Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 73 – 82

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

From a Ranking System to a Confidence Aware Semi-Automatic Classifier

Emmanuel Malherbe[a,b], Yves Vanrompay[c], Marie-Aude Aufaure[b]

[a]*Multiposting, 3 rue Moncey, 75009 Paris, France*
[b]*CentraleSupelec, Grande Voie des Vignes, 92290 Ch¢tenay-Malabry,France*
[c]*IMOB, Hasselt University, Wetenschapspark 5 bus 6, 3590 Diepenbeek, Belgium*

Abstract

Many systems rank outcomes before suggesting them to a user, such as Recommender Systems or Information Retrieval Algorithms. These systems require manual validation, which is time consuming and costly in industrial context. As it is the case in our industrial applications, we assume that the user's needs can be fulfilled by only one relevant outcome. We thus consider an algorithm that systematically selects the top ranked outcome. This approach requires to compute a correctness, estimating the confidence of the automatic decision, or equivalently how likely the first outcome of the ranking system is to be correct. Based on this estimation, we can apply a threshold on the correctness, above which no manual action is required; the system avoids human validation in many cases.

This paper proposes a novel method to estimate this correctness based on a supervised classification approach using the manual validations available in the base coupled with a representation of the system's scores. We conducted experiments on Multiposting real-world datasets generated by algorithms used in the industry; the first algorithm categorizes a job offer, the second recommends semantic equivalents for a given expression in a nomenclature. Our approach has thereby been evaluated and compared, and showed good results on our datasets, even with a limited training base. Moreover, in our experiments, for a given threshold, the better is the correctness estimation, the more performant is the semi-automatic system, showing that the correctness estimation leads thus to a crucial efficiency gain.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of KES International

Keywords: Recommender Systems; Information Retrieval; Recommendation Confidence; Semi-Automatic Classification; E-Recruitment;

1. Introduction

Multiposting is the French leader on e-recruitment solutions. Each year, millions of job offers are posted on recruitment websites, such as Monster.com, through Multiposting's interface. To handle such amount of data, the company is developing an automatic categorization of job offers on a high resolution nomenclature; at the moment, the algorithm suggests categories that need manual validation. Similarly, to help its clients in posting their job ads easily, the company needs to find semantic equivalences between e-recruitment websites nomenclatures. Once these hierarchies are matched, the client only needs to fill one form instead of dozens. The company employees are now matching items after items the nomenclatures, helped by a recommender system.

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of KES International
doi:10.1016/j.procs.2015.08.106

Both of these algorithms (categorization and schema mapping) rely on textual processing and data mining techniques, and are now subject to the following constraints:
- One outcome is required per query, and one is sufficient,
- When taking the top ranked suggestion as answer, the precision is too low for industrial use,
- Manual validation is systematically required (ranked suggestions displayed).

As the manual validation is time consuming, we would like to skip this step when possible. This implies to *estimate when the top suggestion is reliable*, and when it is not, taking it as the unique answer to the query only in the first case. As ranking systems are predicting real-valued scores to rank outcomes, we propose to consider these scores, coupled with a learning base, to estimate when the top ranked outcome is correct.

The paper is organized as follows: Section 2 surveys related research, while section 3 describes our problem and table 1 summarizes our notations. We dissect in section 4 variant approaches to estimate the correctness, and propose a method based on a specific representation of scores. Section 5 describes our algorithms deployed in real-world on which we applied our approach. Tests and comparisons in section Section 6 show the efficiency of the presented method and the industrial impact of our estimation. We finally conclude and discuss the obtained results in section 7.

2. Related Work

Information Retrieval systems and recommender systems are generally platforms suggesting outcomes to a user in response to a query[2,4], by ranking them. For these ranking systems, the computation focuses on the query (including user's preferences for recommendation) to suggest relevant outcomes. Generally, and in our experiments, outcomes and queries features are textual documents. We transform them using natural language processing (tokenization, stemming)[9] and represent as vectors using the vector space model with TF-IDF as term weighting function[8].

A problem we tackle is the industrial cost of manually validating the outcomes suggested by the system. By taking the most likely outcome as the result, our work falls into a classification task. To face the cost of correct or incorrect classification, cost-sensitive learning[25,24] proposes to consider this industrial constraint by balancing the classes weights in the training of the classification algorithm. However, the method remains limited to classification on given fixed classes, and requires as an input the cost matrix of our system.

Our approach to face this industrial cost is to estimate the confidence of our automatic decision. An important question we address in this paper is how to estimate the confidence of the most likely result of a ranking system. Many research has been done on estimating a probability from raw algorithms output, such as in[18] where the system is a single class SVM outputting an unbounded distance to the separating hyper-plane.[11,23] propose approaches for estimating probabilities from any multi-class classifier; this topic still remains an area of interest in the literature[16,10]. However, these studies only apply on classification on fixed classes, and need a large training set, including positive cases for every class.

Our work is also an approach for evaluating recommender systems. Indeed, we do not only evaluate the overall accuracy and coverage of our ranking system, but also derive metrics that indicate the confidence one can have in individual recommendations. In this sense we go further than existing evaluation approaches[2], while recent works suggest the importance of other metrics besides accuracy for evaluating the usefulness of recommendations[15].

3. The Notion of Correctness for a Ranking System

3.1. A Ranking System

The general problem is described by a query q and a set O_q of possible outcomes, a single outcome being denoted o. One notices that O_q and its size $|O_q|$ generally depend on the query, but this is not the case for every ranking system. Ranked outcomes are displayed to the user, and he can validate or not these outcomes, depending on the requirements.

We assume that the relevance of each outcome $o \in O_q$ with respect to the query q is independent to the relevance of the other outcomes $O_q \setminus \{o\}$. This independence assumption usually applies in information retrieval[20], but can be found in other algorithms such as recommender systems.

Outcomes are theoretically ranked with respect to $\mathbb{P}(o\ relevant|q)$. However, in practice, these probabilities are *represented* by a function $s(q, o) \in \mathbb{R}$, evaluated independently for every outcome o. This scoring function can be a direct estimation of the probability $\mathbb{P}(o\ relevant|q)$, but in general it doesn't have a probabilistic interpretation; it can indeed be unbounded, depending on the algorithm. The function s only depends on one outcome o due to the independence assumption. The system then ranks the outcomes with respect to $s(q, o)$ and display them to the user.

Depending on the system, the set of possible outcomes O_q can be fixed or not. In Information Retrieval, for instance, the outcomes are the retrieved documents, and might change regarding the query, when the user applies a filter on the documents for instance.

A first simple example for the scoring function s is the cosine similarity: $s(q, o) = cos(\vec{q}, \vec{o})$, where $\vec{q}$ and $\vec{o}$ are the representations of query q and outcome o, processed as textual documents to form vectors (see section 2). Many different algorithms exists for information retrieval, as[21]. Content-based recommender systems[2] are also ranking systems, where query q includes user's preferences. Multiclass classifiers based on one versus all approach[19] also compute scores, the query being the features vector and the outcome being the class. For more detailed examples, we refer to section 5 which describe two real-world ranking systems are provided in section 5.

To estimate which outcomes are relevant to the user, we suppose that a base Γ is available:

$$\Gamma = \{(q, O_q, O_q^r)\}$$

Where O_q is the set of suggested outcomes for query q, and $O_q^r \subset O_q$ is the set of relevant outcomes, manually validated.

Γ gives an idea of the user's feedback on system suggestions, and thus expresses which outcomes are really relevant. This base can be used to validate the computation of local scores $s(q, o)$, using for instance accuracy evaluation metrics[12]. This base can also be used as to improve the scores computation, by learning on Γ how to compute $s(q, o)$ in the best way possible. The validation metrics are then computed through a cross validation process.

From now on, we assume that the initial system is fixed, meaning that the computation of the scoring function s has been fixed, even if the function s itself can vary depending on the learning set in the case of an algorithm learned on Γ.

3.2. Correctness of the Top-Ranked Outcome

We now focus on the case where the user only needs one relevant outcome for each query, and systematically needs it. In other terms, once a relevant outcome is found, the other ones are useless. Some examples of such systems are detailed in the section 5. It is natural to focus on the most likely outcome, as being the potential unique relevant answer for a given query. We write $o^* \in O_q$ to denote the top ranked outcome for query q, that is to say the outcome with the highest $s(q, o)$ value:

$$o^* = argmax_{o \in O_q} s(q, o)$$

This leads us to propose a semi-automatic system, requiring human action only when the system is not confident enough about the relevancy of its first ranked suggestion o^*:

Algorithm 1: Semi-automatic System

```
input: $q$, $O_q$
foreach outcome $o$ in $O_q$ do compute $s(q, o)$
$o^* = argmax_{o \in O_q} s(q, o)$
if $o^*$ is likely enough to be correct then
    return $o^*$ as a unique final answer
else
    ask a manual selection, displaying ranked outcomes
end
```

The crucial point of this automatization is to determine how likely is o^* to be correct. We define the correctness C_q as the probability that the top ranked outcome is a correct answer:

$$C_q = \mathbb{P}(o^*\ relevant)$$

The correctness differs from the score $s(q, o^*)$ representing $\mathbb{P}(o^*\ relevant)$: first, C_q is a probability, contrary to the scores. Second, scores are computed with independence assumption, whereas C_q takes into account that o^* is the top ranked outcome of the query, and is implicitely aware of the other outcomes. For instance, even if all the scores $s(q, o)$ are very low but $s(q, o^*)$ remains much higher than other scores, it gives more chance to o^* to be a correct answer. On the other hand, if every outcome has a high score, but no outcome seems to stand out from the others, we are less confident about o^* being a correct answer.

Once the system can compute the correctness C_q, the condition o^* *is likely enough to be correct* can be changed to $C_q > t$, where the threshold t is choosen regarding the industrial strategy (see section 6). As our automation requires to compute the correctness, our problem is thus *to estimate the correctness C_q the most precisely as possible.*

4. Various Approaches for Correctness C_q Estimation

4.1. Heuristic-Based Approach

A first natural approach is to rely directly on the scores $s(q, o)$ predicted by the ranking algorithm. As we consider the case of a unique choice (the choice of o^*), we want to interpret the scores as the probability for each outcome to be this unique relevant one. To interpret $s(q, o)$ as a probability, we process the scores for a given query q:

$$s'(q, o) = \frac{s(q, o) - \epsilon}{\sum_{o \in O_q} (s(q, o) - \epsilon)}$$

Where ϵ is the lower bound of scores $s(q, o)$. As a lower bound is not necessarily available, ϵ can be arbitrarily chosen, and cut scores below it. We then have $s'(q, o) \in [0, 1]\ \forall\ o \in O_q$ and $\sum_{o \in O_q} s'(q, o) = 1$.

We use several metrics that are proposed in[22] to estimate the correctness:

- The Maximum; we simply consider highest normalized score: $\hat{C}_q = \max_{o \in O_q} s'(q, o) = s'(q, o^*)$

- Distance; we evaluate how much the best outcome stands out from the second one: $\hat{C}_q = \max_{o \in O_q} s'(q, o^*) - 2^{nd} \max_{o \in O} s'(q, o)$

These approaches only rely on the $s(q, o)$ with highest value for Maximum, and on the two highest ones for Distance. We note that more values $s(q, o)$ could be considered. Second, these estimations are static, as the computation doesn't rely on the manual validations O_q^r available in the base Γ. The following section thus proposes to estimate the correctness by learning an estimator on Γ.

Notations for Ranking System	
Symbol	Description
q	Query
o	Outcome
O_q	set of possible outcomes for query q
O_q^r	set of relevant (validated) outcomes, $\subset O_q$
$s(q, o)$	Scoring function, used for ranking

Notations for Correctness Estimation	
o^*	Top-ranked outcome
$\Phi^k(q)$	Top-k scores vector for query q
C_q	Correctness of the query q
$P_{q,o}$	Probability that o is correct for q
ψ	Learned function estimating $P_{q,o}$ from $s(q, o)$
φ	Learned function estimating C_q from $\Phi^k(q)$

Table 1. Notations.

4.2. Learning on Independent Scores

For this section, considering a query q, we focus on the score $s(q, o)$ for a given outcome o, not taking into account the scores for other outcomes of O_q. It is meaningful to separate every outcome making use the independence assumption (see section 3.1). We try thereby to estimate from any value $s(q, o)$ the probability that a user validates o, that is to say $\mathbb{P}(o\ relevant | s(q, o))$. This estimated probability $\hat{P}_{q,o}$ is computed through a predicting function ψ:

$$\hat{P}_{q,o} = \psi(s(q, o))$$

where the prediction function $\psi(x \in \mathbb{R}) \in [0, 1]$ is learned on the validation data-set Γ. Compared to multi-class probability estimation, this prediction is unaware of the class of o as there is not necessarily defined classes for the outcomes, and can thus be applied to a wider range of systems. The estimation is done independently for every outcome, making abstraction of the fact that they are from the same query. In that case, we express the base as a set of entries $(s(q, o), y)$, where $y = 1$ when o is relevant for the query q and $y = 0$ otherwise. The number of entries is much higher than for Γ, as there is one for each pair (q, o).

The estimated correctness of the query $\hat{C}_q$ is computed as:

$$\hat{C}_q = \hat{P}_{q,o^*} = \psi\big(s(q, o^*)\big) \tag{1}$$

This approach takes advantage of the base Γ, in order to learn when users are satisfied with the best outcome or not. We note that this estimation might be useful even when scores $s(q, o)$ are already computed as probabilities, because the score computation might be computed of the users' needs expressed by Γ. However, we note from equation 1 that the estimation of the correctness C_q only relies on the outcome with the highest value o^*. We can expect that scores for other outputs can give additional information on the quality of our recommendation: indeed, if the second highest local score $s(q, o)$ is much lower than the highest one, it means that first outcome stands out of the others. We thus extend this method by considering all scores in the next section.

4.3. *Learning on Top-k Scores*

For this part and the following, given a query q, we focus on all the corresponding scores $s(q, o)$, at the number of $|O_q|$. We thereby try to estimate the correctness from all the scores $s(q, o)$ for $o \in O_q$:

$$C_q = \mathbb{P}(o^*\ relevant | s(q, o), o \in O_q)$$

For this approach, we will train a classification algorithm using all scores $s(q, o),\ o \in O_q$ as features. We notice that a dimension problem arises: the number of features $|O_q|$ is not fixed for all q. Moreover, even when this dimension is constant (i.e. a fixed number of outcomes), there might be a risk of over-fitting when learning the algorithm for high number of outcomes, as for job categorization in section 5 with high number of outcomes compared to base size $|\Gamma|$. For these two reasons, we build a vector from the scores $s(q, o)$ before applying a classification algorithm.

To build this vector, we want to avoid principal components analysis. It reduces the dimension, but can only be applied in case when outputs o_i belongs to fixed classes, whereas we regard a problem with generally not defined classes. Moreover, the final projection can reduce drastically one component such that it is not taken into account in the correctness estimation. We will thus only consider the ranked k highest values of $s(q, o)$, where $k \in \mathbb{N}$. One can link this process with the distance D computation, where the 2 highest components are considered. This process defines the top-k scores vector $\vec{\Phi}^k(q)$:

$$\vec{\Phi}^k(q) = \begin{pmatrix} max_{o \in O_q} s(q, o) \\ 2^{nd}\ max_{o \in O_q}\ s(q, o) \\ \vdots \\ k^{th}\ max_{o \in O_q}\ s(q, o) \end{pmatrix} \in \mathbb{R}^k$$

One notes that $\vec{\Phi}^k$ can be viewed as a *feature map*, mapping a query q in the feature space $\mathbb{R}^k$.

The correctness is then estimated from the formula below:

$$\hat{C}_q = \varphi(\vec{\Phi}^k(q)) \tag{2}$$

Where the prediction function $\varphi(\vec{X} \in \mathbb{R}^k) \in [0, 1]$ is learned on the validation data-set Γ, expressed as a set of entries $(\vec{\Phi}^k(q), y)$ where $y = 1$ when the top ranked outcome o^* is relevant for the query q, and $y = 0$ otherwise.

5. Two real-world Ranking Systems

We consider two real-world systems for our correctness estimation and automation. Their brief descriptions are more detailed in distinct papers ([6] and [7]). The technical characteristics are detailed in table 2.

5.1. *Categorization of Job Offers*

To standardize its data, and in particular its millions of job ads, Multiposting is developing a computer-assisted tool for categorizing job offers, by matching them with the concepts of an ontology. Job categories are represented by the

ROME ontology, an equivalent of O*Net[5] with 531 categories, provided by *Pole Emploi* (http://www.pole-emploi.fr.), the French national employment service. *Pole Emploi* provides job category descriptions, separated into textual fields: title, description, skills, tasks, place of work, typical job titles.

The algorithm takes a job offer -a query- and tries to match it with a category description - the outcomes. This is done by predicting a similarity $s(q, o)$ between offer q and category o. As the query is a *job offer*, it is separated into 4 textual fields (title, description, profile, company description); for each field $f = 1..4$ we extract a vector $\vec{q}_f$, following the process described in[9]. Thus a job offer is represented by a tuple of vectors defined as follows:

$$q = (\vec{q}_1, ..., \vec{q}_4)$$

Similarly, an outcome o represents a *job category*, separated into 14 vectorized textual fields: $o = (\vec{o}_1, ..., \vec{o}_{14})$.

The method proposed in[6] focuses on the field to field similarity $cos(\vec{q}_f, \vec{o}_{f'})$, computed from cosine similarity as defined in[8]. The idea behind this similarity is to independently compare a field to another. To consider a field represented by $\vec{q}_f$ might be meaningful when compared to $\vec{o}_{f'_1}$ but totally irrelevant when compared to $\vec{o}_{f'_2}$. Approaches weighting terms according to their fields - as proposed in[21] or in[6] - can't handle such situation. The predicted similarity score between q and o is then defined as:

$$s(q, o) = \sum_{f=1}^{4} \sum_{f'=1}^{14} \lambda_{f,f'} cos(\vec{q}_f, \vec{o}_{f'})$$

Where $\lambda \in M_{4,14}(\mathbb{R})$ are the similarities weights, that are learned on the validation base Γ. For this purpose, our experts have manually assigned categories to more than 1,300 job offers, helped by an open source information retrieval system (Solr[21]). A job offer can be assigned to several categories or none, even if in practice, we need one and only one.

5.2. *Semantic Mapping of Schemas*

To save Multiposting clients' time, employees need to semantically map a schema to another one, referred as the source and the target schemas. The schemas represent for instance company sectors, courses provided in a university or a website jobs classification. Each item of the source schema needs one and only one equivalent item in the target schema. The company has developed an interface for employees to complete this task, with source schema on the left, and target one on the right.

The system aims at ranking the possible mappings by semantic relevance, a problem referred as name-based schema mapping. It considers leaves of the source schema one by one; such a leaf is then conceptually a query q. Similarly, the target schema is conceptually the set of possible solutions, each target leaf being an outcome o. The algorithm computes a score for each possible mapping $s(q, o)$, evaluating the semantic equivalence between q and o. In practice, to represent a schema leaf, we consider only the textual label of the leaf and that of the parent, such as ("*Web Development*", "*IT Department*"). We extract then two vectors following the process described in[9] to form q, and define the similarity among leaves $f(q, q')$ as:

$$q = (\vec{q}_1, \vec{q}_2) \qquad f(q, q') = \frac{1}{2}(cos(\vec{q}_1, \vec{q}'_1) + cos(\vec{q}_2, \vec{q}'_2))$$

This definition ensures keeping information about the hierarchy and doesn't mix labels of different depth. In this system, O_q is the target schema, and changes at every schema mapping. We define a similarity $g(O_q, O'_q)$ that captures how the vocabularies of schemas O_q and O'_q are similar, thanks to a cosine similarity between bag of words vectors for these vocabularies.

The idea to compute the semantic similarity $s(q, o)$ is to express the base Γ as a case base $CB = \{(q, O_q, o)\}$, where we only keep the positive cases or semantic maps validated by the employees. The target schema O_q is implicitly precised by the query, and semantic mapping score $s(q, o)$ is then computed using:

$$s(q, o) = \max_{(q', O'_q, o') \in CB} f(q, q') \cdot f(o, o') \cdot g(O_q, O'_q)$$

This approach can be viewed as a case-based reasoning[14]. The definition of f ensure that $s(q,o) \in [0,1]$, but we don't expect any probabilistic interpretation of scores, because of the specificity of their computation. The systematic manual validation through the interface keeps enriching the case base CB.

	Ranking System	
	Categorization	Schema Mapping
$\|\Gamma\|$: Number of queries in data-set	1,338	203,990
Outcomes O_q	Job categories, *Fixed*	Schema items *Change at every query*
$\|O_q\|$: outcomes per query	531	3 to 1000
$\|O_q^r\|$: relevant outcomes per query	0 to 3	1
Number of pairs (q,o)	710,478	72,523,082
Range for $s(q,o)$	Unbounded	$[0,1]$
Accuracy@1 of initial ranking system	65%	49%

Table 2. Characteristic from our data-sets.

6. Experiments on the Systems

6.1. Method For Evaluation

For dynamic methods, our validation relies on a 5-folds cross validation: the algorithm estimating C_q is trained on a part of Γ and then used to predict correctness on the other part. We compared our methods with a random prediction, for which $\hat{C}_q = 0.5$.

The tricky part of this validation is to learn the C_q estimation from unbiased values $s(q,o)$. Generally, and it is the case in our real-world systems, the scoring function s is learned on the base Γ, and we must pay attention to the way we build the training set for $\hat{C}_q$, built from $s(q,o)$ values. When learning functions ψ and φ, every $s(q,o)$ training value needs to reflect the value that would be predicted for an unseen query q. Thus, at each fold of the cross-validation, we perform a second-level cross-validation on the training fold, to compute *unbiased* $s(q,o)$ values on the fold. The sub-level cross-validation is for learning and predicting s, while the main cross-validation is for learning and predicting ϕ or φ.

6.2. Learning functions for φ and ψ

The approaches described in sections 4.2 and 4.3 rely on functions ψ and φ. They are respectively trained on scores $s(q,o)$ and top-k scores $\vec{\Phi}^k(q)$ to estimate the correctness $\hat{C}_q$. We used the following algorithms in our experiments, as being classification algorithms whose output can be interpreted as a probability.

An algorithm showing good results in posterior probability estimation[13] is to fit a sigmoid on the vectors X. The output is $\mathbb{P}(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta.X)}}$, where $\beta_0 \in \mathbb{R}$ and β has the dimension of X.

A popular algorithm is K nearest neighbors[1] ($K = 20$ in our experiments). The idea is to consider the K closest vectors X_i in the base ($i = 1..K$), with respect to a distance $d(X,X_i)$. The output is $\mathbb{P}(Y=1|X) = \frac{1}{K}\sum_i^K \frac{Y_i}{d(X,X_i)}$

We will also consider the random forest[3], combining randomized decision trees (300 trees in our experiments). Each tree outputs a probability $P_{tree}(Y)$, and the forest averages them to output $\mathbb{P}(Y=1|X) = \frac{\sum_{trees} P_{tree}(Y=1|X)}{|\{trees\}|}$

6.3. Performance Metrics Used

To estimate the quality of our estimation $\hat{C}_q$, we compare it with the corresponding ideal values 1 when o^* is relevant, and 0 when it is not, leveraging the following performance metrics:

- The mean square error: $MSE = \frac{1}{|\Gamma|}\Bigg(\sum_{\substack{(q,O_q)\in\Gamma \\ o^*\ relevant}} (\hat{C}_q - 1)^2 + \sum_{\substack{(q,O_q)\in\Gamma \\ o^*\ irrelevant}} \hat{C}_q^2\Bigg)$

- The negative log-likelihood:$LL = -\frac{1}{|\Gamma|}\Bigg(\sum_{\substack{(q,O_q)\in\Gamma \\ o^*\ relevant}} log(\hat{C}_q) + \sum_{\substack{(q,O_q)\in\Gamma \\ o^*\ irrelevant}} log(1-\hat{C}_q)\Bigg)$

- The area under the ROC curve $ROC - AUC$, as defined in[17].

6.4. Experimental Results

6.4.1. Heuristic-Based Estimation

On both data-sets, $ROC - AUC$ values in table 3 show that heuristic approaches seem to detect relatively well the queries with a correct first outcome. Second, $ROC - AUC$ is higher for distance (equation 4.1): it confirms the relevancy to consider several scores. However, these metrics don't have a good probabilistic interpretation, as the log-likelihood and mean square error are even lower than for a random correctness, and would be thus inappropriate for visualization on an interface.

	ROC-AUC	LL	MSE
Random	0.50 ± 0.00	**0.69 ± 0.00**	**0.25 ± 0.00**
Maximum	0.73 ± 0.03	1.41 ± 0.14	0.49 ± 0.05
Distance	**0.75 ± 0.00**	2.20 ± 0.23	0.57 ± 0.06

ROC-AUC	LL	MSE
0.50 ± 0.00	**0.69 ± 0.00**	**0.25 ± 0.00**
0.70 ± 0.03	1.74 ± 0.35	0.36 ± 0.06
0.75 ± 0.02	3.46 ± 0.44	0.43 ± 0.08

Table 3. Performance of Heuristic-Based Estimation, for Job Categorization on the left and Schema Mapping on the right.

6.4.2. Estimation from Independent Scores

We evaluated the estimation from independent scores (table 4). For both tests, the log-likelihood and mean square error are lower than for previous metrics: we succeed in better estimating the probability. However, the $ROC - AUC$ remains quite low, suggesting not to consider the scores $s(q, o)$ independently for our problem. Moreover, the training is much longer as there is an entry for each pair q, o. As a consequence, a computer with 8 cores and 16GB of RAM couldn't perform the tests on the schema mapping database with prediction by random forest and nearest neighbors algorithms.

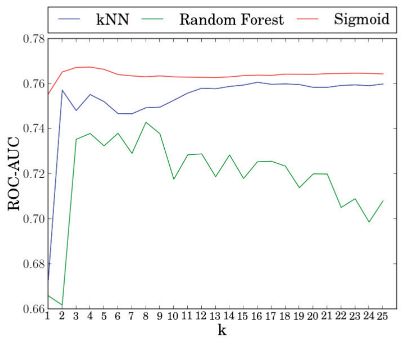
	ROC-AUC	LL	MSE
Neighbors	0.70 ± 0.03	4.02 ± 1.82	0.26 ± 0.02
Forest	0.70 ± 0.03	12.56 ± 3.67	0.28 ± 0.02
Sigmoid	**0.76 ± 0.02**	**0.72 ± 0.02**	**0.22 ± 0.00**

	ROC-AUC	LL	MSE
Sigmoid	0.65 ± 0.02	1.34 ± 0.14	0.35 ± 0.03

Table 4. Performance of Estimation learned on independent scores, for Job Categorization on the left and Schema Mapping on the right.

6.4.3. Estimation from Top-k Scores

First, we studied the value of k to consider for the top-k scores $\Phi^k(q)$. $ROC - AUC$ values when k varies (figure 1) suggest that $k = 4$ is sufficient for both data-set. For the schema mapping data-set, the 3 learning algorithms seems equivalent, while for the categorization (smallest data-set), the logistic regression shows better performance, showing nevertheless an over-fit for high values of k; Indeed, random forest and nearest neighbors generally need a large training set to work well.



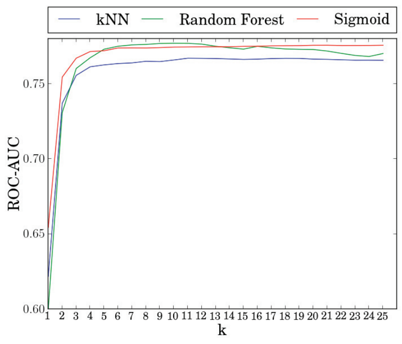


Figure 1. Performances of estimation with respect to k, on categorization data-set in the left and schema mapping data-set in the right.

Table 5 shows performances for $k = 4$. Firstly, it confirms that considering top-k scores is relevant to estimate the correctness, as results are better than for previous estimations. Secondly, using Sigmoid for φ seems to be appropriate for probability estimation: on one hand, it gives high $ROC - AUC$, one the other hand, it approaches relatively well the ideal probability (see log-likelihood). Last but not least, the training is fast with a computational time under one second and not memory costly, because it is performed on entries $\Phi^k(q) \in \mathbb{R}^4$ at the number of $|\Gamma|$ (Experiments done on a 2GB RAM and 2.1GHz computer). If our model gives results on just a thousand of queries (job categorization system), it can also be trained on a large data-set (schema mapping).

	ROC-AUC	LL	MSE	ROC-AUC	LL	MSE
Neighbors	0.76 ± 0.02	1.41 ± 0.56	0.19 ± 0.01	0.76 ± 0.01	1.74 ± 0.15	0.20 ± 0.01
Forest	0.74 ± 0.02	0.58 ± 0.03	**0.20 ± 0.02**	**0.77 ± 0.01**	0.72 ± 0.10	**0.19 ± 0.01**
Sigmoid	**0.77 ± 0.02**	**0.58 ± 0.01**	**0.20 ± 0.01**	**0.77 ± 0.02**	**0.57 ± 0.02**	**0.19 ± 0.01**

Table 5. Performance of Estimations learned on Top-k Scores for Job Categorization on the left and Schema Mapping on the right.

6.5. *Efficiency of the Semi Automatic System: choice of threshold*

We now consider the system that always selects the top ranked outcome as a result (algorithm 1), along with the estimated correctness. As described in table 6, we split the queries in two parts using a threshold on $\hat{C}_q$, automatically process the first one, and keep the initial ranking system for the second one. For a given threshold t on the correctness, let $Coverage_t$ be the proportion of queries of Γ such that $\hat{C}_q > t$ and $Precision_t$ the precision of the automatic system on this sub-part of queries. One notes that $Precision_t$ also corresponds to the accuracy at 1 of the ranking system on the first part of queries.

Ranking System	**Semi-Automatic System**	
For all queries: (100% queries)	Queries such that $\hat{C}_q > t$: ($Coverage_t$ % queries)	Queries such that $\hat{C}_q < t$: (1 − $Coverage_t$ % queries)
Manual validation, Visualization of $\hat{C}_q$	Automatic decision, $Precision_t$	Manual validation

Table 6. Description of a semi-automatic system compared to the basic one

To visualize the optimal threshold for such a system, we plotted the accuracy with respect to the coverage in figure 2. The curve is more regular with the estimation learned on top-k scores, which is crucial for industrial strategy. We note that the first half of the figures is more important - where the accuracy is higher. These curves show which correctness estimation yields to better accuracy for the hybrid system: on the categorization data-set (figure 2, left), for an objective of 90% of job correctly categorized, we can cover 31% of cases when C_q is estimated from top-k scores, against 19% for distance or 25% for estimation from independent scores. This gap is even deeper for schema mapping (figure 2, right); the best industrial strategy corresponds to the one provided by the nearest neighbors estimation from global score. With this estimation, for an objective of 80% of accuracy, we cover 34% of the mappings, while an estimation from independent scores can't provide this accuracy.

7. Conclusion and Future Work

We proposed a method to semi automate a ranking system, involving the estimation of the probability that the system suggests a correct first outcome; this estimation is based on top-k scores and fitting a sigmoid on a learning set. Experiments we conducted on 2 real-world data-sets yields to better results for our approach, compared to heuristic-based or independent score based estimations; moreover, our method doesn't require a large amount of data to learn an efficient estimation, compared to the requirements of literature multi-class probability estimates. The correctness estimation and automation can be applied to a wide range of ranking systems, as the computation is class independent and rely on few assumptions; our real-world systems are for instance a field to field job/category matching and a case base reasoning for schema mapping.

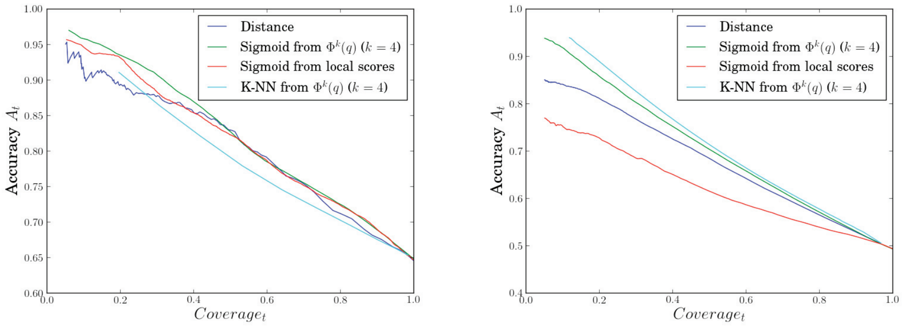


Figure 2. Coverage with respect to the Precision, on categorization data-set on the left and schema mapping data-set on the right.

Our next work will focus on improving the correctness estimation from additional features given by the ranking system. We remained in a general case by relying only on the scores used for ranking, but we hope for instance to better estimate correctness by taking into account the taxonomy of job categories in the job ads categorization.

References

1. N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46:175–185, 1992.
2. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.
3. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
4. P. R. Christopher D. Manning and H. Schtze. *Introduction to information retrieval*, volume 1. Cambridge: Cambridge university press, 2008.
5. E. C. Dierdorff, J. J. Norton, C. M. Gregory, D. Rivkin, and P. M. Lewis. O*net national perspective on the greening of the world of work. *Green Organizations: Driving Change with I-O Psychology*, pages 348–378, 2013.
6. E. M. et al. Field selection for job categorization and recommendation to social network users. *IEEE/ACM International Conference Advances in Social Networks Analysis and Mining (ASONAM)*, pages 588–595, 2014.
7. E. M. et al. A case-based approach for easing schema semantic mapping. *Submitted to International Conference on Case-Based Reasoning - ICCBR*, 2015.
8. G. S. et al. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11), 1975.
9. I. R. et al. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 2003.
10. J. K. et al. Probability estimation with machine learning methods for dichotomous and multicategory outcome. *Biometrical Journal*, 2014.
11. T.-F. W. et al. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 2004.
12. M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM.
13. D. W. Hosmer Jr and S. Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.
14. B. Lopez. Case-based reasoning: a concise introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2013.
15. S. M. McNee, J. Riedl, and J. Konstan. Accurate is not always good: How accuracy metrics have hurt recommender systems. In *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems (CHI 2006)*, 2006.
16. L. J. Nie, Qingfeng and S. Fei. Probability estimation for multi-class classification using adaboost. *Pattern Recognition*, 47(12), 2014.
17. Z. Omary and F. Mtenzi. Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. *International Journal for Infonomics (IJI)*, 3, Sept. 2010.
18. J. C. Platt. Probabilistic output for support vector machines and comparison to regularized likelihood methods. *Advances in large margin classifiers*, March 1999.
19. R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.
20. S. E. Robertson. The probability ranking principle in ir. *Journal of documentation*, 1977.
21. D. Smiley and E. Pugh. *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd., 2011.
22. Y. Vanrompay and Y. Berbers. A methodological approach to quality of future context for proactive smart systems. In S. D. Andreev, S. Balandin, and Y. Koucheryavy, editors, *NEW2AN*, volume 7469 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2012.
23. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 26(3):694–699, July 2002.
24. Z. Zhou. Cost-sensitive learning. *Modeling Decision for Artificial Intelligence*, 26(3):17–18, 2011.
25. Z. Zhou and X. Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.