$$

A geometric interpretation of this dot product is the cosine of the angle between the two vectors. In other words, ranking documents according to word similarity is equivalent to ranking the angles between the document vector and the query vector, knowing that a high similarity between the query and the document corresponds to a low angle.

Table 2.1: Weights of a document and a query

| Document | | Query | |
|---|---|---|---|
| Word | Weight | Word | Weight |
| blood | 0.36 | acid | 0.3 |
| fetus | 0.09 | fetus | 0.3 |
| placenta | 0.09 | placenta | 0.3 |
| infant | 0.18 | infant | 0.6 |
| ... | ... | | |

The statistics and those weights are precomputed by Invenio after the documents are indexed in the database. The weights are recomputed on a regular basis by the task `bibrank -R` in the `bibsched`.

### 2.1.2 Citation ranking

Citations pages [14] are a good way to evaluate how relevant to the community a scientific publication can be (just as the *journal impact factor*). Indeed, the more cited a document will be, and the more relevant it should be. However, the main drawback of *citation count* is that gives all the citations the same importance.

To overcome this problem, the *PageRank* algorithm can be used on the bibliographic graph of the publications. This algorithm was introduced by Larry Page and used by the Google search engine, to rank results of web searches. The representation used is a graph, in which the web pages are nodes and the edges represent how they are linked with one another. The graph is therefore directed. A weight is computed for each page: the more pages will point to a page and the more important this latter will be. The solution to this problem correspond to a stationary distribution of a Markov chain which gives the final weight for each pages.

As scientific documents cite one another, the same representation can be used to compute the impact of a document in the literature: is it cited a lot, cited by major documents?...

As an example of use case, Google Scholar [15] uses citations to rank the results of the queries. Although implemented in Invenio, this method is not used yet on CDS because of limited amount of citation date.

## 2.2 Aggregating rankings or scores

Rank aggregation consists in combining several individual rankers to obtain a new value, which is supposed to better rank the matching records than each method individually. Rank aggregation are classified in two categories:

**score-based aggregation** during the ranking process of each individual ranker, a score is assigned to each record which will then be used in the aggregation function to compute a new one. The records will then be ranked based on this new value.

**order-based aggregation** after the ranking process, the aggregation will be based on the order of the documents (and not on the score given by each individual method)

For example, search engines can allow to combine multiple criteria to [16]:

- merge the outputs of different engines,
- let the user customize his/her search by choosing which criteria should be emphasized,
- analyse the preferences of different users for collaborative retrieval.

One reason why merging ranking is necessary is that it becomes difficult to store and manage continuous growing collections on a single device. The collections are therefore split on several machines and at search time, queries are sent to several search engines which will return several ranking tables. The results are displayed to the final user in one single list by aggregating the rankings. The authors of [17] proposed not to use the ranking but the logarithm of it for aggregation. Indeed, a difference of 10 ranks in the first positions (between rank 20 and 30 for instance) seems to be more meaningful on the relevance of the documents than a difference of 10 farther in the ranking (between rank 500 and 510). This is showed in figure 2.1.

As for *score-based* aggregation, several basic functions can be used such as the mean (arithmetic, geometric or any improvement of it). More complex methods also exist, in particular *distributed-ranking* (D-Rank) which will be presented more in detail later. In any case, the scores provided by the different ranking methods have to be normalized so that the scores can be compared and thus aggregated.

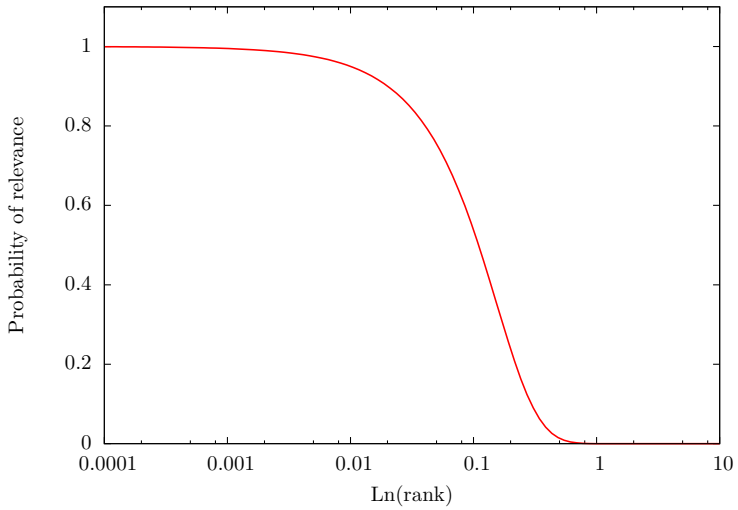



Figure 2.1: Converting logarithm of ranking to probability of relevance for aggregation

## 2.3 Distributed-ranking (D-Rank)

Distributed-ranking [18] is a ranking method developed by EPFL and CERN in the frame of ranking scientific publications for Invenio. The main goal is to aggregate several individual scores to obtain a better rank. For instance in our case, we are going to aggregate the relevance of a document to the search terms (word similarity) with its history: how many times it was displayed, downloaded, seen, cited,...

### 2.3.1 Overview

We first briefly describe the main steps of the D-Rank method.

1. The first step is to choose the attributes to aggregate. Just as with other statistical approaches such as SVM, it is important for the attributes not to be correlated.

2. The second step is the normalization: this phase consists in normalizing the initial scores between 0 and 1 so that the new scores reflect the underlying distribution of the scores. As an example, if an initial score $n$ corresponds to the median score of all the observed scores for the given attribute, its corresponding normalized score is 0.5.

3. The third step is the aggregation. It is performed using *logistic regression*. The combination of the (normalized) scores is done with a weighted sum, and returns a single value (between 0 and 1) which can

be interpreted as the probability for the document to be relevant (to the query and all the statistical attributes of the document).

### 2.3.2 Score normalization

Aggregation should be performed on comparable scores in order to avoid the predominance of some feature upon others.

Straightforward methods such as rescaling the scores in the range $[0, 1]$, shifting mean to 0 and variance to 1 could be used but they do not take into account the distribution of the scores which make them sensitive to noise and outliers.

Existing works such as [16, 19] address this issue. One way to make the normalized scores reflect the distribution of the scores is to use percentiles. A score close to the median of the distribution should have a normalized score of 0.5. This normalisation is performed in three steps:

1. compute values at a percentile level,
2. smooth values using standard estimation techniques,
3. construct the cumulative distribution.

Figure 2.2 shows an example of score normalisation for a randomly generated distribution of scores: we used three gaussian distributions of mean 6, 10 and 20 respectively. The median of the score is around 10 and receives a normalized score of 0.5. We also note that in the range $[14, 18]$, there are few scores. As a result, they all receive a similar normalized score, meaning that in this range, there is not that much distinction to make between the records.

### 2.3.3 Logistic regression

This section describes the theoretical background for using logistic regression and gives some simple examples of its application. It also presents the method commonly used to estimate the regression coefficients.

#### Overview

In a large number of problems, the response variable is categorical, such as binary (success/failure, diseased/healthy) or ordinal (normal, mild or severe) and depends on a set of explanatory variables. Basically, we have observations of a binary variable and we have the values or the states of several explanatory variables, for which we think they can have an impact on the value of the response variable.

As an example, in 1991, in Alberta State, Canada, a study was conducted before the legislation on the seat belt wearing, based on 86769 car-crash

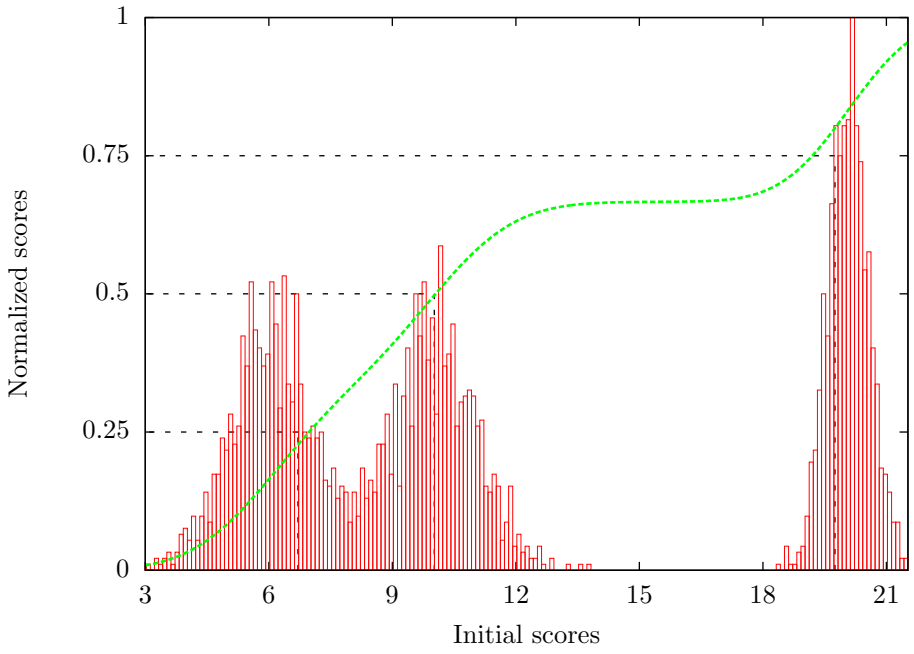



Figure 2.2: Percentile normalisation

reports, to explain the seriousness of the injuries after an accident. The following variables were used:

- the driver was on the influence of alcohol (yes/no),
- sex of the driver,
- the driver was wearing his/her seat belt.

The seriousness of the injuries ranged from 0 (nothing) to 3 (lethal).

More generally, logistic regression is used in epidemiological studies to evaluate risks such as sanitary and is also convenient because one can interpret easily how an explanatory variable will impact the outcome value in comparison to the others.

This statistical tool is quite suitable for our purpose. Indeed, we want to distinguish between relevant records and irrelevant records, from a set of explanatory variables (relevance to the query, number of downloads, views,...)

### Theoretical background: the logit model

The most natural way to quantify the chances that an event will occur is the probabilities, ranging from 0 to 1. But other ways exist such as *odds*, widely used by gamblers. The odd of an event is the ratio of the probability of the event to the probability of its complementary. As a result, odds range

from 0 to $\infty$. This transformation removes the upper bound, and taking the logarithm removes the lower one.

The logit model consists in equating the log-odd of the output variable (of probability $\pi$) as a linear combination of the input variables $x_i$:

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_i \beta_i x_i$$

where $\beta_i$ correspond to the influence of variable $x_i$. The probability of the output variable can be computed by the inverse equation:

$$\pi = \frac{1}{1+\exp(-\beta_0 - \sum_i \beta_i x_i)}$$

In other words, the logit model aggregates the different scores of the explanatory variables to produce the probability for the observation to be a success or a failure. The choice of the coefficients is therefore crucial to be able to explain the observation. Those coefficients can be estimated to be optimal to the observation.

### Maximum likelihood estimation

The optimal coefficients to fit the data can be estimated with maximum likelihood [20]. From a formal view, the binary response variable $Y$ can be seen as a random variable which depends on the explanatory variables $X_i$. The probability density of Y can be written as:

$$p(Y|X_i)$$

and we can also write:

$$\mathbb{E}[Y|X_i] = \ln\left(\frac{p}{1-p}\right) = \boldsymbol{\beta}.\mathbf{X}$$

where $\boldsymbol{\beta}$ denotes all the weights for the explanatory variables $X_i$, with $\beta_0$ the intercept.

For example, the observations of $Y$ can be the each of the accident report for the case study describe above. As all the observations $Y_k$ are independent, the joint distribution is:

$$f(\mathbf{Y}|\boldsymbol{\beta}) = \prod_k p(Y_k|X_i)$$

The optimal coefficients $\hat{\boldsymbol{\beta}}$ correspond to the maximum of the joint probability density, so we solve:

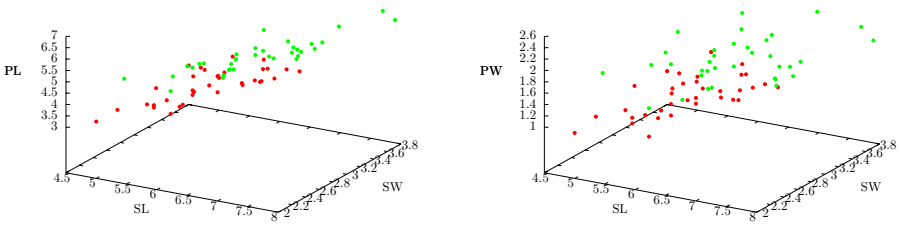
$$\frac{\partial f(\mathbf{Y}|\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$$

The method used to compute those coefficients is the Newton-Raphson method. The *R* software provides a library to compute those coefficients from a data file.

**Example 1: Fisher's iris data set**

The first example that we are going to give is the very popular Fisher's iris data set. There are three species of Iris (*Iris setosa*, *Iris virginica* and *Iris versicolor*) which can be differentiate from one another by the length and the width of the sepals and petals. This data set is commonly used to train a machine learning algorithm to classify the data into the three classes. The first species (*Iris setosa*) can be linearly separated from the others. The other two are are linearly separable like showed on the scatter plot of 2.3.

Figure 2.3: Different combinations of scatter plot showing that the two classes are not linearly separable




(a) SL-SW-PL

(b) SL-SW-PW

To illustrate logistic regression as we will need it for D-Rank, we will use the two species *Iris virginica* and *Iris versicolor*. The objective is to compute the logistic coefficients on a data set to learn how the length and width of the sepals and petals determine to which species the given flower belongs to. Then we test it on another set (for which we know the class of the flower).

The following data is an excerpt from the data, where S denotes *sepal*, P *petal*, W *width* and L *length*. Species 0 is *Iris versicolor* and 1 is *Iris virginica*:

```
 SL  SW  PL  PW Species
6.0 3.4 4.5 1.6 0
6.7 3.1 4.7 1.5 0
6.3 2.3 4.4 1.3 0
...
6.3 2.5 5.0 1.9 1
6.5 3.0 5.2 2.0 1
6.2 3.4 5.4 2.3 1
...
```

We use `R` to compute the coefficients which gives following values.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -41.034     24.954  -1.644   0.1001
SL            -2.339      2.330  -1.004   0.3156
SW            -6.882      4.512  -1.525   0.1272
PL             9.451      4.896   1.930   0.0536 .
PW            16.998      9.431   1.802   0.0715 .
```

Those coefficients can now be used to classify a new flower. Providing the length and width of the petals and sepals, we can compute the corresponding *logit* and the probability of the flower to belong to one of the two classes. Table 2.2 show the predicted classes on the test data (for which we know the species but they were not used in the training). All of them were correctly reclassified. Note that for one of them, the probability is 0.56, which could make the classifier hesitate.

Figure 2.4 shows the probabilities of the testing. It is clear that the coefficients are optimal for the data, as the two classes are correctly separated.

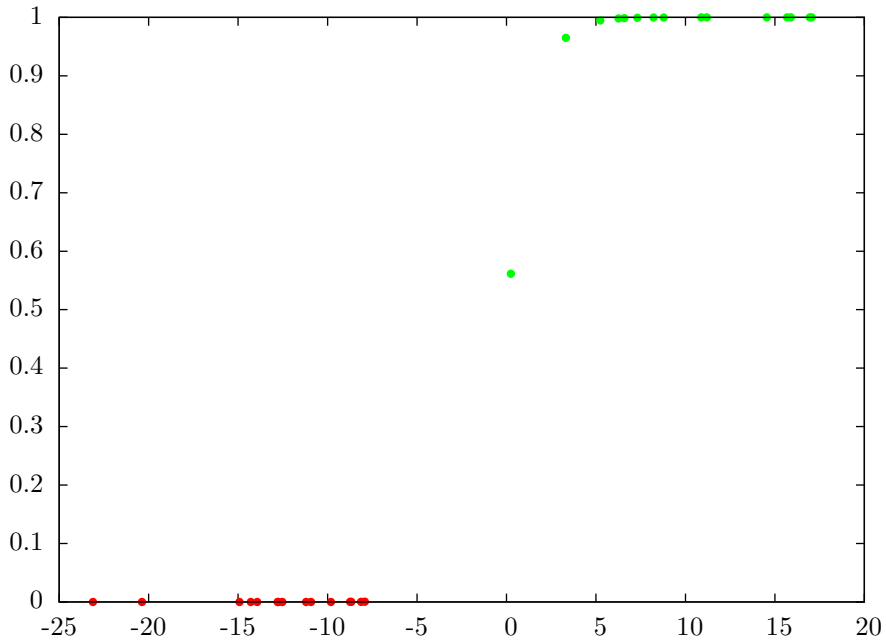



Figure 2.4: Influence of the logistic regression parameters

### Example 2: Epidemiological study

In the previous example, the interpretation of the logistic coefficients is not that intuitive. Interpretation of such coefficients is more common and

Table 2.2: Predicted class after training (excerpt from 15 values for each class)

| SL | SW | PL | PW | Class | Predicted class |
|---|---|---|---|---|---|
| 6.0 | 3.4 | 4.5 | 1.6 | 0 | 1.6e-04 |
| 6.7 | 3.1 | 4.7 | 1.5 | 0 | 2.9e-04 |
| 6.3 | 2.3 | 4.4 | 1.3 | 0 | 3.6e-04 |
| 5.6 | 3.0 | 4.1 | 1.3 | 0 | 8.9e-07 |
| ... | ... | ... | ... | ... | ... |
| 7.7 | 3.0 | 6.1 | 2.3 | 1 | 1.0 |
| 6.3 | 3.4 | 5.6 | 2.4 | 1 | 0.99 |
| 6.4 | 3.1 | 5.5 | 1.8 | 1 | 0.99 |
| 6.0 | 3.0 | 4.8 | 1.8 | 1 | 0.56 |
| ... | ... | ... | ... | ... | ... |

more understandable when used on epidemiological studies. The following example is taken from a case study on hypertension of blood pressure. For this purpose, 9 independent explanatory variables were kept and are presented in table 2.3.

Table 2.3: Explanatory variables to explain the hypertension

| | | | | |
|---|---|---|---|---|
| Sex | Smoker | Exercise | Overweight | Alcohol |
| Stress | Salt | Income | Education | |

The response variable is divided in 4 steps of seriousness: normal, high, very high and severe based on the systolic value of the patient corresponding to hypertension. The input variables are also classified in several classes: sex: M/F, exercise: little, medium, high, ... which is the main difference with the previous example, because the length of the flowers came from measurements, and not categorical classification.

Just as before, we use `R` to compute the logistic coefficients to explain the explain the influence of the parameters from step 2 of hypertension (high) or step 3 (very high). We obtain the following results:

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.844025   1.188434  -1.552   0.1207
sex          0.560075   0.357126   1.568   0.1168
smoker       0.854682   0.351303   2.433   0.0150 *
exercise     0.162729   0.208968   0.779   0.4361
```

```
overweight    0.060407   0.192292   0.314   0.7534
alcohol       0.449477   0.222637   2.019   0.0435 *
stress       -0.492103   0.216350  -2.275   0.0229 *
salt          0.134356   0.207504   0.647   0.5173
income        0.091265   0.218105   0.418   0.6756
education     0.009297   0.208126   0.045   0.9644
```

To understand the meaning of the logistic coefficients, we are going to focus on one particularly variables (smoker). For two identical patients in all categories but *smoker*, the increase of the logit for the smoker patient is 0.854:

$$\begin{aligned} logit(Odd_{\text{smoker}}) - logit(Odd_{\text{non smoker}}) &= \sum_{\text{smoker}} \beta_i x_i - \sum_{\text{non smoker}} \beta_i x_i \\ &= \beta_{\text{smoker}}(1_{(\text{smoker})} - 0_{(\text{non smoker})}) \\ logit(\frac{Odd_{\text{smoker}}}{Odd_{\text{non smoker}}}) &= 0.854 \\ Odd_{\text{smoker}} &= e^{0.854}\ Odd_{\text{non smoker}} \\ Odd_{\text{smoker}} &= 2.34\ Odd_{\text{non smoker}} \end{aligned}$$

In this case, the interpretation of the coefficient has an easy meaning: a smoker has 2.34 as many chances (from the odds point of view) to be in category 3 than 2 in comparison to a non-smoker person.

### 2.3.4 Conclusion on D-Rank

Distributed-ranking has been tested on artificially generated ranked data corresponding to the individual ranking to be aggregated. In our case, we would like to aggregate relevance scores (word similarity) with some quality attributes some of which reflect how a given record was accessed in the system. This aggregation will provide a new ranking score which will be used. This new score will reflect the probability for a document to be relevant to the query by taking into account its relevance and its access frequency in the system.

# Chapter 3

# Attributes to aggregate

This chapter presents the attributes that will be used in the aggregating process. We also present some leads that were discussed over the internship as possible alternatives or improvements to the D-Rank method. In the end, we present the methodology that we chose to work on.

## 3.1 Defining the quality of a record

Our main concern here is to take into account the user behaviour as searches are performed and in particular to interpret the clicks generated bu the users. We would like to quantify in some way the searches.

We are interested in the standard behavior of a user looking for a document in a database: the user enters a pattern in the search engine. This may sounds commonplace, but as stated before, the default display order is *latest first*. Very often, no query is used for the search in order to get the latest documents, especially to look for the last videos or photos.

Once the search is done, the results are displayed to the user. This is the first count that we can take into account to define the quality of a record: the number of times a records was displayed, which we will call *number of displays*.

The user may click on a document in order to read a more detailed description of it. From this new page, (s)he can download the file attached to the record (a download can also be initiated from the results page). These can be considered quality scores: the *number of views* and the *number of downloads*.

Finally, another attribute can be used. If the user clicked (viewed or downloaded) the fourth document without clicking on either of the first three (see figure 3.1), we can reasonably assume that (s)he read the title, the author(s) and the beginning of the abstract of the latter, and therefore that those records did not fulfill his/her expectations. As a result, we can say that those records were *seen*. This is a negative attribute for a document:

the more it has been seen and the less interesting it should be.

Picture 3.1 shows how the attributes can be interpreted from a visual point of view. Note that record displayed in position 4 was clicked, but it is also seen because record at position 5 was also clicked.

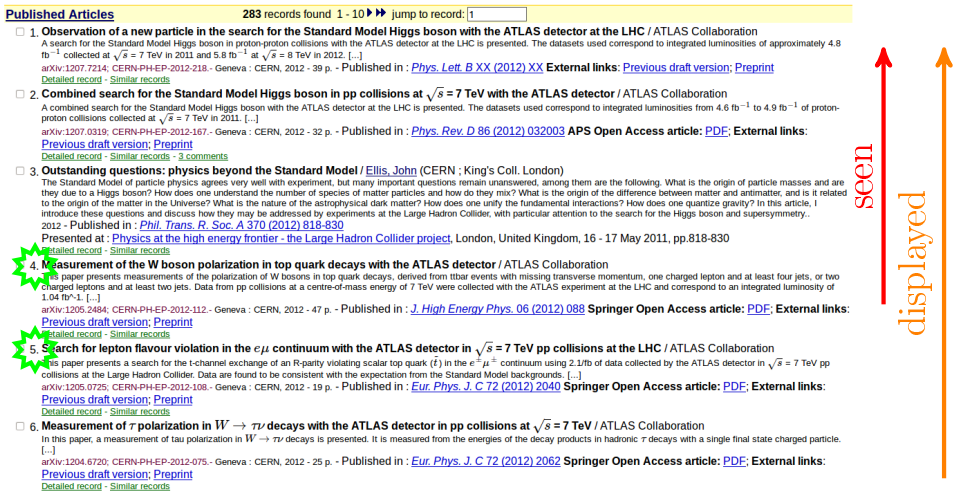



Figure 3.1: Clicks of a search

## 3.2 Fresh vs. mature records

Time should also be an important factor to take into account to rank documents. Indeed, a newly created document cannot possibly have a lot of quality counts.

We can therefore make a distinction between *quality* records (or mature records, which have a sufficient history search in the library) and new documents (which have recently been inserted and did not have time to acquire enough counts) This distinction should prevent those new records to be penalized in the ranking process.

Therefore, fresh records and mature records should be processed separately and merged into a single list in the end, before displaying the results to the user.

However, the notion of *freshness* is subjective. The first idea to define a freshness score computed was to use the number of days since when the record was created in the library (*creation date*). The more recently it has been created into the database, the newer it was considered, thus the freshness score was close to 1. But a record can be very popular from the beginning, and be displayed a lot quickly which would make it comparable with *mature* records quickly. The second idea was then to use a minimum number of

displays as a possible threshold to distinguish between *mature* and *fresh* documents:

- $nb_{displays} > t \Rightarrow$ mature record $\Rightarrow$ use quality
- $nb_{displays} < t \Rightarrow$ fresh record $\Rightarrow$ use time

In addition to those remarks, given two fresh records created at the same date, we could say that if the former has already been displayed and even viewed or downloaded, it is already interesting and should be favored over the latter. As a result, we proposed a new freshness function based on:

- the number of days of the record,
- the number of displays,
- the number of views.

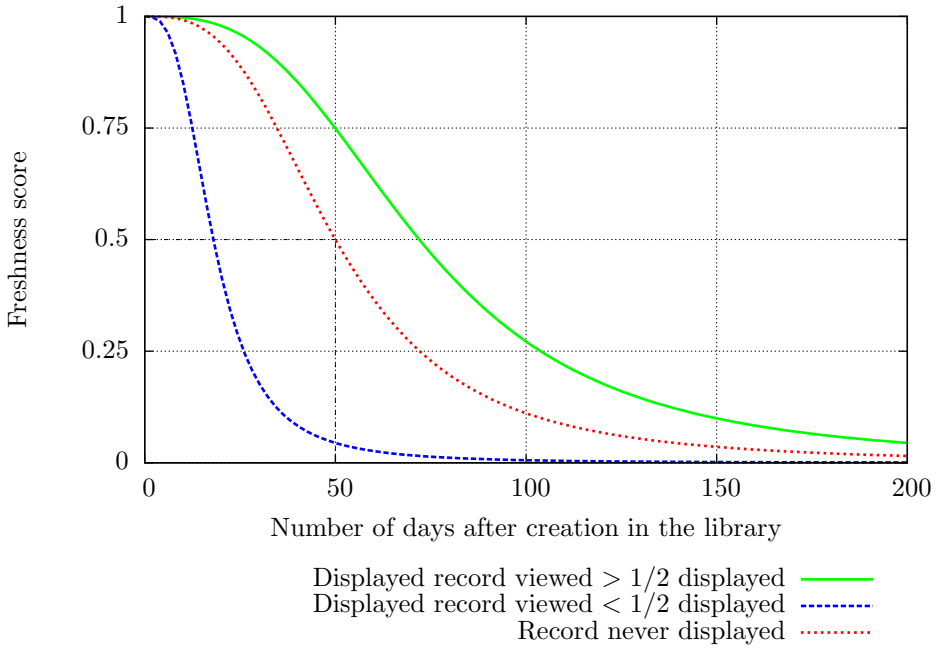



Figure 3.2: Freshness score for different records (half-time = 50 days)

The idea is to say that if a record has never been displayed, its freshness score should be decreasing over the time (the document is less and less considered new in the database, see the red curve on 3.2). If the record has been displayed (while still being considered fresh), we want its freshness score to be modified and take into account the number of times the document was viewed:

- if the record was viewed less than half of its number of displays, its freshness score should be reduced,

- if it was viewed more than half of its displays, its score should be increased.

Picture 3.2 shows how the initial freshness score (red curve) is modified considering the number of views and displays: in green when viewed more than half of the displays, and in blue when viewed less. The base function is:

$$freshness = \frac{1}{1 + \left(\frac{x}{t_{1/2}}\right)^3}$$

and the modification to take into account the number of displays is:

$$freshness = \frac{1}{1 + \left(\frac{x}{2\frac{views}{displays}t_{1/2}}\right)^3}$$

In the two equations above, $t_{1/2}$ is the number of days after which the freshness scores falls bellow 0.5 (see black dashed line on 3.2), and a parallel could be made with the half-life of a radioactive particle in physics.

It is important to note that some records cannot be viewed because they point to an external source. Others can be both viewed and downloaded. It would then be important to take the number of downloads into account: the main difficulty would be to compare records that cannot be downloaded with those which can.

But the main drawbacks of the previous methods is that they require a parameter as input:

- the threshold to distinguish mature (quality) and fresh records,
- the half-time in days.

Those parameters should therefore be changed by the administrator of the digital library. Indeed, as Invenio is used by different institutions with different focuses and types of documents, the content of the collections will be different and it would be difficult of put default parameters.

One idea which was discuss was to analyse the distribution of the quality scores (for instance the number of displays or views) and to separate the documents at some particular point of the distribution. In this case, the threshold could be recomputed regularly and would be dynamic and would fit any institution need. This idea was not further developed due to lack of time.

## 3.3 Aggregation of two scores

As stated before, D-Rank consists in aggregating several scores. In our case, if we aggregate relevance with quality attributes of a record, the logit used in the logistic regression would look like:

$$logit = cst + \underbrace{w_R R}_{\text{relevance}} + \underbrace{w_{vi} vi + w_{do} do + w_{sn} sn}_{\text{quality}}$$

As relevance should be one very important attribute to take into account for ranking, it seems to be lost in the equation among all those attributes. Even if a high coefficient $w_R$ on the *relevance* score could be used, it seems to be less easy to have influence on it. As a result, alternatives to this aggregation were discussed in order to have more influence on the relevance score.

An alternative is to split the logit into two parts like showed on the previous equation:

- one referring to the relevance of the document (which could also be composed of several attributes in addition to word similarity, like the image similarity at search time),
- the other referring to the quality of the record if it is a mature record, or to the freshness if it is a new one.

The *relevance* score $R$ is computed at search time with the words and the fields of the query. However, the *quality* score Q (and the *freshness* score F) correspond to the state of a record in the database and can therefore be precomputed. An alternative aggregation function to use was discussed. It consists in using a modified version of the geometric mean such as:

$$\text{new score } A = R^w X^{1-w},\ 0 \leqslant w \leqslant 1$$

where $X$ denotes either the quality score $Q$ or the freshness score $F$ depending on the nature of the document.

- If $w = 1$, we get $A = R$, only the relevance is taken into account,
- If $w = 0$, we get $A = X$, all the weight is on the other attribute (quality or freshness).

Figure 3.3 shows several examples of the final aggregated score for different combinations of $(R, Q)$. For instance, the green curve corresponds to a relevance of $R = 0.8$ and a quality of $Q = 0.2$. The important point to notice is that when $w = 0.5$, the aggregated score does not correspond to the mean of the two scores.

As a result, we proposed a modified version of this aggregation function of two scores which is:

$$\begin{cases} \text{new score } A & = R + (Q - R)\dfrac{1}{1 + \left(\dfrac{1}{w} - 1\right)^3},\ 0 < w \leqslant 1 \\ & = R,\ \text{if } w = 0 \end{cases}$$

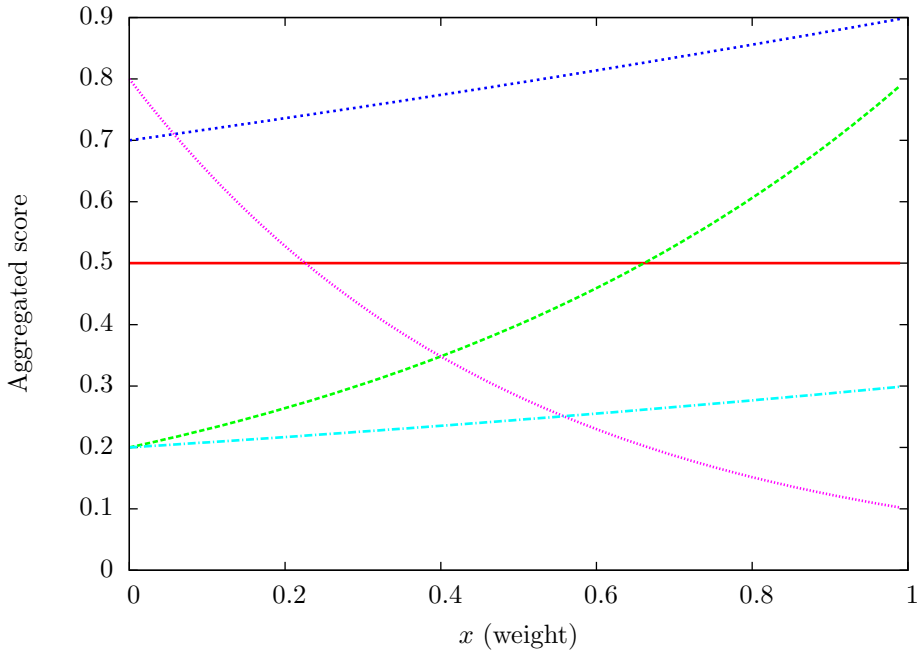



Figure 3.3: Aggregation with $R^x Q^{1-x}$

The same observations regarding the extreme values of $w$ can be made, for which all the emphasis is put on either of the two scores. Figure 3.4 shows the aggregation of the same scores as $w$ varies from 0 to 1.

In this frame of score aggregation, the optimization of the parameters seems to be more difficult:

- first, we have all the weights on the quality scores to compute $Q$,
- second, we have $w_Q$ for the aggregation of quality records with relevance,
- third, we have $w_F$ for the aggregation of freshness records with relevance.

All the coefficients to optimize are no longer in the same equation which will make the optimization much more difficult.

## 3.4 Normalizing the quality regression coefficients to 1

At the beginning of the project, it had been discussed as a possible quality function to use a weighted sum for which the sum of the coefficients would

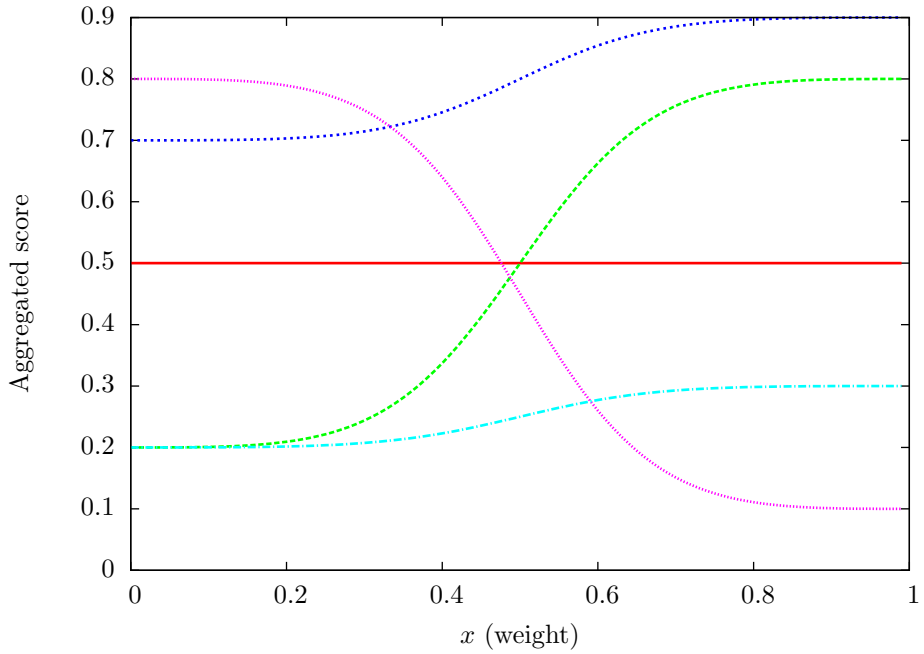



Figure 3.4: Aggregation with $\frac{1}{1+(\frac{1}{x}-1)^3}$

be 1 [21]:

$$Q = \sum \lambda_i x_i, \text{ with } \sum_i \lambda_i = 1$$

where in this case $x_i$ are quality scores such as:

- number of downloads, views, seens or displays,
- number of downloads, views, seens normalized over the number of displays.

If this method has the advantage to produce a quality score $Q$ between 0 and 1, which make the aggregation easy with the previous method, all the coefficients are positive. That is to say that event a negative attribute such as the number of seen will positively influence the quality score, which should not be the case. Just as logistic regression allows negative coefficients, so any aggregation formula for quality based on a weighted sum should allow negative coefficients.

## 3.5 Considered approach during the project

As shown in the previous sections and in the previous chapter, there are many ways to aggregate several scores into a new one, and the choice is not

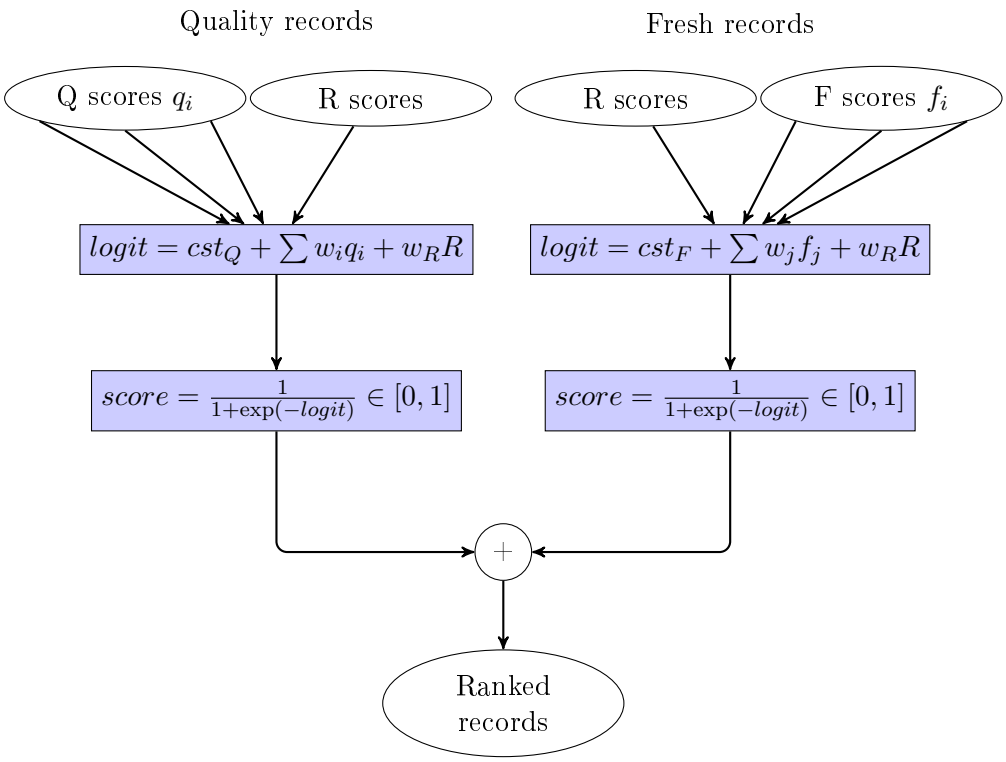



Figure 3.5: General workflow considered to work on during the internship

obvious. Considering the state of the art and also the possible alternatives, we chose to use the distributed-ranking method as it is describe in [19] and to extend it to fresh records as well:

- the first step will be to aggregate quality and relevance for mature records on one side,
- the second will be to aggregation freshness and relevance for fresh records, with the same methods but with different regression coefficients.

Picture 3.5 shows the overview of the aggregation process.

For lack of time, it was not possible to work on both kinds of records, and the freshness was not taken into account. As a result, we only dealt with quality records to find the optimal coefficients of the *logit*. Picture 3.6 show the aggregation as it is made at the end of the internship.

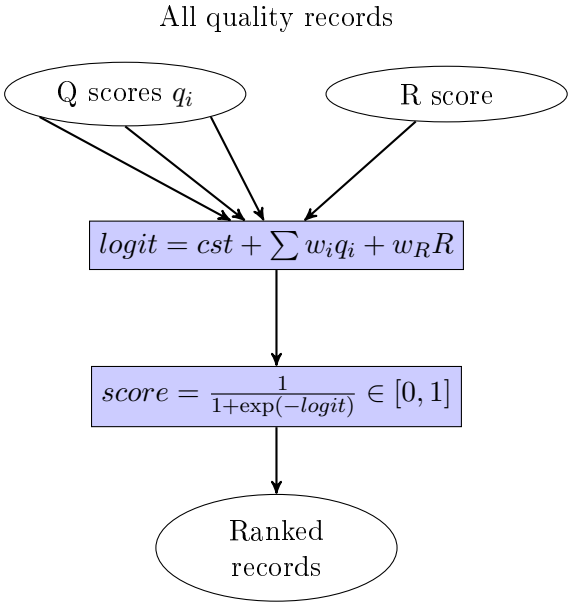



Figure 3.6: Implemented workflow at the end of the internship (the distinction between fresh and mature records is not made). All the documents are considered mature

# Chapter 4

# Implementation of distributed-ranking in Invenio

This sections presents how the D-Rank method is implemented in Invenio. In particular, the way the queries are analysed and how the relevant documents are extracted to compute optimal coefficients.

## 4.1 Workflow

First of all, searches are done by the users on the digital library: queries are made, documents are downloaded, users log in, ...

Then, offline, the module `queryanalyser` analyses the queries made by the users to update the quality counts in table `rnkUSAGEDATA`. This tables contains for each documents how many times it was:

- displayed,
- viewed,
- downloaded,
- seen.

In the same time, it saves in table `rnkWORDSIM` the raw word similarity scores which corresponds to the pattern of the query and the displayed records.

Second, the module `bibrank` used on the D-Rank configuration file normalizes the quality scores from the `rnkUSAGEDATA` table and the word similarity score from `rnkWORDSIM`. Two other tables are thus populated:

`rnkDRANK` this table contains the look-up table and the normalized scores for each record.

`rnkDRANKQUALITY` this table contains the logit quality part ($\sum w_i x_i$) which can be computed offline in the logit formula $logit = cst + \sum w_i x_i + w_R R$

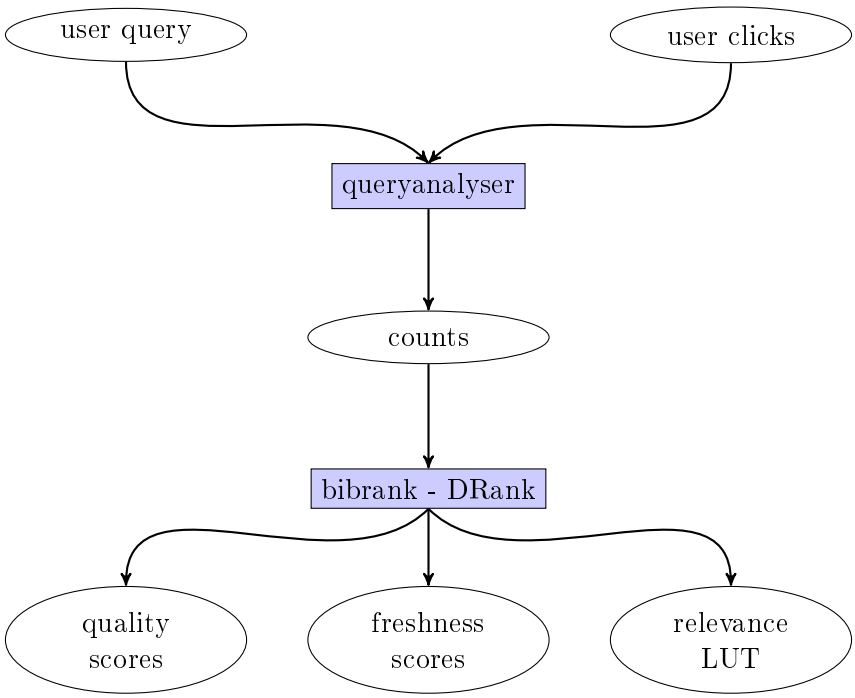



Figure 4.1: General workflow to post-process the user query and normalizing the counts

Picture 4.1 shows the main steps described and picture 4.2 shows the tables involved.

At search time (see picture 4.3), when the user chooses to rank the results with the D-Rank method, the hit set is split between mature and fresh records.

- For mature records, table `rnkDRANKQUALITY` provides the quality part of the logit; table `rnkDRANK` contains the look-up table for word similarity which will give the corresponding normalized score of each document to the query, and the final score is computed by the logistic regression.
- For fresh record, table `rnkDRANK` provides the fresh scores for all known fresh records. If a records has been inserted in the database between the last time the `bibrank` module was ran and search time, the record has not been indexed yet. As a result, its fresh scores will be 1, the maximum value.

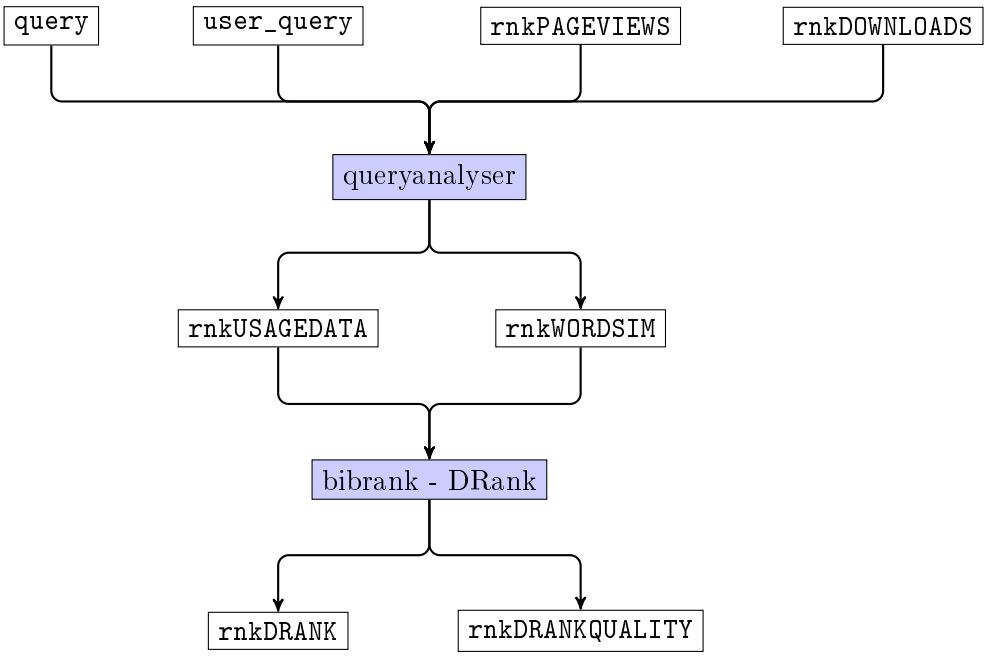



Figure 4.2: Database tables involved in the post-processing

## 4.2 Analysing users queries

The computation of the different quality scores requires to know all the records that were displayed to each user following a search query. It would be inefficient to make those counts at search time. The response time would be too long, and the whole navigation information of the user is needed, which might take several hours when the user reads an article before carrying his/her search.

As a result, the query analysis is divided into two steps:

1. At search time, the queries are stored in the database for each user as well as the actions (abstract viewed and downloads),

2. Offline, the queries and the clicks are reassembled to compute the counts.

### 4.2.1 Storing the actions of the users

Three tables are used to save all the actions of the user. First, the `query` table consists of an identifier and the arguments of the search query (see table 4.1). As an example, on March $16^{th}$ 2012, 7,618,230 different queries had been made since 2002.

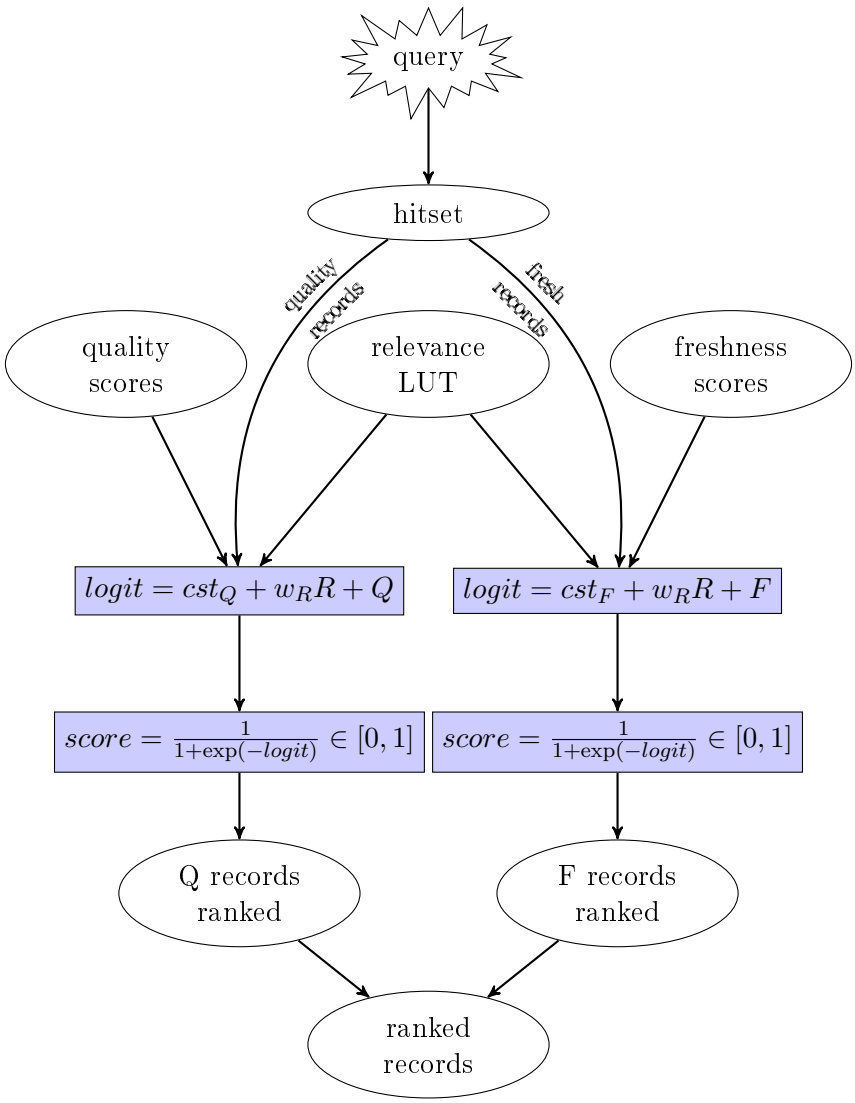



Figure 4.3: General workflow for aggregation of quality and freshness with relevance at search time

Second, the `user_query` table allows to link a user to the query made. Table 4.2 shows the information used to reconstruct the user actions. For display reason, we only put the field `user`. Actually, three different fields are used to identify a user:

- the user identifier (`id_user`),
- the client host (`client_host`) which allows to know where the query was made from,

And third, the actions of the users are stored in three tables depending on the type of the action:

- `rnkPAGEVIEWS` when the user views an abstract,
- `rnkDOWNLOADS` when the user downloads a document from the repository,
- `rnkEXTLINKS` when the user downloads a document stored in an external collection but which is displayed in the results.

When a user does a click (view or download), the identifier of the document is stored (`id_bibrec`) alongside with the user identifier, the date and the referer (the referer is the web-page which the user is coming from), see table 4.3.

Table 4.1: Table query

| id | urlargs |
|---|---|
| 310 | c=Articles+%26+Preprints&p=author%3higgs |
| 2263270 | sc=1&ln=en&p=&f=&action=Search&cc=Photos&c=CMS+Photos |
| 1411986 | sc=1&p=introduction+to+the+elementary+particles&f=&action=SEARCH&cc=Books |
| 1366267 | sc=1&p=arduini&f=&cc=CERN+Document+Server&c=Articles+%26+Preprints |

Table 4.2: Table user_query

| user | date | id_query | reclist | referer |
|---|---|---|---|---|
| 2456215 | 2010-05-04 08:25:52 | 310 | [15, 19, 24, ...] | http://cdsweb.cern.ch/... |
| 24785412 | 2010-05-04 10:02:47 | 21542 | [48, 57] | http://www.google.fr/... |

Table 4.3: Table rnkPAGEVIEWS (similar to rnkDOWNLOADS)

| id_bibrec | id_user | view_time | referer |
|---|---|---|---|
| 19 | 2456215 | 2010-05-04 08:26:15 | http://cdsweb.cern.ch/search?p=higgs... |
| 48 | 2574444 | 2010-05-04 11:25:25 | http://www.google.fr/search?q=... |
| 95147 | 1207374 | 2010-10-02 10:04:59 | http://cdsweb.cern.ch/search?cc=ATLASjrec=11&p=heinemann&f=author |

### 4.2.2 Counting quality scores

The previous section described how the user actions were stored at search time. Now, we are going to see how the queries are analysed (which is done offline, later on).

First of all it is important to note that all the documents matching a query are not displayed on the first page. By default, the number of displayed document on a page is 10, and to see the following documents, the user can click on "next page" which constitutes a new query. If the user searches for "higgs boson", the first ten results (records from position 1 to 10 for instance sorted by *latest first*) will be displayed on page :

`http://cdsweb.cern.ch/search?ln=en&c=Articles&p=higgs+boson`

When the user clicks on the following page to display result from 11 to 20, the new URL is

`http://cdsweb.cern.ch/search?ln=en&cc=Articles&jrec=11&p=higgs+boson`

where field `jrec` indicates from which position the display starts. This is therefore a new row in table `user_query`. As an example, we suppose that a user performs those two queries and clicks on record 123 and 3, and then on record 87 on the second page (see table 4.4).

Table 4.4: Clicked records on the two pages

| **First page** | | | **Second page** | | |
|---|---|---|---|---|---|
| Position | identifier | Clicked? | Position | identifier | Clicked? |
| 1 | 156 | | 11 | 18 | |
| 2 | 235 | | 12 | 6 | |
| 3 | 48 | | 13 | 19 | |
| 4 | 4 | | 14 | 87 | X |
| 5 | 123 | X | 15 | 15 | |
| 6 | 97 | | 16 | 20 | |
| 7 | 44 | | 17 | 12 | |
| 8 | 3 | X | 18 | 255 | |
| 9 | 72 | | 19 | 1024 | |
| 10 | 13 | | 20 | 7 | |

At this point, table `query` has two rows which are given in table 4.5 and table `user_query` (table 4.6) contains the two queries made by the user. When the user clicks on the different records, table `rnkPAGEVIEWS` is populated as describe by table 4.7.

Queries are processed one by one:

Table 4.5: Queries made

| id | urlargs |
|---|---|
| 1 | `ln=en&c=Articles&p=higgs+boson` |
| 2 | `ln=en&cc=Articles&jrec=11&p=higgs+boson` |

Table 4.6: Attributes of the two queries made

| user | date | id query | reclist |
|---|---|---|---|
| 7 | 2012-09-09 08:00:00 | 1 | [156, ..., 13] |
| 7 | 2012-09-09 08:01:00 | 2 | [18, ..., 7] |

| referer |
|---|
| `http://cdsweb.cern.ch/` |
| `http://..../search?ln=en&c=Articles&p=higgs+boson` |

Table 4.7: Recorded actions

| record | id_user | view_time |
|---|---|---|
| 46 | 7 | 2012-09-09 08:00:20 |
| 43 | 7 | 2012-09-09 08:00:40 |
| 37 | 7 | 2012-09-09 08:01:30 |

| referer |
|---|
| `http://.../search?ln=en&c=Articles&p=higgs+boson` |
| `http://.../search?ln=en&c=Articles&p=higgs+boson` |
| `http://.../search?ln=en&cc=Articles&jrec=11&p=higgs+boson` |

**counting display** for each of the query, table `user_query` gives the records which were displayed at search time. The number of displays for those records will be incremented by 1.

**counting the clicks** for all the clicks made, the referer is used to know to which query the click belongs to. In addition to that, the user identifier (and the client host) to differentiate users performing same searches simultaneously. The date stored in both `user_query` and `rnkPAGEVIEWS` allows to check that the click really came as a result of the search.

**counting seens** as a reminder, a record is seen by the user if it is not clicked and if a record below is clicked. On the first page, the last clicked record is at position 8. Therefore, the number of seens for records

from position 1 to 7 will be incremented. On the second page, records from position 11 to 13 will be added one seen count. In addition to this, as the user went to the second page, it also implies that (s)he saw all the documents displayed by the referer, which (s)he is coming from. As a result, records from 1 to 7 get an extra seen counts. If the user went to the third page, all the documents of the second page will get one more extra seen count.

After the post-process of those two queries, the documents will have counts presented in table 4.8

Table 4.8: Final counts for the analysis of the two queries

| id | displays | views | downloads | seens |
|---|---|---|---|---|
| 156 | 1 | 0 | 0 | 2 |
| 235 | 1 | 0 | 0 | 2 |
| 48 | 1 | 0 | 0 | 2 |
| 4 | 1 | 0 | 0 | 2 |
| 123 | 1 | 1 | 0 | 2 |
| 97 | 1 | 0 | 0 | 2 |
| 44 | 1 | 0 | 0 | 2 |
| 3 | 1 | 1 | 0 | 1 |
| 72 | 1 | 0 | 0 | 1 |
| 13 | 1 | 0 | 0 | 1 |
| 18 | 1 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 1 |
| 19 | 1 | 0 | 0 | 1 |
| 87 | 1 | 1 | 0 | 0 |
| 15 | 1 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 |
| 255 | 1 | 0 | 0 | 0 |
| 1024 | 1 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 |

As we can see on this example, records from 156 to 44 are seen twice:

- one because the were seen when the first page was displayed,
- and another time when there are considered seen by displaying the second page.

The is a misconsideration of the original implementation which could be corrected. The first drawback of this is that a record can be more seen than displayed, and the second one is that it gives more counts to some records

and will penalize them later on, because the attributes seen is not a positive attribute for a record.

## 4.3 Normalizing the counts

### 4.3.1 Implementation and tables involved

As described before, the attributes values have to be aggregated before being used in the logistic regression formula. This is performed by the module `bibrank`.

The module gets all the counts computed before and saved in table `rnkUSAGEDATA` (see the general workflow and the tables involved from pictures 4.1 and 4.2). The distribution of scores is estimated by the python function `gaussian_kde` and the cumulative distribution function gives the normalized scores.

Table `rnkDRANK` (4.9) stores the look-up table of the attribute, which is the correspondence between the raw scores and the normalized scores, and the normalized scores for each record. The quality part of the logit can also be precomputed to be then used at search time to save computation time. This precomputed part is stored in table `rnkDRANKQUALITY` (4.10), which was created in the database for the purpose of the project. The LUT for word similarity it computed with the raw word similarity scores which were saved in table `rnkWORDSIM` during the analysis of the queries by the `queryanalyser`. There are no normalized scores for word similarity as they are query dependent and only known at search time.

Table 4.9: Table rnkDRANK

| id | name | lut | percentiles |
|---|---|---|---|
| 1 | nb_views | {0:0.2, 1:0.3, 9:0.5, ... } | {154: 0.4, 9856:0.2, ... } |
| 2 | nb_seens | | |
| 3 | nb_displays | | |
| 4 | nb_downloads | | |
| 5 | word | | Does not exist |

### 4.3.2 Analysis of the distribution of scores

Picture 2.2 showed the normalized scores for a random distribution of scores. The scores where ranging from 0 to 1 with enough disparities among all the initial scores.

Table 4.10: Table rnkDRANKQUALITY

| name | percentiles |
|---|---|
| drankArticles | {154: 0.54, 9856: 2.45, ... } |
| drankCustom | |

Working on the real scores from table `rnkUSAGEDATA` provides a completely different distribution (see picture 4.4 for views, but the distribution is the same for downloads, displays and seens). Lots of queries are made and yet few documents are clicked. As a result, more than 75% of the documents have 0 view and the normalized score for views (which is the cumulative of the distribution) gets a high normalized score. There is therefore little disparity between records viewed more than 3 times which will all be normalized with a score close to 1.

On the contrary, the normalization for word similarity is much more regular (see figure 4.5). The initial raw scores range from 0 to more than 100 (for very relevant queries) but we only displayed it to 40. We can also note that this distributed is quite stable over the months which shows a common use of the digital repository.

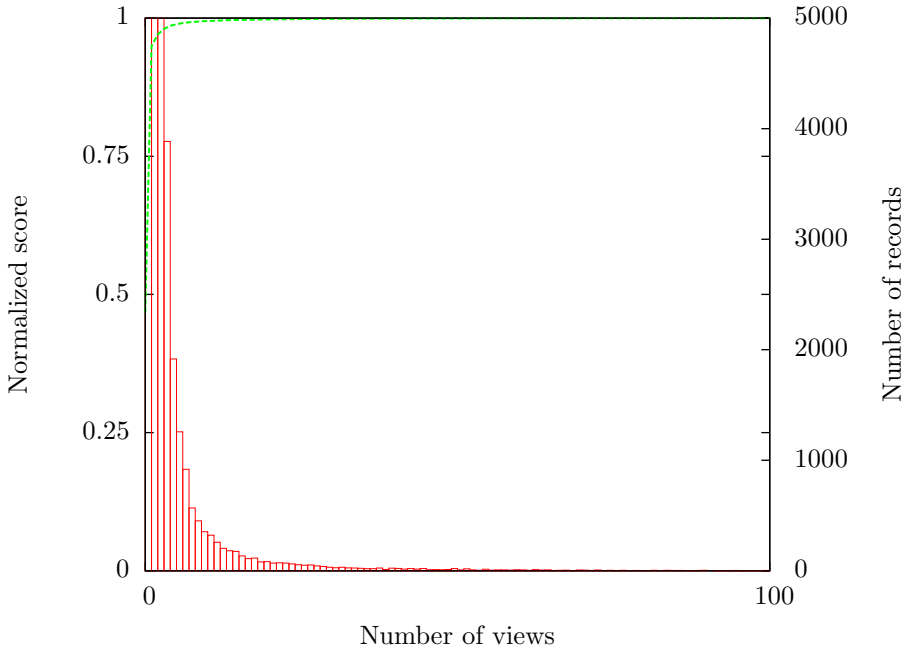



Figure 4.4: Normalized scores for views

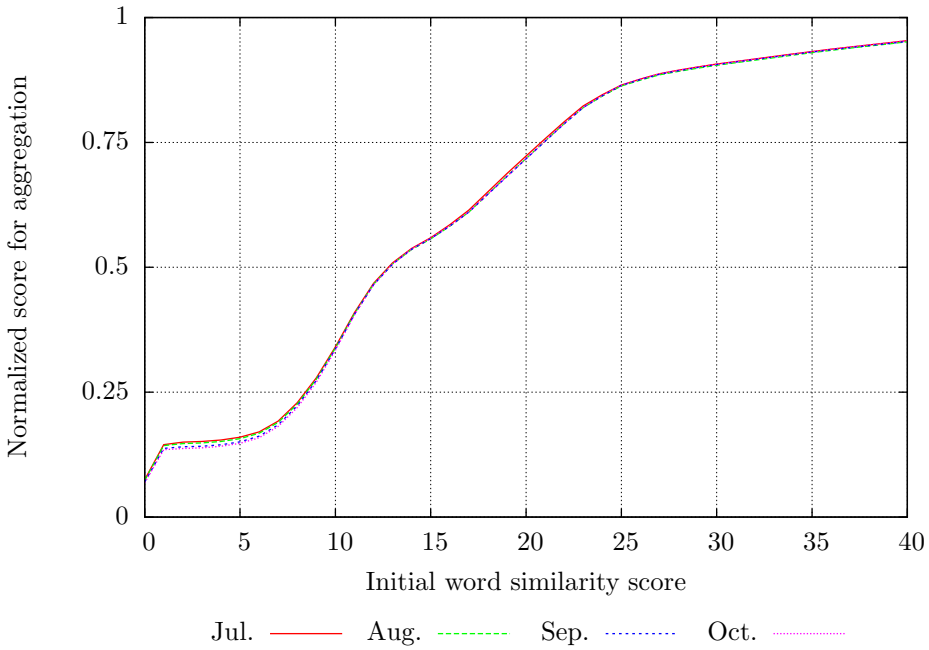



Figure 4.5: Normalized scores for word similarity

## 4.4 Ranking with D-Rank at search time

### 4.4.1 General D-Rank search

At search time (see figure 4.3), the hit set is split into two parts: the mature and the fresh records.

For both, word similarity scores are computed with respect to the pattern and are normalized using the LUT from `rnkDRANK.word`.

For quality records, the precomputed quality part Q stored in `rnkDRANKQUALITY` is aggregated with the word similarity score R. For fresh record, the precomputed freshness F is in table `rankDRANK.freshness`. We have:

$$logit_Q = cst_Q + w_R R + Q$$

$$logit_F = cst_F + w_R R + F$$

The final score is computed with the logistic regression function:

$$score = \frac{1}{1 + \exp(-logit)} \in [0, 1]$$

In the end, the two lists are merged to produce the final ranking and are displayed to the user.

### 4.4.2 Customization of the weights at search time

More and more applications let users customize their own searches. As three main attributes are aggregated here (relevance, quality and freshness), even if we reach an optimal combination of those scores, we would like to let the possibility to the user to customize its search by putting the weight of its choice on the attributes.

As a reminder, the logit for the two aggregations is:

$$logit = cst + \underbrace{w_R R}_{\text{relevance}} + \underbrace{\sum w_i q_i}_{\text{quality}},$$

where $q_i$ is views, downloads, ... and

$$logit = cst + \underbrace{w_R R}_{\text{relevance}} + \underbrace{\sum w_i q_i}_{\text{freshness}},$$

The two aggregation functions have been modified to add an additional weight on the attributes:

$$logit = cst + x_R(w_R R) + x_Q \sum w_i q_i$$

$$logit = cst + x_R(w_R R) + x_F F$$

The three values $x_R$, $x_F$ and $x_Q$ can be selected via sliders on the web page at search time when this ranking method is used (see figure 4.6). We should have $x_R + x_F + x_Q = 1$ and all positive values. The picture shows three sliders but in fact the weights are linked. A better slider should look like picture 4.7 but there was no time to develop it.

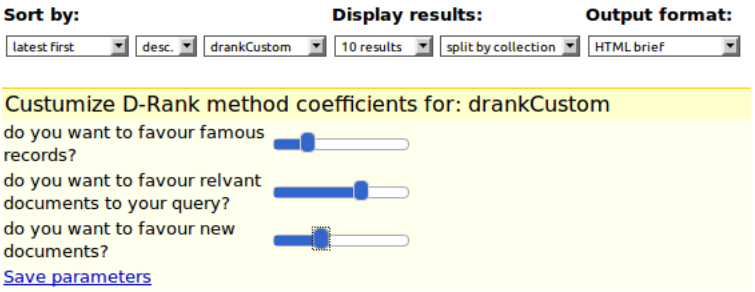



Figure 4.6: Sliders to choose the coefficients

At the end of the internship, we did not have time to work more in details on the aggregation of freshness. As a result, we only provide one slider (see picture 4.8) to put emphasis either on quality or on relevance such as

$$logit = cst + (1 - x) \times Q + x \times R,$$

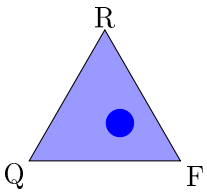



Figure 4.7: Triangle slider for customizing the weights

where $0 \leqslant x \leqslant 1$ is the value of the slider. Choosing value 1 for the slider will put all the weight on relevance and the final ranking will be equivalent to word similarity; choosing value 0 will put all the emphasis on quality and the most famous records will be displayed first. In between, the two features will be mixed and the final score will take both of them into account.

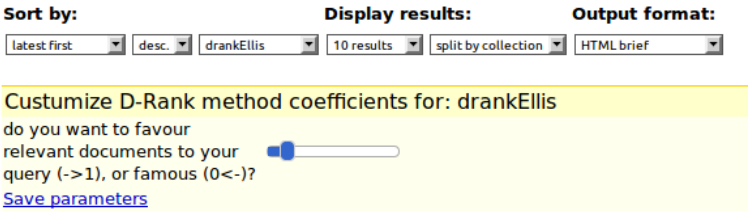



Figure 4.8: Slider currently implemented

## 4.5 Conclusion on the implementation

The implementation of *D-rank* relies on two main steps:

- the analysis of the queries to count how many times the records were accessed throughout the queries,
- the normalization process and the precomputation part of the fresh and quality scores.

At search time, the word similarity score is normalized before being aggregated with the quality of freshness part. The final ranking therefore takes into account the relevance of the document to the query and also the access frequency of the document.

For now, we have not talked a lot about the coefficients used for aggregation, i.e. the weights we put on the attributes which participate in the final scores.

One idea would be to choose manually coefficients, with *a priori* knowledge on the influence of each attribute.

But as seen before, logistic regression coefficients can be precomputed from the data to explain the outcome of the binary variable. In our case, we have the queries and what the users did, following chapter will present how we used this user feedback to computed the optimal logistic regression coefficients.

# Chapter 5

# Learning the optimal coefficients from the user feedback

This chapter presents how the optimal regression coefficients are estimated to fit the user expectations. The purpose is to train the coefficients on a five-month set of queries and then to test it on the sixth month, to see if the records which were clicked at that time are ranked higher.

## 5.1 Building the observations

As presented in the logistic regression part, the regression coefficients are estimated from the observations. In the examples given, the hypertension level was explained by the fact that the patient smoked, practiced a sport,... and the class of the Iris was predicted from the features of the petals and sepals, and the observation looked like the following:

```
 SL  SW  PL  PW Species
6.0 3.4 4.5 1.6 0
6.7 3.1 4.7 1.5 0
...
6.3 2.5 5.0 1.9 1
6.5 3.0 5.2 2.0 1
...
```

In our case here, we make the assumption that if a record was clicked (abstract viewed or downloaded), it was relevant for the user (class 1). If a record is seen without being clicked, we consider it an irrelevant document (class 0). When a record is displayed and beyond the last clicked one, we do not take it into account, because we cannot know if the user saw it.

We are going to use the following attributes:

- number of views,
- number of displays,
- number of downloads,
- number of seens,
- word similarity to the query.

As queries are analysed by the `queryanalyser`, observations can be made on whether a document is relevant or not and a file is created: the relevant field corresponds to the binary variable success/failure of the logistic model (is the document relevant?) and the other fields consists in the explanatory variables taken into account to explain the response variable.

| relevant | recid | word | nb_views | nb_downloads | nb_displays | nb_seens |
|---|---|---|---|---|---|---|
| 0 | 1048741 | 11 | 0 | 0 | 9 | 6 |
| 0 | 1083536 | 12 | 0 | 0 | 9 | 6 |
| 0 | 986518 | 11 | 0 | 0 | 1 | 1 |
| 0 | 1101914 | 12 | 0 | 0 | 8 | 6 |
| 1 | 978015 | 31 | 1 | 3 | 6 | 2 |
| 0 | 1041754 | 8 | 0 | 8 | 6 | 1 |
| 0 | 1007493 | 12 | 0 | 0 | 4 | 3 |
| 0 | 322566 | 15 | 0 | 1 | 4 | 7 |
| 0 | 322567 | 12 | 0 | 0 | 5 | 7 |

## 5.2 The data set

The queries made by the users have been recorded from May $4^{th}$ 2010 to February $16^{th}$ 2011. The data consists of 1139832 queries. It is important to remind that when the user clicks on the next page to display the following documents, this is considered a new search. As a result, lots of the queries are linked together.

It is also important to note that the documents of CDS are indexed by other search engines such as Google. As a result, documents can be downloaded by a user coming from a Google page (the referer of the download will look like `http://www.google.com/search?...`)

During this period:

- 4,707,398 abstracts were viewed. 3,722,795 had a referer coming from CDS but only 384,600 from searches through CDS search engine. (see table 5.1)
- 2,733,230 records were downloaded. 1,542,774 had a referer from CDS,
- 120,524 external documents were found among which 113,286 from CDS.

Table 5.1: Referers of the abstract views. *Our only interest will be views coming from search pages*

| Referer prefix | Number of views |
|---|---|
| `http://cdsweb.cern.ch%` | 3,722,795 |
| `http://cdsweb.cern.ch/search%` | **384,600** |
| `http://cdsweb.cern.ch/collection%` | 152,154 |
| `http://cdsweb.cern.ch/journal%` | 7,331 |
| `http://cdsweb.cern.ch/event%` | 8,371 |
| `http://cdsweb.cern.ch/record%` | 3,140,068 |
| other | 30,271 |

The majority of the results were displayed by *latest first*: the new documents are displayed first and only 11,646 queries were ranked by word similarity.

## 5.3 Methodology

As we would like to compute optimal coefficients for the repository, we will only take into account the queries made from CDS, i.e. referer like

`http://cdsweb.cern.ch/...`

Indeed, we want to study the user behavior on the repository only. Similarly, only clicks following a CDS query will be kept, for which referer is like

`http://cdsweb.cern.ch/search?...`

Therefore, if a user did a search on Google, which redirected him/her to CDS, this click is not taken into account.

As for the downloaded documents, we are interested in those:

- either coming directly from a search (referer like `http://cdsweb.cern.ch/search?`),
- or coming after the corresponding abstract page has been viewed.

As usually in artificial intelligence, when it comes to learning some parameters from a data set, it is important to test them on a another data set to avoid over-training.

In our case, we have more than eight months of data (from May to the beginning of February). As the user behaviour might be different at the end of the year (Christmas, ...), we chose to retain queries from May to October. The five first months of data will be analysed to extract the relevant

and irrelevant documents for each query and the optimal coefficients will be estimated over those queries. Then, those coefficients will be used to re-rank the queries made in October. For those queries, we also have:

- the records which were displayed,
- the records which were viewed/downloaded.

Our purpose is to see whether the documents that were clicked at that time are better ranked with D-Rank, optimized over the previous period.

## 5.4 Defining a distance from the first rank

Our interest here is to compare the original ranking at the time the queries in October were made with the new ranking given by the D-Rank method. In particular, we would like to see how the records of interest are ranked.

There are a lot of methods in the literature to compare several ranking returned by search engines. Those methods are mainly based on the number of permutations between the rankings. But our concern here is different: we would like to see how the relevant records are newly ranked, and we do not care about those which were not clicked by the user.

Table 5.2: Example of initial ranking and re-ranked results

| Display position | Initial ranking | D-Rank |
|---|---|---|
| 1 | 47 | 12 |
| 2 | 18 | 11 |
| 3 | 17 | 10 |
| 4 | 16 | 14 |
| 5 | 15 | 13 |
| 6 | 14 | 17 |
| 7 | 13 | 18 |
| 8 | 12 | 47 |
| 9 | 11 | 15 |
| 10 | 10 | 13 |

We would like to evaluate the performance of the re-ranking from a quantitative point of view. Table 5.2 shows an example of an initial ranking and the new ranking obtained. The green records correspond to the ones which were clicked, so which we will be consider relevant. As we can see, record 12 was better ranked (from position 8 to position 1) but record 15 was worse ranked (from position 5 to 9).

From this observation, how can we evaluate this new ranking: is it better or worse?

- on the one hand, we could say it is better because one record gained 7 ranks and the other only lost 4.
- But on the other hand, more records will be seen, because the user will have to go down to position 9 to find what (s)he is looking for.

Since it seems to be subjective depending on the point of view, we choose to work with several criteria:

- the distance from the first rank,
- a modified distance from the first rank to prevent outliers to have too much influence on the overall records of interest,
- the number of documents better/worse ranked,
- the number of relevant documents on the first page.

### 5.4.1 Distance from top position

The natural way to evaluate the ranking of an element is its distance from the top position. To evaluate the overall ranking of the documents of interest, we sum all the distances from the top:

$$D = \sum_i r_i$$

where $r_i$ denotes the position of the $i^{th}$ record of interest. The difference between the initial distance and the new distance can reflect how good or bad the new ranking is. In the previous example, the initial distance is $4 + 7 = 11$ and the final distance is $0 + 8 = 8$ which is a good point.

But if record would have been rank much lower (more than rank 20 for instance), the final distance would have been completely different and much bigger. This method is sensitive to outliers. This is all the truer so, because if we had 5 records of interest, and 4 of them re-ranked in the first 4 positions, and the last one at position 20, the difference between initial and final distance is likely to be negative, although we could consider the new ranking better because 4 of the records of interest are ranked at the first 4 positions.

As a result, we modify this distance to be less sensitive to outliers.

### 5.4.2 Modified distance from top position

This method is inspired from [17] previously discussed. The idea is to say that after a certain position, the difference between two ranks is less important than at the beginning of the ranking. Instead of taking the rank

itself as the distance from the top (which is always increasing), we modify it to be bound, as follows:

$$D = 1 - \frac{1}{1 + \left(\frac{x}{10}\right)^3}$$

The factor 10 is the rank for which the distance is 0.5 (i.e. half of its maximum value). The value was chosen because the default number of display document is 10. After 10, the user has to click on the next page to see the following records. Picture 5.1 shows the corresponding function.

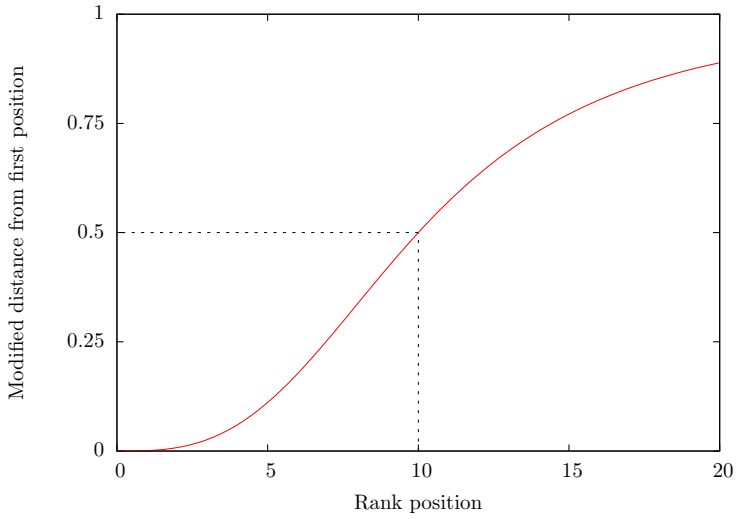



Figure 5.1: Modified distance from the top position

In this case, if a document is ranked at position 20 or 40, its modified position will not affect that much the overall cumulative distance of the documents of interest.

Once optimal parameters are estimated, the purpose is to re-run those queries, rank the records with those parameters, recompute the distance of the records of interest from position 1 and see if the distance has decreased.

To avoid an over-learning, optimization should be perform on a set of queries and verification on an other set (e.g. find optimized parameters on queries ran for two months and use those parameters on the third month).

## 5.5 Estimation of logistic regression coefficients

### 5.5.1 Values of the coefficients

The logistic coefficients were estimated on queries made from July to September 2010. We used the `R` software and in particular the `glm` (general-

ized linear model) function on the data file made during the analysis of the queries.

Data from May to June was only used for the counts to let enough records to be accessed a sufficient amount of time. We also wanted to observe the impact of the training set size on the coefficients. So we first used one-month data (July), then two and eventually three (from July to September).

Table 5.3 presents the coefficients that we estimated. First, we used the whole data for the estimation, then only documents related to scientific publications (articles, preprints, ...). For the two first computations, the formula used in the model was:

```
relevant ~ word + nb_displays + nb_views + nb_downloads + nb_seens
```

In the last computation, we took the number of displays off the formula.

```
relevant ~ word + nb_views + nb_downloads + nb_seens
```

Indeed, we can state that only downloads, views or seen really reflect the quality of a document, and that a document matching a lot of queries with key words (as during a important event) should not be favored upon others in a more precise query. The number of displays for a document is characterizing more the search behaviour than the quality of the records.

Table 5.3: Estimated coefficients

| **All documents** | relevance | display | view | download | seen |
|---|---|---|---|---|---|
| Jul. | 0.544 | 2.67 | 2.99 | 1.73 | -7.12 |
| Jul. Aug. | 0.634 | 2.65 | 2.92 | 1.92 | -6.96 |
| Jul. Aug. Sep. | 0.726 | 2.67 | 2.79 | 2.04 | -7.03 |
| **Articles** | | | | | |
| Jul. | 0.161 | 2.82 | 3.08 | 1.81 | -8.31 |
| Jul. Aug. | 0.263 | 2.79 | 2.23 | 2.98 | -8.15 |
| Jul. Aug. Sep. | 0.366 | 2.49 | 2.19 | 3.52 | -8.02 |
| **All Documents** | | | | | |
| Sep. | 1.015 | $\emptyset$ | 2.805 | 2.422 | -4.353 |
| Jul. Aug. Sep. | 0.7047 | $\emptyset$ | 3.1073 | 2.1502 | -4.2916 |

### 5.5.2 Interpretation

#### Sign of the coefficients

The coefficients for word similarity, displays, views and downloads are positive. This means that these attributes have a positive effect on the

outcome of the final aggregated score: the more a record had been viewed or downloaded, and the bigger its final score will be. We can make the same observation for word similarity.

On the contrary, the sign of the seen coefficient is negative, which shows that this attribute penalizes the final score of the document.

These results were expected in the sense that they correspond to the intuition that one can have of the attribute meanings. When a record is displayed first and not clicked, it is not relevant. So for later queries, it should be lower ranked. And when a record is downloaded or viewed but with few seen counts, it means that the users stopped their search after clicking it. So for future queries, not only will it be favored because of its access counts, but in addition to that it won't be penalized by its seen counts.

#### Order of magnitude of the coefficients

What is more surprising in these results is the order of magnitude between the coefficients. Indeed, the word similarity score (relevance in the table) is between 6 and 10 times smaller than the quality coefficients. It means that only a small change in the counts of the quality features can greatly affects the final score in comparison to word similarity. As a result, a much less relevant document to the query yet quite famous will be better ranked than a very relevant document yet little accessed.

The possible causes explaining these values will be discussed at section 5.7.

#### Stability

Over the three months, we can notice little evolution in the coefficients and the values are quite the same for each of them. This means that the training set size has little influence to compute optimal coefficients.

## 5.6 Re-ranking the queries

Our purpose here is to use the queries made in October by the users for which we know:

- the records which were displayed,
- the records which were clicked.

We are going to rank the hit set as if the users had chosen the D-Rank method to display the results. The D-Rank method will use the coefficients that we estimated over the three previous months. Then, we are going to compare the initial ranking with the new ranking using the criteria described in section 5.4. We are only interested in the records of interest, those clicked

at the time by the user. The goal is to see whether or not those documents are better ranked.

Table 5.5 presents the results obtained. Only 20,965 queries in October could have been retained for re-ranking. Indeed, we were interested in queries reflecting a "normal" search. For this purpose, we removed queries:

- without any pattern,
- containing a pattern which was a regular expression (`/^j*/`),
- related to a report number or containing only numbers,
- which did not lead to any clicks.

Out of those queries, we kept queries containing more than one displayed record which gave 14421 queries for re-ranking.

Looking at the raw number of records ranked better or worse, 15,114 were better ranked by D-Rank. 537 of them were fresh (that is to say they were not quality record at the time and the logistic coefficients were computed to better ranked quality records). So we are not going to take them into account. 11,688 records were ranked worse by the method, out of which 419 of them were fresh. In addition to that 3598 of the worse ranked records were clicked for the first time in October: that is to say that is was the first time those records were relevant for the user. So they had a very low quality part in the logit and were therefore lower ranked. In the end, 11,776 quality records were better ranked (60%) and 7671 were worse ranked.

Considering the modified distance presented at section 5.4.2, 8,336 queries (58%) were considered better ranked: it means that the average distance from position 1 of all the clicked records was smaller; the records of interest were closer to the beginning of the ranking. 6085 (42%) were considered worse ranked. If we do not take into account the fresh records in the evaluation, 8,889 queries are better ranked (62%) and 5532 are worse.

## 5.7 Comments of the results

On the one hand the sign of the estimated coefficients are coherent. On the other hand, the order of magnitude are quite unexpected considering the low value for word similarity and also the percentage of better ranked queries (64%) and of better ranked quality records (61%). Several reasons may explain these results.

### 5.7.1 Normalization

As we saw on picture 2.2, the distribution of scores is such as lots of documents have the same value for a given attribute (0 for seen and downloads and 1 for displays and seens). As the normalization is supposed to reflect the distribution (percentile level described at section 2.3.2), those values get

Table 5.4: Results of re-ranking in October

| | |
|---|---|
| **General figures** | |
| Total queries | 20,965 |
| Queries with clicks and $> 1$ display | 14,421 |
| **Modified distance** | |
| Better ranked queries | 8,336 |
| Worse ranked queries | 6,085 |
| Better ranked (without fresh records) | 8,889 |
| Worse ranked queries (without fresh records) | 5,532 |
| **Records** | |
| Better ranked records | 15,114 |
| Worse ranked records | 11,688 |
| **Fresh records** (no quality scores) | |
| Better ranked | 537 |
| Worse ranked | 419 |
| Better clicked for the first time in October | 2,801 |
| Worse clicked for the first time in October | 3,598 |
| **Only quality records** | |
| Better ranked records | 11,776 |
| Worse ranked records | 7,671 |

a high normalized score from the beginning (0.47 for 0 views for example), which leave only 0.53 for all the other scores (from 1 to 50 views). In addition to that, lots of documents have 1 views and the normalized is 0.95. As a result, records having more than 2 views will have a normalized score between 0.95 and 1. Very little distinction is thus made between records having 3, 5 or 50 counts. The normalisation for those scores is more like either 0.5 or 1.

Moreover, the normalisation process estimates the density with gaussian techniques but as the interval is compact on one side (starting from 0), a bias is introduced. And as the distribution is discrete, the first score has a non null score. As a result, a document which is not viewed gets a non null normalized score: the corresponding logit is increased even if the document has no views. Unfortunately, this problem was discovered late during the work and there was no time to adjust the normalisation.

As far as the word similarity normalization is concerned, it is performed on all the raw scores obtained over the months: all the different queries are put at the same level. In the end, a query with few terms will be normalized with a small score whereas a detailed query will have a large normalized score. In the first case, quality will be naturally more emphasised. Another

Table 5.5: Queries with more than 10 displays (several pages)

| | |
|---|---|
| Number of queries with > 10 displays | 4707 |
| **Better ranked records** | |
| Better ranked record (still on the first page) | 1056 |
| Better ranked (not on the first page yet) | 3529 |
| Better ranked (now on the first page) | 2584 |
| **Worse ranked records** | |
| Worse ranked (but still on the first page) | 925 |
| Worse ranked (was not on the first page) | 2760 |
| Worse ranked (no longer on the first page) | 2049 |

approach to consider would be to normalise the raw scores with the most relevant score of the hit set: the most relevant document will get the score of 1. In this way, the first documents of two different queries would be normalised with the same score.

Finally, as discussed in section 5.5.1, the number of displays might not be a good quality indicator, but more a general trend on all the database. However, instead of normalising the coefficients at a percentile level, they could be normalised over their number of displays. In this case, a record downloaded 5 times out of 10 displays will have the same normalized as a document downloaded 50 times out of 100 displays. Again, it is subjective: which one should be favoured? The first normalization will favour the second record, but the other would put both records at the same level. But the advantage of the second one will naturally reduce the disparities in the distribution of the scores.

### 5.7.2 *latest first* as default sorting

The queries used for training and counting were sorted by *latest first*: the more recent documents were at the top of the displayed records. As a result, the number of seen of the records was important: the documents of interest were not necessary at the top. In addition to that, the `queryanalyser` overestimates the number of seens.

Furthermore, it is difficult to know why a user clicked on a document, displayed by *latest first*: was it because the document is relevant, or is it because it is displayed first? Would the user have clicked on it if it was sorted differently and displayed a few pages after?

This default sorting method also explains why the weight for word similarity is so low in comparison to the others. *Latest first* sorting does not take into account word similarity, but put all emphasis on the quality and especially on the seen count(based on the large coefficient for seen). It would

have been better to take into account only ranked queries (ranked by word similarity) and to do the counts and estimation on these data. But with few queries, the counts would not have been relevant (only 16000 ranked queries over the 9 months). Another problem raises: how can we compare a seen count for a document coming from a ranked query and a seen count coming from a *latest first* display?

In the end, maybe only the download and the view counts should be taken as quality scores at the beginning, and the seen counts be used only from D-Rank queries as a negative parameter coming from the same method we would like to optimize.

# Conclusion

In this dissertation, we studied, adapted and used *distributed-ranking* in the frame of ranking documents in Invenio, the CERN software to manage an online digital library. This ranking method aggregates different ranking scores to produce a new one, which reflects the contribution of the individual scores all together.

Among the attributes to aggregate, we used the frequency access of a document, considering that documents which have already been accessed should be favoured in later queries. These attributes can come in addition to standard attributes such as word similarity.

The aggregation relying on logistic regression has been optimized by using the user behaviour at search time on the repository: his/her actions were taken into account so that already accessed documents could be better ranked in future queries. We used five months of user actions to estimate the optimal coefficients and tested them on the following months: we compared the original ranking of the documents that the user was looking for with the new ranking obtained with the optimal *distributed-ranking*. These coefficients provided a better ranking for 64% of the records and for 61% of the queries, the clicked records were on average better ranked.

The coefficients showed that the more a document is viewed or downloaded, the better it will be ranked later, and the more it has been seen, the lower ranked it will be. Word similarity (i.e. relevance to the query) is also contributing positively to the ranking, but surprisingly with a smaller weight.

This can be explained by the fact that the queries were displayed by *latest first* which gave lots of weight to seen counts. Moreover, the distribution of scores lead to a poor normalization of the attributes, which could have underestimated the importance of word similarity.

Further developments could consider improving the score normalization and also a better filtering of the data in order to work with relevant queries. Another development is the aggregation of mature records with fresh records, which do not have enough quality counts to be ranked with the others.

# Bibliography


[1] CERN, "Cern in a nutshell," http://public.web.cern.ch/public/en/About/About-en.html, Retrieved on April 27th, 2012.

[2] CERN, "History highlights," http://public.web.cern.ch/public/en/About/History-en.html, Retrieved on April 27th, 2012.

[3] CERN, "Facility," http://isolde.web.cern.ch/isolde/default2.php?index=index/facilityindex.htm&main=facility/facility.php, Retrieved on August 12th, 2012.

[4] CERN, "Atlas," http://public.web.cern.ch/public/en/LHC/ATLAS-en.html, Retrieved on August 12th, 2012.

[5] CERN, "Cms – compact muon solenoid," http://public.web.cern.ch/public/en/LHC/CMS-en.html, Retrieved on August 12th, 2012.

[6] CERN, "A nobel discovery – hunting the heavyweights with ua1 and ua2," http://public.web.cern.ch/public/en/research/UA1_UA2-en.html, Retrieved on August 12th, 2012.

[7] Katarina Anthony, "Getting to grips with antihydrogen," http://cdsweb.cern.ch/record/1431851?ln=en, Retrieved on August 12th, 2012.

[8] CERN Press Release, "Cern experiments observe particle consistent with long-sought higgs boson," http://cdsweb.cern.ch/record/1459454?ln=en, Retrieved on August 12th, 2012.

[9] CERN Document Server, "Cern document server," http://cdsweb.cern.ch/, Retrieved on April 27th, 2012.

[10] Anne Gentil-Beccot, Salvatore Mele, Annette Holtkamp, Heath B. O'Connell, and Travis C. Brooks, "Information resources in high-energy physics: Surveying the present landscape and charting the future course," *CoRR*, vol. abs/0804.2701, 2008.

[11] Invenio, "About invenio," http://invenio-software.org/, Retrieved on April 27th, 2012.

[12] E. Chisholm and T.G. Kolda, "New term weighting formulas for the vector space method in information retrieval," *Research Report*, 1999.

[13] Invenio, "Word similarity/similar records methods," http://invenio-demo.cern.ch/help/hacking/bibrank-word-similarity, Retrieved on August 12th, 2012.

[14] Ludmila Marian, Jean-Yves Le Meur, Martin Rajman, and Martin Vesely, "Citation graph based ranking in invenio," *Lect. Notes Comput. Sci.*, vol. 6273, no. CERN-IT-2010-001. 10.1007/978-3-642-15464-5_25, pp. 236–247. 12 p, Sep 2010.

[15] Google Scholar, "About google scholar," http://scholar.google.com/intl/en/scholar/about.html, Retrieved on June 29th, 2012.

[16] M. Fernández, D. Vallet, and P. Castells, "Probabilistic score normalization for rank aggregation," *Advances in Information Retrieval*, pp. 553–556, 2006.

[17] A. Le Calvé and J. Savoy, "Database merging strategy based on logistic regression," *Information Processing & Management*, vol. 36, no. 3, pp. 341–359, 2000.

[18] Martin Vesely and Martin Rajman, "Rank aggregation in scientific publication databases based on logistic regression," Technical Report No. LIA-REPORT-2009-002 LIA-REPORT-2009-002, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), October 2009.

[19] M. Veselỳ, M. Rajman, J.Y. Le Meur, L. Marian, and J. Caffaro, "D-rank: a framework for score aggregation in specialized search," *3rd International Conference on Agents and Artificial Intelligence*, 2010.

[20] S.A. Czepiel, "Maximum likelihood estimation of logistic regression models: theory and implementation," 2002.

[21] Peter Amoako-Yirenkyi, "Distributed ranking in invenio – ranking by relevance and quality," https://indico.cern.ch/contributionDisplay.py?contribId=5&confId=103885, 2010.