Statistical Analysis to Establish the Importance of Information Retrieval Parameters

Julie Ayter, Adrian-Gabriel Chifu, Sébastien Dejean, Cecile Desclaux, Josiane Mothe

Statistical Analysis to Establish the Importance of Information Retrieval Parameters

Julie Ayter
(Institut National des Sciences Appliquées de Toulouse, France
ayter@etud.insa-toulouse.fr)

Adrian-Gabriel Chifu
(IRIT UMR5505, CNRS, Université de Toulouse, France
adrian.chifu@irit.fr)

Sebastien Déjean
(IMT UMR 5219, CNRS, Toulouse, France
sebastien.dejean@math.univ-toulouse.fr)

Cecile Desclaux
(Institut National des Sciences Appliquées de Toulouse, France
cdesclau@etud.insa-toulouse.fr)

Josiane Mothe
(IRIT UMR5505, CNRS, ESPE, Université de Toulouse, France
josiane.mothe@irit.fr)

Abstract: Search engines are based on models to index documents, match queries and documents and rank documents. Research in Information Retrieval (IR) aims at defining these models and their parameters in order to optimize the results. Using benchmark collections, it has been shown that there is not a best system configuration that works for any query, but rather that performance varies from one query to another. It would be interesting if a meta-system could decide which system configuration should process a new query by learning from the context of previousqueries. This paper reports a deep analysis considering more than 80,000 search engine configurations applied to 100 queries and the corresponding performance. The goal of the analysis is to identify which configuration responds best to a certain type of query. We considered two approaches to define query types: one is post-evaluation, based on query clustering according to the performance measured with Average Precision, while the second approach is pre-evaluation, using query features (including query difficulty predictors) to cluster queries. Globally, we identified two parameters that should be optimized: `retrieving_model` and `TrecQueryTags_process`. One could expect such results as these two parameters are major components of IR process. However our work results in two main conclusions: 1/ based on post-evaluation approach, we found that `retrieving_model` is the most influential parameter for easy queries while `TrecQueryTags_process` is for hard queries; 2/ for pre-evaluation, current query features do not allow to cluster queries to identify differences in the influential parameters.

Key Words: Information Retrieval, query difficulty, query clustering, IR system parameters, Random Forest

1 Introduction

Being able to find relevant information about a topic is becoming more and more important for everyone. Indeed, everyone needs to find information, however, due to the great number of existing documents, it is very likely that most of them will not answer accordingly, even though it is highly probable that relevant information exists.

This is why search engines are widely used tools when one needs to find accurate information. Indeed, they allow users to express their question, through a query submitted to a search engine, and retrieve a list of documents that might be of interest for this specific query. Therefore, IR can be defined as a "set of tools and techniques that enable to find documents containing information that is relevant to a need" [Radhouani, 2010]. This represents the context of our work.

IR is a very active research area that aims at proposing and evaluating the indexing, matching and ranking methods to retrieve relevant documents according to users' queries. Many models have been defined in the literature, most of them having various parameters. To evaluate a model, the IR community relies on international benchmark collections, such as the ones from the Text REtrieval Conference (TREC)[1], composed of sets of documents, topics, relevance judgments and performance measures. It has been shown that there is an important variability in terms of queries and systems for these collections. System or search engine variability refers to the fact that a best system does not exist. A single configuration of a search engine that would treat any query as an oracle does not exist; rather, specific search engines are better for specific queries. By the way, system variability has not been deeply studied. Are systems sensitive to query difficulty? Are they sensitive to other query characteristics? If it is the case, some systems would be specialized in some types of queries. For example, ambiguous queries would need a system that first disambiguates the query terms, particular queries would be improved if first expanded, etc.

Having in mind the system variability, our goal is to identify which search engine (or configuration) would respond best for a particular type of query. If we were able to characterize query types and learn the best system for each query type, then, for a newly submitted query, we would be able to decide which system should process it. This requires to thoroughly analyze systems, queries and obtained results, in order to find whether some system parameters are more influential on results and whether this influence is query dependent. In our context, the best system means which retrieval model to choose, whether the query should be expanded, or not and if the query should be expanded, in what manner should this be done. This is why we consider parameters regarding indexing, retrieval and query expansion all together.

[1] `http://trec.nist.gov/`

This paper reports a deep analysis considering more than 80,000 search engine configurations applied to 100 queries and the corresponding system performance. The related work is presented in Section 2. The data set is described in Section 3. Section 4 reports the analysis of the system parameters according to query types. Defining query types is not obvious. We considered two approaches: the first one is based on clustering queries according to the query difficulty; the second approach uses various query features to cluster queries. Section 5 concludes the paper.

This work represents an extension of the paper [Ayter et al., 2014] and the extension consists in the higher level of experimentation details, more discussions, the addition of explained figures and the introduction of a new section in which we discuss the characteristics of two queries with a particular behavior.

2 Related Work

Statistical analysis of search results. Some studies in the literature have been conducted using statistical analysis methods to observe search results over sets of queries. In [Banks et al., 1999] the author analyzed TREC results; their data set consisted in the average precision values (see Section 3) for systems, on a per query basis. Various data analysis methods were used to identify the existence of different patterns. However, the authors stated the results are inconclusive. The same type of data matrix was used in [Chrisment et al., 2005], where the authors employed hierarchical classifications and factor analysis to visualize correlations between systems and queries. Mizzaro and Robertson [Mizzaro, 2007] aimed at distinguishing good systems from bad systems by defining a minimal subset of queries obtained from data set analysis. The authors concluded that "easy" queries perform this task best. Dinçer [Dinçer, 2007] compared performances of various search strategies through the means of Principal Component Analysis. Compaoré et al. [Compaoré et al., 2011] employed, as we do, indexing parameters and search models to determine which parameters significantly affect search engine performance. However, the number of parameters they used was significantly lower.

System variability and query difficulty. System variability is a major issue when choosing the most fitted system for a certain query. Harman and Buckley [Harman and Buckley, 2004], [Harman and Buckley, 2009] underlined that understanding variability in results (system A working well on topic 1 but poorly on topic 2 and system B behaving in an opposite manner) is difficult because it is due to three types of factors: topic statement, relationship of the topic to the documents and system features. Query difficulty is closely linked to system variability. Some features have been shown to predict query difficulty; these predictors cover various aspects of the three previous factors: term ambiguity (the number

of WordNet senses for query terms [Mothe and Tanguy, 2005]), term discrimination (IDF [Jones, 1972]), document list homogeneity (NQC [Shtok et al., 2009]), and result lists divergence (QF [Zhou and Croft, 2007]).

Clustering queries. In this paper, we use query difficulty predictors as features for query clustering. We then learn a model to associate a query cluster to the relevant system; in this way any new query that can be associated with an existing cluster can be processed by the corresponding system. In [He and Ounis, 2003] query features were also used to cluster TREC topics. The best term-weighting schema in terms of precision/recall measures was then associated with each topic cluster. Different term weighting schemas were applied depending on query characteristics. On TREC Robust benchmark, this method improved results on poorly performing queries. Kurland [Kurland, 2009] considered query-specific clusters for re-ranking the retrieved documents. Clustering queries has also been studied in the web context, based on query logs [Wen et al., 2002]. Bigot et al. [Bigot et al., 2011] showed how to group queries by their difficulty, using TREC submission results for training. A new query for the system, according to its type, was then assigned to the right search engine. Mothe and Tanguy [Mothe and Tanguy, 2007] also showed that queries can be clustered according to their linguistic features and that these features are correlated with system performances. In addition, combining various predictors for a better correlation with query performance [Kurland et al., 2012], [Chifu, 2013] also suggested that query difficulty predictors values could represent features to differentiate query types.

3 Data set Presentation and Descriptive Analysis

The data we analyze in this paper has been built by using various search engine components and parameters on TREC ad hoc collections. To do so, we have employed the Terrier[2] platform [Ounis et al., 2006].

TREC provides various topics corresponding to users' needs (having three parts: title, descriptive and narrative). Queries for retrieval are constructed from these topics. TREC also provides the documents for the search and relevance judgments to evaluate the retrieval results; several performance measures can be computed. On the other hand, the Terrier environment provides several modules and parameters for indexing, retrieval and query expansion models.

The initial data set was built using about 80,000 different combinations of term weights, retrieval algorithms and parameters applied on 100 queries from the TREC ad hoc track (TREC7 and TREC8). The redundant parameters were filtered out and the final data set consisted in 8 parameters (1 for indexing, 2 for retrieval and 5 for query expansion), for which different values were chosen (see

[2] `http://terrier.org/`

Table 1). Subsequently, the results were evaluated using the Average Precision (AP) performance measure. AP is extensively used in IR to compare systems effectiveness [Voorhees and Harman, 1997]. Indeed, the average of APs across all the queries is the Mean Average Precision (MAP) and represents a global performance measure for a system. When considering a retrieved document list, the AP is defined as the average of precision calculated each time a relevant document is retrieved. AP takes its value between 0 and 1; the latter being the best. The AP for a query q is defined as:

$$AP\left(q\right) = \frac{\sum_{k=1}^{R}\left(P@k \times \mathbb{1}_{\{k\in\mathcal{D}_q\}}\right)}{|\mathcal{D}_q|}, \tag{1}$$

where $\mathcal{D}_q$ represents the set of relevant documents for the query q and $\mathbb{1}_{\{k\in\mathcal{D}_q\}}$ equals 1 if the k-th document from $\mathcal{D}_q$ is relevant and 0 otherwise. The trec_eval tool has been employed to compute AP for the experiments.

We also employ the Average Average Precision (AAP) [Mizzaro, 2007], which represents the average of the AP values for a single query over all systems. AAP for a query q is defined as follows:

$$AAP\left(q\right) = \frac{1}{m}\sum_{i=1}^{m} AP\left(q, s_i\right), \tag{2}$$

where m is the total number of systems and s_i (with $i = 1, \ldots, m$) represents a system from the system set.

Of course, when we discuss about query expansion the robustness is crucial (Robustness Index [Sakai et al., 2005]). However, when the AP is optimized per query and if the expansion is considered necessary, the robustness should not be harmed.

The analyzed data is made of 8,159,800 rows and 10 columns (8 parameters, 1 query, 1 AP).

3.1 Descriptive statistics

First of all, we wanted to see if some parameters were obviously better than others, as far as the AP is concerned. Figure 1 shows the boxplots of the AP for each value taken by each search parameter. The boxplots are very similar from one parameter value to another. For example, the median and the first and third quartiles of the AP are almost identical for the four stemming algorithms.

Regarding the retrieving algorithm, there are a few differences: boxes of the `DFRee` and `DirichletLM` models are a little thinner and lower than boxes of the other algorithms. It means that both their median and their variability are smaller than those of other algorithms. However, this difference is not strong enough to conclude that these two algorithms are worse than the others.

Table 1: The considered parameters for the analysis

Name	Explanation	Possible Values
stemmer	name of the stemming algorithm	Crop, FSS, PS, TRWPS
retrieving_model	name of the retrieving algorithm	BB2, BM25, DFRee, DirichletLM, HiemstraLM, InB2, InL2
TrecQueryTags_process	what topic fields is used to build the query	N T T,D T,D,N T,N
parameter_free_expansion	is the parameter for expansion model free?	TRUE, FALSE
trec_qe_model	model of query reformulation	Bo1, Bo2, Info, KL, KLCom, KLCor
expansion_documents	number of top documents to use for query reformulation	2, 5, 10, 20, 50, 100
expansion_terms	number of highest weighted terms to use	2, 5, 10, 15, 20
expansion_mindocuments	number of documents a term should appear in to be eligible for expansion	2, 5, 10, 20, 50

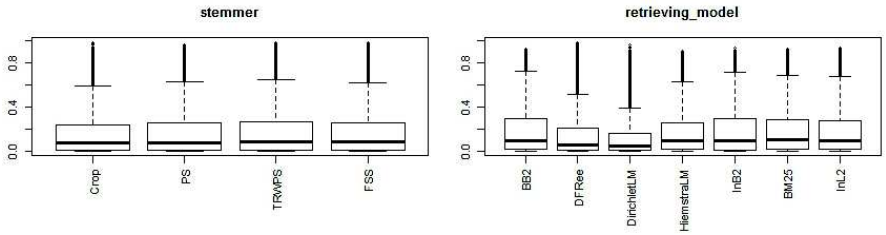

Figure 1: Boxplots of the AP for the first two parameters

We can also see that most of the good AP values are outliers beyond the top whisker of the boxplot, which means that few systems and queries give good results (less than 10% of the data with a specific value of a parameter). Consequently, it may be difficult to efficiently predict good AP results. Finally, we can notice that the first quartile of the boxplots is very close to 0 and most of the AP values are smaller than 0.4, which means there are many systems that give bad results and there might also be a lot of rows for which the AP is null.

3.2 Poorly performing queries and systems

There is a large number of low AP values (as seen in Section 3.1). If there are much more bad results than good ones, a predicting model would tend to predict bad results better than good results. This could represent a problem since we are more interested in good results. For this reason, we removed some from the data set in order to be able to focus on good results. There were 437,672 rows with a null MAP and almost 70,000 systems have at least one query with a null AP, thus we could not remove all of them from the data set without losing a lot of information. Another lead was to check whether any particular queries yielded worse AP values than others. We underlined that queries 442 and 432 are responsible for many more null AAP values than other queries representing 11.9% and 11% of the null data, meaning that these queries are much harder than the others.

We finally remove the parameter combinations that yielded really bad results, no matter what the query was. Doing so, by removing about 2.5% of the data, we removed more than 30% of the AP values that were equal to 0, while ensuring that the distribution of the results in the new data set was similar to the original one. We also notice that queries 442 and 432 remain the queries with the most null AP values, which represent about 16.9% and 15.7% of the rows with AAP equal to 0.

3.3 Analysis of queries 432 and 442

Since no major bias was introduced when we removed rows, we analyse the results of queries 442 ("heroic acts") and 432 ("profiling motorists police"). These queries are responsible for more than 30% of the null MAP measures, therefore it would be interesting to identify parameters that work better for these queries.

However, the AP measurements of these two queries are very low, regardless of the system: maximum AP for the query 442 is 0.0564 and is 0.0746 for 432 and a few systems reach AP over 0.05 for these queries. Thus, we raise the question of what can be considered as better results for these queries, since better results will be very low. We also try to find if there are parameters that give less bad AP values. We will say that a "good" AP value for these two queries is greater than 0.05. There are only 120 and 18 systems, respectively, for which the AP of queries 442 and 432 is greater than this threshold.

Subsequently, we looked at the values taken by each parameter in these 120 and 18 rows, to see if some of them were identical. The results of this study are presented in Table 2. Unfortunately, it is very rare to see the same values of the parameters between queries 442 and 432, so it is not possible to know how to set parameters in order to get less low AP.

Moreover, we can see that only two out of the seven retrieving model algorithms appear in this table: DirichletLM and DFRee. The same thing happens

Table 2: Summary of systems for which MAP$\geq$ 0.05 (Queries 442 and 432)

		Query 442	Query 432
stemmer	Crop	0	4
	PS	80	10
	FSS	40	0
	TRWPS	0	4
retrieving_model	DirichletLM	120	14
	DFRee	0	4
	other	0	0
Tags process	TITLE	120	0
	TITLE, DESC	0	18
	other	0	0
free expansion	TRUE	60	11
	FALSE	60	7
qe model	Bo1	32	4
	Bo2	32	14
	KL	18	0
	KLCp	18	0
	KLCt	20	0
	other	0	0

		Query 442	Query 432
expansion docs	2	60	0
	5	60	0
	10	0	8
	20	0	10
	other	0	0
expansion terms	2	120	1
	5	0	3
	10	0	7
	20	0	7
	other	0	0
min documents	2	16	8
	5	15	0
	10	29	0
	20	31	2
	50	29	8

with two out of the five values of `TrecQueryTags_process`. It could mean that these algorithms and fields give slightly better results on queries 432 and 442. However, we should carefully interpret these results because very few systems have been represented and we only focused on two queries, so we cannot know whether this is specific for these queries or it could be generalised to other queries.

4 System parameter tuning based on query performance

A single best system for all the queries is utopic, thus defining a model that could work for unseen queries implies to learn somehow from query characteristics. For this reason, we define clusters of queries for which we aim to associate a system with, making the hypothesis the most efficient systems would be different depending on the query cluster. Defining query clusters is not obvious. Two complementary approaches have been used to define query clusters. The first one is based on query difficulty (Section 4.1). In the second approach, we cluster queries according to query characteristics such as their length, their IDF, query difficulty predictors, etc. (Section 4.2).

4.1 Clustering queries according to their difficulty

We first consider a post-evaluation framework using the effectiveness of systems on each query. The queries are gathered in evenly proportioned clusters, according to their difficulty as measured by the AP: easy, medium and hard queries.

4.1.1 Analysis of variance

The first question we wanted to answer was whether all parameters have a significant effect on MAP. Indeed, there is a possibility that, for one or more of the eight search parameters, to have no significant difference in terms of MAP, when we vary the values of this parameter. In that case, it would mean that, whatever the value we give to the parameter, the MAP measure will stay almost the same. It would not directly help us finding the best system, but it could reduce the number of parameters to optimise. Moreover, some parameters could have an influence on easy queries and none on hard ones, for example.

In order to study this significance, we performed analysis of variance (ANOVA) testson the three query difficulty levels. Due to the size of the data set, we could not test all queries in a difficulty level, but instead we randomly selected two queries for each cluster.

In a first model, we considered there was only one effect: the parameter we wished to study:

$$\mathrm{AP}_{ij} = \mu + \mathrm{param}_i + \epsilon_{ij}, \quad \text{where } \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$$

param_i is the effect of the i^{th} value of the parameter on AP. This was done for each of the eight parameters.

In a second model, we considered all effects at the same time with no interaction:

$$\mathrm{AP}_i = \mu + \mathrm{param1}_{k_1(i)} + ... + \mathrm{param8}_{k_8(i)} + \epsilon_i, \quad \text{where } \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$k_j(i)$ is the value taken by the j^{th} parameter for the i^{th} system.

These two models were only applied to six randomly chosen queries with the following conclusion: all parameters seem to be significant, whatever the query difficulty is, except for the *free expansion* parameter (one non significant p-value). Therefore, all the parameters should be kept in the following analysis.

4.1.2 Classification and regression trees

We noticed in the previous section that all parameters have an influence on AP values, but ANOVA tests do not tell what values to give for these parameters toward optimizing the results. Indeed, if we could find a model that predicts well the data, we might be able to use it in order to optimize the parameters and to get the best AP as possible for any query.

The first statistical approach we used to model the data was Classification And Regression Trees (CART) [Breiman et al., 1984]. An advantage of this method is that, besides predicting results, we can see which are the best discriminating variables by plotting the trees.

Table 3: Definition of the good and bad results by difficulty

Difficulty	Bad	Average	Good
Easy	AP < 0.16	$0.16 \leq$ AP ≤ 0.5	0.5 ¡ AP
Medium	AP < 0.03	$0.03 \leq$ AP ≤ 0.18	0.18 ¡ AP
Hard	AP < 0.001	$0.001 \leq$ AP ≤ 0.04	0.04 ¡ AP

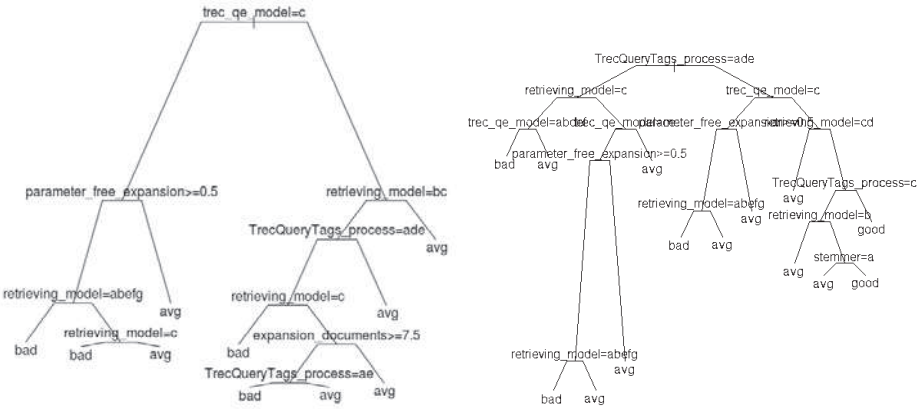

Figure 2: Classification tree: (a) for easy queries; (b) for hard queries

Here, we are not interested in knowing exactly what the AP value will be for one combination of parameters, we would rather like to know how good this value will be. Therefore, we need to define what a good or bad result is.

In order to do so, we looked at the first and third quartiles of the AP for each query group and decided that the results would be good when they are greater than the third quartile, bad when they are lower than the first quartile and average otherwise. The chosen limits of each level of satisfaction, for each difficulty level, can be found in Table 3.

Trees were computed for all systems and all queries of each difficulty level. In order to provide more interpretable trees, pruning was performed by setting a threshold on a complexity parameter (cp=0.001).

Easy queries. The parameter that best discriminates the AP results is the model of query reformulation (`trec_qe_model`) (see Figure 2 a.). Thus, it separates the systems with the Info algorithm (`trec_qe_model=c`, the letter `c` representing the third expansion model - Info) on the left from those with other algorithms on the right. However, we can see that no combination of parameters gives "good" results, we only get "bad" and "average" (avg) on the leaves of the tree. Maybe we could have had "good" results if we had pruned the tree later (using a smaller complexity parameter) but the model would have been too

complex and there might have been an overfitting. This is not very satisfying because it means that this method does not allow us to predict the best results and, therefore, to know what parameter values to use. We can see however that there are a few "bad" results, so we know which parameter values to avoid. For example, when the Info algorithm is used for query reformulation and the parameter for expansion model is free, it is very likely that results will be "bad".

Medium queries. The predictions are very similar to hard queries (see below), except for the stemmer algoritm which is not involved here for "good" predictions.

Hard queries. With regard to hard queries (Figure 2 b.), there are two "good" predictions which occur when using the fields (`TrecQueryTags_process`) TITLE or TITLE, DESC, the reformulation algorithms (`trec_qe_model`) Bo1, Bo2, KL, KLCom or KLCor and the retrieving model algorithms either BB2, DFRee, InB2, BM25 or InL2. Also the stemming algorithm has an importance in one of the predictions.

To conclude, we noted that results were not always satisfactory as they could not always predict "good" AP values. Nevertheless, it is more interesting to know which parameters to be careful about with hard queries than with easy ones. Indeed, for easy queries, results are supposed to be quite good and parameter values do not have a big influence, as opposed to hard queries.

Since CART tends to be quite unstable, random forests [Breiman, 2001] are employed to confirm the previous results.

4.1.3 Random Forests

In this aggregation method, the algorithm (*randomForest* function in R) randomly chooses a given number of variables (among all the variables of the data) for every split of classification trees. Many trees are computed in this manner and results are averaged in order to obtain an importance measure for each variable, based on the changes its presence or absence caused. We used this method on each of the three query difficulty clusters and we optimized the number of variables to select. For computational reasons we applied random forests to 10,000 randomly chosen training rows.

We used cross-validation and an accuracy measure to fix the number of variables to select. The accuracy was the greatest when there were four parameters in the trees, for every difficulty level. The accuracy decreases with the growth of the number of variables (see Figure 3).

We plotted the parameter importance graphs for each difficulty level, over the obtained random forests (500 trees in each forest). For instance, the search parameters that appear to be the most important to differentiate the goodness of AP results are: `retrieving_model` and `trec_qe_model` with an importance of 0.022 and 0.021, respectively. The other parameters have an importance of less

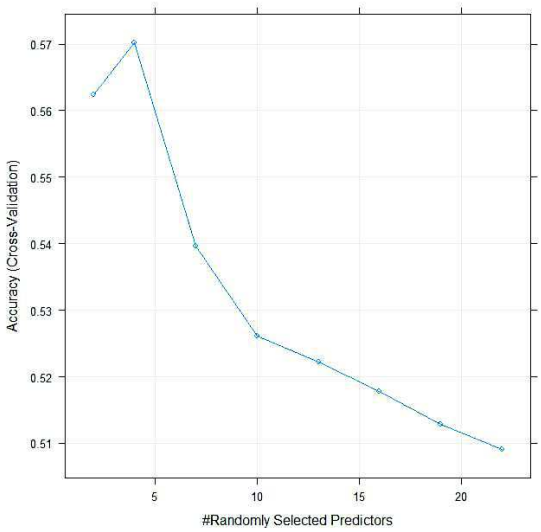

Figure 3: Accuracy for random forest (Easy queries)

than 0.007 (see Figure 4). For the medium queries the important parameters are `TrecQueryTags_process`, `retrieving_model` and `trec_qe_model`, respectively. In the case of hard queries the last two parameters interchange. First of all, we note that, even though they are a few differences between easy, medium and hard queries, two parameters are always among the three most important ones: the retrieving model and the reformulation model (`trec_qe_model`). We summarize the top 3 most important parameters, per difficulty level, in Table 4.

Table 4: Most important parameters by level of difficulty

	Easy	Medium	Hard
1st	retrieving_model	TrecQueryTags_process	TrecQueryTags_process
2nd	trec_qe_model	retrieving_model	trec_qe_model
3rd	expansion_documents	trec_qe_model	retrieving_model

Moreover, the obtained results are quite similar to the results when using CART. There are differences in the order of importance between CART and random forests, which can be explained by the fact that CART are unstable and depend on the training set. However, the results of random forests are presumably more reliable because of the average that is computed.

The results we obtained with queries divided into three groups of different

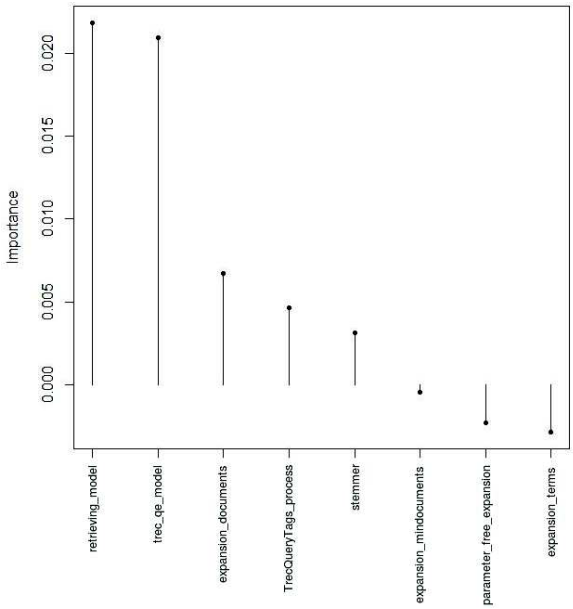

Figure 4: Importance of variables (Easy queries)

difficulty levels are quite encouraging. However, it would be interesting to have the same kind of information when queries are clustered into groups defined by other criteria. Indeed, the defined difficulty levels in our case depend entirely on the considered queries, since they were ordered and then separated into equal sized groups. Therefore, query characteristics were computed for every query (e.g. number of words) in order to define a new clustering.

4.2 Clustering queries hierarchically based on query features

In this section, we consider a pre-evaluation approach without any information about the system effectiveness but based on query features only. The characteristics we used to define new query clusters correspond to query difficulty predictors from the literature (pre and post retrieval), predictor combinations, statistics either on query terms, or on their representation in the indexed documents. This new data, after removing redundant features, consisted of 100 rows and 83 columns, corresponding to the 100 TREC queries and 82 characteristics (+ one column for the AP). The features are described in the following section.

4.2.1 Query features

We present here the query feature types employed in our clustering experiments. The features are heterogeneous by nature and cover various aspects of the query

and its terms, such as ambiguity, term popularity, syntax and morphology. Other features characterize the retrieved document lists for a system, regarding the score homogeneity or list divergence.

Morphological features. We used morphological query features as follows: the number of acronyms in the query, the number of terms with suffixes, the average number of morpheme, the average number of proper nouns, the average number of numerical values and the number of unknown tokens.

Syntactic features. Regarding syntactic features, we employed the average number of conjunctions, prepositions, personal pronouns in queries together with measures such as syntactic depth and syntactic distance between terms.

Term ambiguity. The *WordNet Senses* (WNS) [Mothe and Tanguy, 2005] measures the term ambiguity and it has been used as a pre-retrieval linguistic predictor. WNS is computed by the average of sense numbers in WordNet[3], for the query terms . We derive several features from this predictor, such as the maximum, the average and the sum of the number of senses for the query terms.

Term discrimination. The *Inverse Document Frequency* (*idf*) measures whether a term is rare or popular in a corpus of documents [Spärck Jones, 1972] and it has been used as a statistical pre-retrieval difficulty predictor. Its value for a query is represented by the average of *idf* for the query terms. We use here several variants: the minimum, the maximum, the average and the sum of *idf*.

Document list homogeneity. The *Standard Deviation* (STD) measures the degree of variation with respect to the average for the list of scores assigned to the retrieved documents, corresponding to a query and it has been employed as a post-retrieval statistical predictor. It is a variant of NQC [Shtok et al., 2009], without normalization.

Document lists divergence. *Query Feedback* [Zhou and Croft, 2007], models retrieval as a communication channel problem. It computes the overlap between two retrieval lists obtained by processing an initial query and the same query expanded by additional terms, respectively. Having the two retrieved lists, the overlap represents the number of documents that these lists have in common. A cut-off level x could be established for the document lists, that means the first x retrieved documents only are considered for the overlap. We consider here four cut-off levels: 5, 10, 50 and 100. It has been used in the literature as a post-retrieval predictor.

4.2.2 Clustering using PCA coordinates

To solve feature scaling issues, or low feature variability from one query to another, we performed a principal component analysis (PCA) on the data, using the *PCA* function of the `FactoMineR` package. In this manner, distances are

[3] http ://wordnet.princeton.edu/

computed on the query coordinates in this new base and they all have roughly the same value. Moreover, it is easier to represent clusters this way.

Then we look at the variable coordinates in the first three dimensions, viewing the correlation circles (Figure 5). The closer a variable is to the circle, the more correlated it is with the axes. Even if the plots are not easily readable, we can extract some relevant information detailed below. If we look at the horizontal axis on the Figure 5, which corresponds to the first dimension of the PCA, we can see that there are, on the left side, variables related to the query features `LOG` and `TF` (term frequency in the collection). On the right side, we can find characteristics such as the inverse document frequency (`IDF`) and the normalised inverse document frequency (`IDFN`).

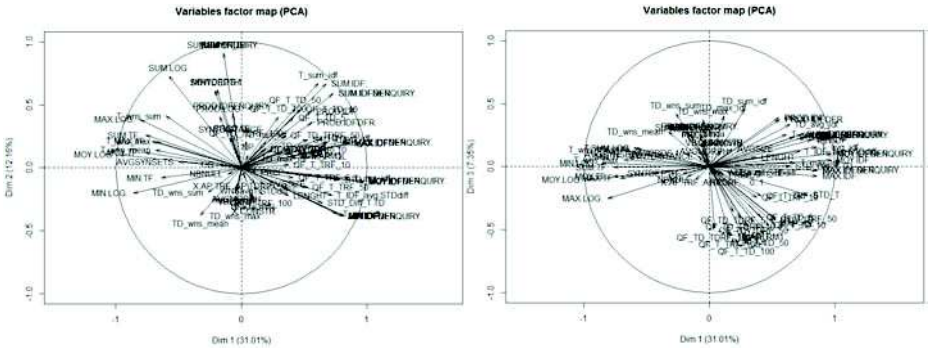

Figure 5: Variables in the three dimensions of the PCA: dim1/dim2 on the left, dim1/dim3 on the right.

Considering the second dimension (left side graph), there are, on the upper side, the characteristics `SUM(TF_Q)`, `SUM(LOG)`, `SUM(IDFENQUIRY)` and `NBWords`. On the bottom, characteristics concern the terms ambiguity and minimum inverse frequencies.

Finally, the third dimension (right side graph) differentiates characteristics that measure the similarity between original and extended queries on the bottom, from characteristics that measure ambiguity of the extended query on the upper side.

From the new coordinates, we compute the Euclidean distance matrix and the hierarchical clustering. We opted for Ward's for the agglomeration criterion: this method tends to give cluster of similar size (see Figure 6).

To choose how many clusters to keep, we plotted the height of the dendrogram branches against the number of clusters. Keeping 5 clusters seemed appropriate because it allowed us to have enough groups so that data sets would not be too large, but not too many so that there would be enough queries inside each group.

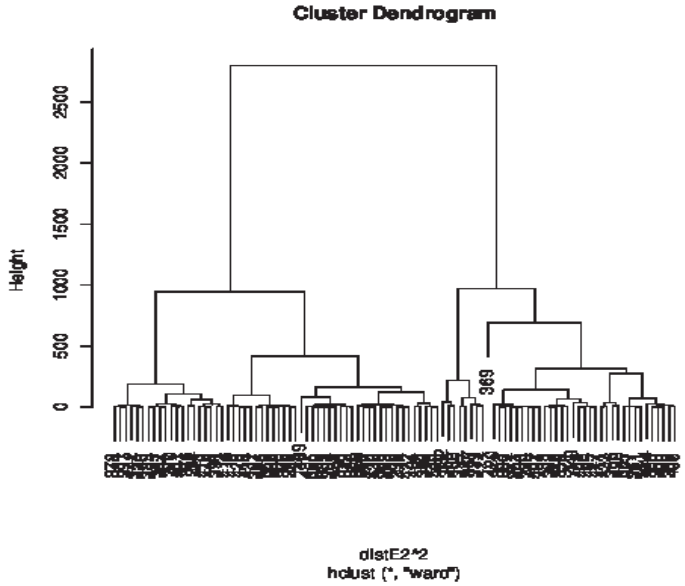

Figure 6: Dendrogram of the PCA coordinates (Ward's method)

It is possible to graphically represent the queries and their corresponding cluster using 5 different colors on the axes defined by the PCA (see Figure 7). We can see that clusters 1 to 3 (respectively black and green) are not very different when we consider their coordinates on the vertical axes.Cluster 2 is the one for which queries have the greatest `LOG` and `TF` whereas Cluster 1 is the one where queries have a great `IDF`.

The majority of queries are assigned to three clusters (containing 33, 20, and 38 queries respectively). The fourth and the fifth clusters contain only 8 queries and 1 query, respectively. Therefore these queries are not included in the statistical analysis that follows.

4.2.3 Classification and regression trees.

As in Section 4.1.2, we study CART on queries from the first 3 clusters as defined by the query features. We also defined goodness of AP values by using quartiles, in the same way as before (see Table 5 for the values).

We wanted to know what the best discriminating parameters were when considering query features rather than query difficulty; focusing on combinations of parameters that would give "good" results.

Cluster 1. Figure 8 shows that the parameters which best discriminate the goodness of MAP values are `TrecQueryTags_process` and `retrieving_model`.

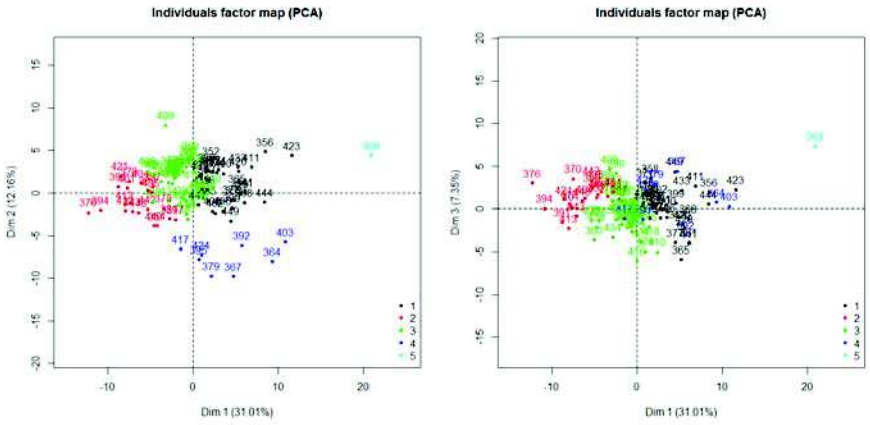

Figure 7: Five clusters in the PCA coordinates

Table 5: AP quartiles by cluster of queries based on query features

	Bad	Average	Good
Cluster 1	MAP ¡ 0.046	$0.046 \leq$ MAP ≤ 0.367	0.367 ¡ MAP
Cluster 2	MAP ¡ 0.002	$0.002 \leq$ MAP ≤ 0.082	0.082 ¡ MAP
Cluster 3	MAP ¡ 0.020	$0.020 \leq$ MAP ≤ 0.274	0.274 ¡ MAP

However, using a smaller complexity penalisation (cp=0.0005) did not lead to "good" results with this tree.

Cluster 2. `TrecQueryTags_process` and `retrieving_model` are most important variables for the second cluster. CART for cluster 2 were not able to predict "good" results either, even when we changed the penalisation.

Cluster 3. For the last cluster, the most important parameters are the same as the ones of the two other clusters.There were no "good" predictions with this cluster either.

The most important variables that we have observed with these trees are the same as those obtained when we considered the level of difficulty of the queries (see Section 4.1). This means that parameters `TrecQueryTags_process` and `retrieving_model` are indeed very important in the prediction of the AP values. However, with any of these trees, it was not possible to find the parameter combinations which yields "good" results. We changed the training set and the complexity penalisation ("cp") several times, but still without any "good" predictions. The cause might be the high number of "average" results in the data (twice as many as "good" or "bad" results), but in this way we would have had the same problem with the "bad" predictions and it is not the case here.

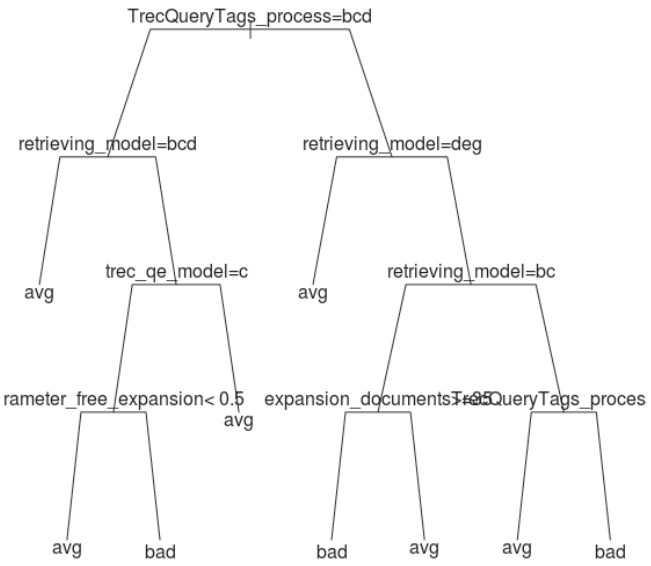

Figure 8: Classification tree for queries from Cluster 1

4.2.4 Random Forests

As previously, we use random forests and this method was applied on 10,000 random rows of each of the three query clusters defined by their features and 500 trees were computed. By plotting graphs of accuracy, we saw that the accuracy was the greatest when there were four variables in the trees, regardless of the cluster. We computed the parameter importance for MAP goodness prediction using the mean decrease in accuracy for each cluster. The results for the top 3 parameters per cluster are shown in Table 6.

Table 6: Most important parameters for the three clusters based on query features

	Cluster 1	Cluster 2	Cluster 3
1st	TrecQueryTags_process	retrieving_model	retrieving_model
2nd	retrieving_model	TrecQueryTags_process	TrecQueryTags_process
3rd	trec_qe_model	trec_qe_model	trec_qe_model

First of all, for all three clusters, the most important parameters are the same, however in different order (`retrieving_model`, `TrecQueryTags_process`

and `trec_qe_model`). Moreover, these parameters are basically the same as the ones we obtained with clusters based on query difficulty (see Section 4.1.3). Therefore, it seems the kind of clustering we use does not have a significant influence on what the most important search parameters are. Alternatively, it could also mean that it is not necessary to divide queries into clusters, in order to know what parameters are the most important.

From the 10,000 observations used in the random forest we randomly choose 7,000 as a test set to check the prediction quality of our model. The error rates obtained from the confusion matrices are high, especially with "good" and "bad" predictions (0.74-0.76).On the other hand, for the "average" predictions the error rates are around 0.50.

This difference between errors is due to the greater number of "average" AP results in the training set. Therefore, it is easier for the model to predict "avg" results. However, even for average AP results, the error rates are still very high. For this reason, it will not be possible to know for sure what parameter values should be used in order to get good AP values.

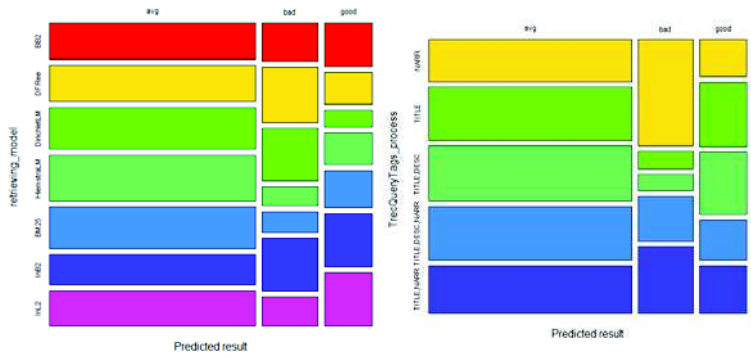

Figure 9: Mosaic plots for Cluster 1 crossed with `retrieving_model` and `TrecQueryTags_process`

To get a graphical idea of what values to choose, we use mosaic plots on the training set predictions. Therefore, for a given search parameter, we can compare the proportions of each value in every AP goodness category. If the proportion of a specific algorithm in the predictions "good" is much greater than its proportion in the "average" and "bad" predictions, it could mean that this algorithm gives better AP results, or at least it does so for the computed model. Mosaic plots are only reported with the two previous most important variables: `retrieving_model` and `TrecQueryTags_process` for each of the three clusters (see Figure 9 for the first cluster).

From Figure 9, one can notice that, in the "good" predictions of Cluster 1, there are more `InL2` algorithms than for the other levels of satisfaction. On the other hand, there are less `DirichletLM` algorithms than in "avg" and "bad" results (see the graph on the left). It could mean that `InL2` is an algorithm that works better for queries that belong to the first cluster and that it would be good to avoid the `DirichletLM` algorithm. Of course there are parameters and configurations that yield bad results no matter what query type is involved. However, it is still important to know which configuration to avoid, by query type, since some combinations are dangerous for certain query types and beneficial for others. Results are less obvious on the right side graph (`TrecQueryTags_process`). There might be a little bit more TITLE and TITLE,DESC in the predictions predictions "good" than in the others, but we could say for sure that there are many more NARR fields used in the "bad" predictions. Therefore, it is probably a good idea to avoid using this field in order not to get too bad AP values. Using the same kind of arguments, it is possible to identify a few algorithms, for each cluster, that seem to be more present in the predictions "good". These conclusions are summarized in Table 7.

Table 7: Parameters values giving "good" predictions and parameters to avoid

	Cluster 1	Cluster 2	Cluster 3
Parameters values giving "good" predictions			
retrieving_model	InL2	BB2 and InB2	BB2 and InB2
TrecQueryTags_process	T and T,D	T and T,D	T
Parameters to avoid			
retrieving_model	DirichletLM, DFRee	DirichletLM	DirichletLM, DFRee
TrecQueryTags_process	N and T,N	N and T,N	N and T,D,N

We can notice from Table 7 that parameter values were very similar between the three clusters. Hence, it would seem like, to get good AP results, the system should use the TITLE field (maybe also the DESC field) and avoid NARR. Moreover, retrieving algorithms `InL2` and `InB2` seem to give better results whereas it would be better to avoid `DirichletLM` algorithm.

The results are to be considered in the light of error rates which were relatively high, although they could represent a lead for new studies.

5 Conclusion

Search engines are tuned to perform well over queries. However, IR research has shown that there is not an optimal system that would answer the best on any

query. Rather, system variability has been shown. The objective of this work is to characterize query types and learn which system configuration is best for each query type. If we were able to do so, when an unseen query is submitted by the user, we would be able to decide which system configuration should process it.

This implies first to define query types, then to learn the best system for each query type. In this paper we defined two different types of query clustering. One is post-evaluation and based on the system effectiveness on each query measured by AP. The second one is pre-evaluation and based on query features (including query difficulty predictors). On each of these types, we analyzed which search parameters influence the most AP and tried to find what values these parameters should be.

From a massive data set of more than 80,000 system configurations, we found that two parameters (`retrieving_model` and `TrecQueryTags_process`) were frequently the most influential. One could expect such results as these two parameters are major components of IR process even if they are not the only ones; stemming and query expansion could also have been highlighted. However, when considering post-evaluation approach, we showed that `retrieving_model` is the most influential parameter for easy queries while `TrecQueryTags_process` is for hard queries. This result confirms the assumption that for hard queries, the system should manage to obtain more information on the user's intention, since the description part of a TREC topic capture the user's intention for his search. To make these results useful for real applications, we intend to draw same kind of conclusions on a pre-evaluation basis. Unfortunately, the current query features do not allow to cluster queries to identify differences in the influential parameters. Indeed, whatever the cluster determined on query features, the most influential parameters are the same. Nevertheless, some important trends have been identified for both pre- and post-evaluation approaches: it seems easier to determine the system configurations that lead to bad performances than to good ones.

Regarding our future work, this study highlights the need to define new query features which allow the system to determine the best configuration to deal with a query and thus implementing a selective IR process. Following another lead, this study has used the AP as the system performance measure for a query. This measure is used when global evaluation is made. However, some other measures are more task oriented. For example, high precision measures the precision when considering the top retrieved documents, while some other measures are more recall oriented. It would be interesting to analyze if when considering different goals (recall/precision or other) results differ.

Acknowledgments

We would like to acknowledge the French ANR agency for their support through the CAAS - Contextual Analysis and Adaptive Search project (ANR-10-CORD-001 01).

References

[Ayter et al., 2014] Ayter, J., Desclaux, C., Chifu, A., Mothe, J., and Déjean, S. (2014). Performance Analysis of Information Retrieval Systems (regular paper). In *Spanish Conference on Information Retrieval, Coruna, 19/06/2014-20/06/2014*, page (electronic medium). Springer-Verlag.

[Banks et al., 1999] Banks, D., Over, P., and Zhang, N.-F. (1999). Blind men and elephants: Six approaches to trec data. *Information Retrieval, Springer*, 1(1-2):7–34.

[Bigot et al., 2011] Bigot, A., Chrisment, C., Dkaki, T., Hubert, G., and Mothe, J. (2011). Fusing different information retrieval systems according to query-topics: a study based on correlation in information retrieval systems and trec topics. *Information Retrieval, Springer*, 14(6):617–648.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

[Chifu, 2013] Chifu, A. (2013). Prédire la difficulté des requêtes : la combinaison de mesures statistiques et sémantiques [Predicting query difficulty: combinations of statistical and semantic measures]. In *Proceedings of CORIA, Neuchatel, Switzerland*, pages 191–200.

[Chrisment et al., 2005] Chrisment, C., Dkaki, T., Mothe, J., Poulain, S., and Tanguy, L. (2005). Recherche d'information. analyse des résultats de différents systèmes réalisant la même tâche. *ISI*, 10(1):33–57.

[Compaoré et al., 2011] Compaoré, J., Déjean, S., Gueye, A. M., Mothe, J., and Randriamparany, J. (2011). Mining Information Retrieval Results: Significant IR parameters. In *Advances in Information Mining and Management*.

[Dinçer, 2007] Dinçer, B. T. (2007). Statistical principal components analysis for retrieval experiments: Research articles. *Journal of the Association for Information Science and Technology*, 58(4):560–574.

[Harman and Buckley, 2004] Harman, D. and Buckley, C. (2004). The NRRC Reliable Information Access (RIA) Workshop. In *Proc. 27th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 528–529.

[Harman and Buckley, 2009] Harman, D. and Buckley, C. (2009). Overview of the reliable information access workshop. *Information Retrieval, Springer*, 12(6):615–641.

[He and Ounis, 2003] He, B. and Ounis, I. (2003). University of glasgow at the robust track- A query-based model selection approach for the poorly-performing queries. In *Proc. Text REtrieval Conference*, pages 636–645.

[Jones, 1972] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

[Kurland, 2009] Kurland, O. (2009). Re-ranking search results using language models of query-specific clusters. *Information Retrieval, Springer*, 12(4):437–460.

[Kurland et al., 2012] Kurland, O., Shtok, A., Hummel, S., Raiber, F., Carmel, D., and Rom, O. (2012). Back to the roots: A probabilistic framework for query-performance prediction. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM '12)*, pages 823–832.

[Mizzaro, 2007] Mizzaro, S. (2007). Hits hits trec: exploring ir evaluation results with network analysis. In *Proc. 30th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 479–486.

[Mothe and Tanguy, 2005] Mothe, J. and Tanguy, L. (2005). Linguistic features to predict query difficulty. In *ACM SIGIR 2005 Workshop on Predicting Query Difficulty - Methods and Applications*.

[Mothe and Tanguy, 2007] Mothe, J. and Tanguy, L. (2007). Linguistic analysis of users' queries: Towards an adaptive information retrieval system. In *Signal-Image Technologies and Internet-Based System*, pages 77–84.

[Ounis et al., 2006] Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., and Lioma, C. (2006). Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*.

[Radhouani, 2010] Radhouani, S. (2010). Introduction à la recherche d'information. Available from: `http://fr.slideshare.net/radhouani/introduction-la-recherche-dinformation`. [accessed 12 January 2014].

[Sakai et al., 2005] Sakai, T., Manabe, T., and Koyama, M. (2005). Flexible pseudo-relevance feedback via selective sampling. *ACM Transactions on Asian Language Information Processing*, 4(2):111–135.

[Shtok et al., 2009] Shtok, A., Kurland, O., and Carmel, D. (2009). Predicting query performance by query-drift estimation. In *2nd International Conference on the Theory of Information Retrieval*, pages 305–312.

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

[Voorhees and Harman, 1997] Voorhees, E. M. and Harman, D. (1997). Overview of the sixth text retrieval conference (TREC-6). In *Proc. Text REtrieval Conference*, pages 1–24.

[Wen et al., 2002] Wen, R., J., Nie, Y., J., Zhang, and J., H. (2002). Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81.

[Zhou and Croft, 2007] Zhou, Y. and Croft, W. B. (2007). Query performance prediction in web search environments. In *Proc. 30th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 543–550.