A hybrid recommender system to predict online job offer performance

Julie Séguéla[*,**], Gilbert Saporta[*]

[*]Cédric-CNAM, 292 rue Saint-Martin, 75003 Paris, France
gilbert.saporta@cnam.fr
[**]Multiposting.fr, 23 rue d'Aumale, 75009 Paris, France
jseguela@multiposting.fr

Abstract. With the expansion of internet to advertise, the number of potential channels is increasing every day. In the Human Resource domain, recruiters have to choose between hundreds of job search web sites when they post a job offer on the internet. In order to save costs, assessing job board expected performance has become necessary. In this paper, three recommender systems providing job board performance estimation for a given job posting are introduced. This work refers principally to the new item problem, which is still a challenging topic in the literature. The first system (PLS-R) is a content-based approach, while others are hybrid recommendation approaches. Estimation is made on item neighborhood according to a "naive" similarity or a supervised similarity measure. These predictive algorithms are compared through experiments on a real dataset. In this application, supervised similarity-based system overcomes the lacks of other approaches and outperforms them.

1 Introduction

Context and objectives. In domains such as Marketing, Advertising or even Human Resources (sourcing), decisionmakers have to choose the most suitable channels according to their objectives when starting a campaign. With the expansion of internet to advertise, the number of potential channels (and targets) is exponentially growing. Today, a great challenge common to several research domains is the development of intelligent tools to support users in their choices. In this work, we focus on an issue encountered in the Human resource area. The rapid increase in the number of job search web sites (job boards) has made crucial the assessing of job posting performance. A job posting is a job offer published on the web, possibly on several job boards simultaneously. Performance is assumed to be the number of applications received on a job board. Posting a job offer on a job board is very expensive, so choosing wisely the channels to use is a very relevant task.

In this context, we are introducing a predictive algorithm of job posting performance (or return) on job boards. We resort to an innovative application of recommender systems: the goal of such systems is to help *users* to find *items* that they should appreciate from huge catalogues on the basis of previously attributed *ratings*. In our application:

– *items* are assumed to be job postings,
– *users* are assumed to be job boards,
– *ratings* are assumed to be observed returns of job postings on job boards.

Given the context, our system can be considered as a particular case of recommender systems commonly encountered in the literature (Adomavicius and Tuzhilin, 2005).

Data complexity and issues. Dataset is provided by *Multiposting.fr*, an online job posting solution which distributes jobs simultaneously on several dozens of job boards. We want to predict the number of applications received on the job board studied for a new job posting. Job postings are characterized by textual data (job description) and some structured data. These structured and unstructured data have to be handled simultaneously. Structured data are job characteristics (contract type, industry, education level required, etc.) represented by categorical variables and location characteristics represented by quantitative variables (population, unemployed people, etc.).

Thousands of features are extracted from job description texts thanks to information retrieval techniques. Since we have to deal with high dimensional data, job features have to be weighted in the predictive algorithm according to their power of explanation.

Section 2 is dedicated to a brief overview of recommender system literature and to the introduction of our system as a particular case (conditioned by the context and the type of our data). Then, approach is presented in section 3: handling of data is explained, and predictive algorithms are introduced (PLS regression and two versions of a hybrid recommender with a collaborative estimation). Experiments on our dataset and results are presented in section 4: different text representation techniques and three predictive algorithms are compared.

2 Recommender systems

2.1 Context

Today, companies are registering purchases, preferences and ratings of their customers or users. They want to make the most of those informations in order to provide personalized recommendations for the next purchases of their customers.

Recommender systems are involving three elements: *items*, which are exposed to *users*, who give them *ratings*. Data are represented in a matrix which rows and columns are users and items, and which elements are ratings. For example, users can be the customers of an on-demand film provider, items are movies, and ratings are integer values between 0 and 5 given by users for the movies they have seen. Purpose of the recommender system is to suggest to the user movies he has not seen yet and to which he is likely to give a high rating. In general, a ranking of items is provided by the system for a given user.

General issues are occuring in the literature. Most of the time, the total number of items is very high and each user has rated a small proportion of items, so the datatable is very sparse. Moreover, some items have very few or no ratings, and symmetrically, a new user has not given a rating yet. This issue is known as “cold-start” problem (related to new items or new users).

2.2 Brief overview of recommender systems

Recommender systems are usually classified into three main categories: content-based recommendations, collaborative recommendations, and hybrid approaches. In a content-based system, recommendations are generated from the features associated with items and the ratings that a user has given them, and the system recommends items similar to the ones the user liked in the past (e.g. Pazzani and Billsus, 1997; Mooney and Roy, 1999). Collaborative systems recommend to a given user items that people with similar preferences liked in the past (e.g. Shardanand and Maes, 1995; Konstan et al., 1997). Collaborative systems are based on user/item similarities (regarding to ratings) and expected ratings are computed as an aggregate of the ratings of most similar users/items. Hybrid approaches are combinations of collaborative and content-based methods, which helps to avoid limitations of each system (e.g. Balabanovic and Shoham, 1997; Schein et al., 2002).

To assess the capability of a system to provide good rating estimations, predicted ratings are usually compared to actual ratings thanks to the *mean absolute error* (MAE) or the *root mean squared error* (RMSE).

2.3 A particular case of recommender system

Our objective is to recommend job postings (items) to job boards (users) according to the expected return. In our case study, a new item is introduced and we want to suggest it to the users the most likely to appreciate it. The set of job boards is given but the perimeter of job postings is not limited. Indeed, each new recruiting campaign can be different of the previous ones. However, if a new job posting is identified as being the same as a previous posting, prior returns are used to improve the prediction. The dataset is made up of jobs posted on one or several job boards and which returns have been observed on each used channel. For each job posting, few channels are used, so few ratings are known. Consequently, the resulting matrix is very sparse.

Some of our issues are not usual in recommender system literature. Most of applications concerns very large datasets (thousands of users and items) where users have already rated several items according to their preferences. Our application concerns only about thirty users and our main case is the new item problem: ratings are estimated for items which have never been rated by users yet. In that case, only descriptive variables are used. It is not possible to obtain ratings for new items because each item is only recommended once to one or several users at the same time. Moreover, rating variability within and between users is high (see figure 1), instead of being limited to integer values between 0 and 5 (or 0 and 10) as usually.

3 Approach

An overview of our approach is presented in figure 2.

3.1 Data handling

As said previously, we have to handle simultaneously structured and unstructured data (textual data). Unstructured data are job description texts from which features are extracted

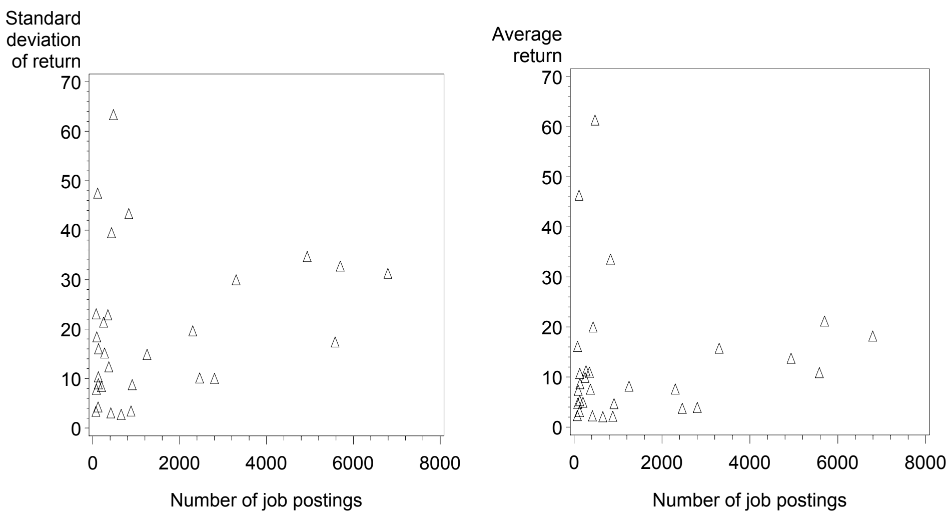

FIG. 1 – *Job boards represented in the maps (number of job postings, standard deviation of return) and (number of job postings, average return).*

thanks to an information retrieval technique, after a preprocessing of texts. These features will be used as input in learning approaches.

3.1.1 Preprocessing of texts

First, text preprocessing is made as following :

- Lemmatization and tagging, with algorithm proposed by Schmid (1994).
- Filtering according to grammatical category. Job offers are short texts (between 1000 and 3000 characters), compouned with short sentences and task listings, so we decide to keep only nouns, verbs and adjectives (as in Kessler, 2009).
- Filtering words occuring in less than five postings (a minimum number of occurrences is necessary to use terms in learning based methods).

The well-known Vector Space Model (also called “bag-of-word” representation; Salton et al., 1975) is used to obtain a vector of term frequencies for each job posting.

3.1.2 Feature extraction

To improve text representation we will use the TF-IDF (term frequency-inverse document frequency) measure (Salton, 1989). This common weighting technique decreases weights of keywords which appear in many documents and are not useful to discriminate documents. After weighting, documents will be indexed by Latent Semantic Analysis (LSA), a dimensionality reduction technique through a singular value decomposition of term-document matrices (Deerwester et al., 1990; Martin and Berry, 2007). The basic postulate is that there is an underlying latent semantic structure in word usage data that is partially hidden or obscured by the variability of word choice (synonymy problem). Correspondence Analysis (CA), widely

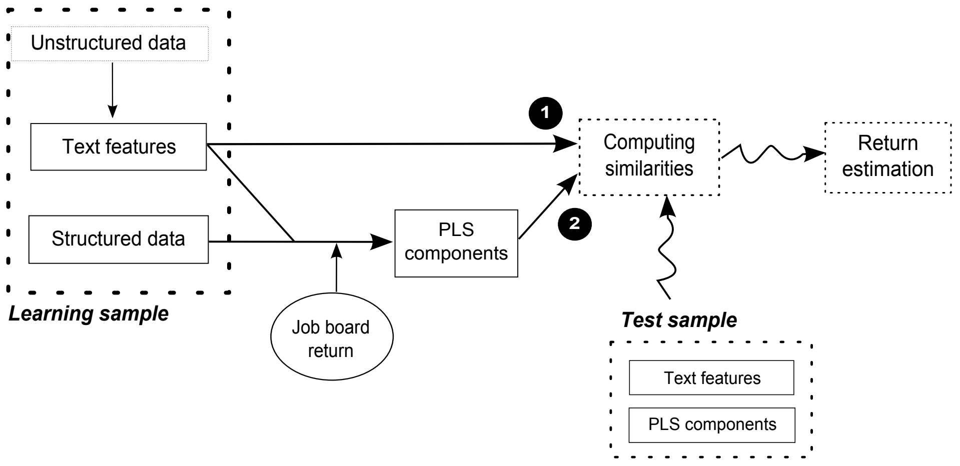

FIG. 2 – *Recommender system overview.*

used in statistical data analysis, strongly resembles to LSA when used in the context of textual data (Lebart et al., 1998). In a previous work (Séguéla and Saporta, 2011), we noticed the good performance of Correspondence Analysis in document representation task, so we will add this indexing technique to the list of experiments.

Other unsupervised methods, such as pLSA (Hofmann, 1999) and LDA (Blei et al., 2003), can be used to model semantics. However, these methods are not reported to generally outperform LSA (Gehler et al., 2006), so we will not try these techniques to avoid unnecessarily experiments. In section 4, feature indexing methods will be compared to underline their impact or absence of impact on recommender system performance.

3.2 PLS regression (PLS-R)

Introduced by Wold et al. (1983a,b) and first presented as an algorithm used for computing eigenvectors, Partial Least Squares regression (PLS-R) was rapidly interpreted in a statistical framework (e.g. Helland, 1990). PLS-R is a technique that generalizes and combines features from principal component analysis and multiple regression (for further details, see Abdi, 2010). This method is used when the number of predictors is large, and even large compared to the number of observations. The higher the number of predictors is, the higher the risk of important correlations between variables is. In case of high correlations between predictors, standard regression methods fail in estimating coefficients. As an alternative, PLS-R provides robust components (linear combination of predictors), independant and highly correlated with the dependent variable. In this work, we have to face to a large number of predictors (number of features extracted from job descriptions and structured variables) and high correlations, so PLS-R seems to be a very suitable technique.

3.2.1 Computing of PLS components

One part of PLS-R algorithm consists in computing non-correlated components, highly correlated with the dependent variable. The computing of components is based on NIPALS algorithm, first introduced by Wold (1966) for principal component analysis. Considering $(x_1, ..., x_p)$ a set of features, potentially highly correlated, and a dependent variable r, the first component is: $t_1 = w_{11}x_1 + \ldots + w_{1p}x_p$, where $w_{1j} = \frac{cov(x_j,r)}{\sqrt{\sum_{j=1}^{p} cov^2(x_j,r)}}$. Residual r_1 is obtained by regressing r on t_1. The second component t_2 is a linear combination of x_{1j}, residuals coming from the regressions of x_j on t_1: $t_2 = w_{21}x_{11} + \ldots + w_{2p}x_{1p}$, where $w_{2j} = \frac{cov(x_{1j},r_1)}{\sqrt{\sum_{j=1}^{p} cov^2(x_{1j},r_1)}}$. The regression of r on t_1 and t_2 gives the residual r_2. This iterative process continues until reaching H components, where H is determined by cross validation (Hoskuldsson, 1988). In PLS regression, r is then regressed on components: $r = c_1t_1 + c_2t_2 + \ldots + c_Ht_H + r_H$.

3.2.2 VIP indicator

To measure the importance of x_j variable to explain r through t_h components, we use VIP criterion (*Variable Importance in the Projection*):

$$VIP_{hj} = \sqrt{\frac{p}{\sum_{l=1}^{h} cor^2(y,t_l)} \sum_{l=1}^{h} cor^2(y,t_l) w_{lj}^2}$$

Since $\sum_{j=1}^{p} VIP_{hj}^2 = p$, the minimum threshold to keep a variable is often fixed to 1. We will use this rule to select variables used in the computing of PLS components.

3.3 A hybrid recommender system

In our context, items (job postings) are described by a set of features, extracted from their textual description and from structured variables (contract type, location, education level required, etc.). The description is a text on which we have applied information retrieval techniques mentioned in section 3.1.2. The following notations will be used in this section:
- $X = (x_1, \ldots, x_p)$ is the set of features describing items,
- $X_i = (x_{1i}, \ldots, x_{pi})$ is the vector of features describing item i,
- $r_{u,i}$ is the actual rating of item i for user u,
- $p_{u,i}$ is the predicted rating of item i for user u.

3.3.1 Computing of similarity between two job postings

Hybrid systems introduced in this paper are based on the computing of similarities between items, supposing that similar items have similar ratings for a given user. We propose to compute similarities in two different ways: a naive similarity between two sets of features, and a supervised similarity in order to identify nearest neighbors with respect to relevant features. Relevant features are those which are contributing in explaining job board returns. When a new item has to be rated for all users, his similarity with respect to all past items is computed.

Similarity functions. To compare two vectors of features extracted from texts, we first use cosine similarity measure (Baeza-Yates and Riberto-Neto, 1999) between items i_1 and i_2:

$$sim(i_1, i_2) = cos(X_{i_1}, X_{i_2}) = \frac{\sum_{j=1}^{p} x_{ji_1} x_{ji_2}}{\sqrt{\sum_{j=1}^{p} x_{ji_1}^2}\sqrt{\sum_{j=1}^{p} x_{ji_2}^2}}$$

Other similarity functions based on Euclidean distance d_{i_1,i_2} are used (the lower the Euclidean distance, the greater the similarity). As a particular case, we will test constant similarity measure. We use the same method as Shardanand (1994) to generate a third similarity measure:

$$sim(i_1, i_2) = \max_{i_k \in K}(d_{i_1,i_k}) - d_{i_1,i_2}$$

where K is the set of $|K|$ nearest neighbors of i_1 according to Euclidean distance.

Finally, we will test two additional similarity functions: gaussian and exponential functions decreasing with Euclidean distance. These functions are depending on a standard deviation parameter, for which we will try different values. Let σ_d be the standard deviation of distance between two items, gaussian and exponential similarity functions are respectively defined by:

$$sim(i_1, i_2) = exp\left\{-\frac{1}{2}\left(\frac{d_{i_1,i_2}}{\sigma_d}\right)^2\right\} \text{ and } sim(i_1, i_2) = exp\left\{-\frac{d_{i_1,i_2}}{\sigma_d}\right\}$$

"Naive" vs supervised similarity measure. Two different approaches are proposed. In the first approach, ratings are predicted with an heuristic technique. This approach assumes that similar items regarding to their features should have similar ratings for a given user. It is based on a "naive" similarity since it assumes that all item features have the same importance in the explaining of ratings. In order to not take into account non relevant features while searching item neighborhood, we propose a supervised similarity measure "rating-oriented" in a second approach. Instead of computing Euclidean distance on item features, it is computed on PLS components (see section 3.2.1), linear combinations of initial features extracted so as to explain as much as possible actual ratings.

3.3.2 Performance estimation

Ratings are estimated thanks to an aggregating function computed on item neighborhood (e.g. Adomavicius and Tuzhilin, 2005). Expected rating of user u for item i_1 is given by:

$$p_{u,i_1} = \frac{\sum_{i_{k'} \in K} sim(i_1, i_{k'}) \times r_{u,i_{k'}}}{\sum_{i_{k'} \in K} sim(i_1, i_{k'})}$$

where K is the set of $|K|$ nearest neighbors of i_1 with respect to the correspondent similarity measure. For constant similarity, which means computing average rating on i_1 neighborhood, nearest neighbors are determined by Euclidean distance.

3.3.3 Prediction improvement by *relevance feedback*

Sometimes, a new offer is in fact exactly the same as a previously posted one. An offer which has already been posted is called a *repost*. Indeed, some vacations are recurrent jobs and recruiters may need to post again an old job offer. When a repost is identified, we have an additional information: returns on job boards which have been chosen by the recruiter previous time (eventually several times).

We remind that $X_i = (x_{1i}, \dots, x_{pi})$ is the vector of features describing job posting i. We discretize job board return into l classes and code them through dummy variables. We create an enriched vector $\tilde{X}_i = (x_{1i}, \dots, x_{pi}, r^1_{u_1 i}, \dots, r^l_{u_1 i}, \dots, r^1_{u_N i}, \dots, r^l_{u_N i})$ which will be used to modelize and estimate job board returns in case of repost. In $\tilde{X}_i$ vector, r^l_{ui} is equal to 1 if the observed return of posting i on job board u belongs to class l.

3.3.4 System evaluation

Our recommender systems are systematically compared with the performance of the "average recommender" (AR). The average recommender provides recommendations based on job board average rating (computed on all past postings) and does not take job features into account. $p^{AR}_{u,i}$, rating predicted by average recommender for user u and all items, is defined by:

$$p^{AR}_u = \frac{\sum_{i \in D_u} r_{u,i}}{|D_u|}$$

where D_u is the set of items previously rated by user u.

To assess the capability of a system to provide good rating estimations, estimated values are usually compared to actual values thanks to MAE or RMSE. In our context, users have big differences in the number of rated items. This can bias our quality criterion because there is a high variability between average ratings of users (see figure 1). In this work, we will use $\overline{MAE}$ to compare system performances on the prediction task, which is the mean of users MAE:

$$\overline{MAE} = \frac{1}{N} \sum_{u \in U} \frac{\sum_{i \in D_u} |p_{u,i} - r_{u,i}|}{|D_u|}$$

where U is the set of users, $N = |U|$ the number of users, and $p_{u,i}$ the estimated rating for user u and item i.

As shown in figure 2, every learning step is made on a specific sample (learning sample), and predictive accuracy of algorithms is assessed on a test sample. Attribution is made randomly on the perimeter of job postings which are not reposts.

4 Experiments

4.1 Data description

Data are provided by *Multiposting.fr*, an online job posting distributor, and concern generalist and specialized job boards with an history of 80 postings at least. We decide to focus our attention on job boards with a mean return higher or equal to 2 because returns on other job

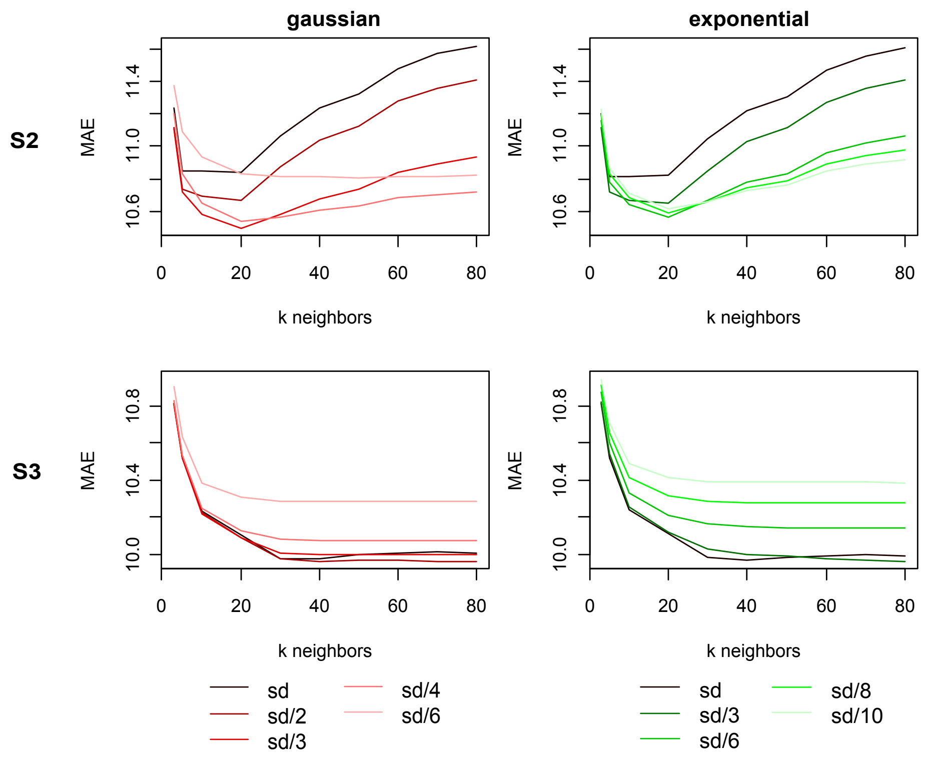

FIG. 3 – *$\overline{MAE}$ with $S2$ and $S3$ systems, according to standard deviation parameter (sd) in gaussian and exponential similarity functions (TF representation).*

boards are easy to predict and we don't want to take them into account when assessing system performance.

This kind of application is quite unusual in recommender system literature because we are studying a small dataset: few users (30 job boards), and about 42 000 ratings. We are studying about 18 000 items, each one has 3 ratings in average (a job is posted on 3 job boards of the dataset in average).

Reposts are identified and will be considered seperately. Remaining dataset is affected to learning and test samples, with a 60% and 40% distribution respectively. In experiments, repost sample will be treated like the test sample.

4.1.1 Experiments on text representation method

We first want to compare TF, LSA with TF-IDF weighting, and CA representations and to identify which one provides the best result (the lowest $\overline{MAE}$). To make this comparison, only textual features will be used in a first time. Different document representation techniques will be compared over three predictive algorithms: PLS-R ($S1$), hybrid recommender with naive similarity (or non-supervised recommender, $S2$) and hybrid recommender with supervised similarity (or semi-supervised recommender, $S3$). In $S2$ and $S3$ approaches, several similarity functions are compared (see section 3.3.1) and the optimal number of neighbors is studied. Cosine similarity is not relevant in S3 approach because the number of components kept is very small (less than 10 components) and sometimes equal to 1. Since $S2$ is a non-supervised system, the number of dimensions kept with LSA and CA can impact the recommender performance: preliminar experiments lead us to keep 50 dimensions for both methods.

PLS-R algorithm results are presented in table 1.

Approach	AR	PLS-R		
Representation method	None	TF	LSA (TF-IDF)	CA
$\overline{MAE}$	12.5	10.8	11.1	11.2

TAB. 1 – *PLS-R ($S1$) results.*

We first discuss the optimal standard deviation parameter in $S2$ and $S3$ approaches for gaussian and exponential similarity functions. The following sets of values are respectively used for gaussian and exponential function parameters: $\sigma_g \in \{\sigma_d, \frac{1}{2}\sigma_d, \frac{1}{3}\sigma_d, \frac{1}{4}\sigma_d, \frac{1}{6}\sigma_d\}$ and $\sigma_e \in \{\sigma_d, \frac{1}{3}\sigma_d, \frac{1}{6}\sigma_d, \frac{1}{8}\sigma_d, \frac{1}{10}\sigma_d\}$. Results are illustrated in figure 3, for TF representation (same “pattern” for LSA and CA representation methods). Best parameter values are reported in table 2.

Approach	$S2$			$S3$		
Representation method	TF	LSA	CA	TF	LSA	CA
Gaussian function	1/3	1/3	1/4	1, 1/2	1, 1/2	1, 1/2
Exponential function	1/6, 1/8	1/6, 1/8	1/6, 1/8	1, 1/3	1	1/3

TAB. 2 – *Best parameter value(s) ($\times\sigma_d$) for gaussian and exponential functions in $S2$ and $S3$ approaches.*

In $S2$ approach, lowest $\overline{MAE}$ for gaussian and exponential functions are reached respectively with $\sigma_g^{opt,S2} \in \{\frac{1}{3}\sigma_d, \frac{1}{4}\sigma_d\}$ and $\sigma_e^{opt,S2} \in \{\frac{1}{6}\sigma_d, \frac{1}{8}\sigma_d\}$. In $S3$ approach, decreasing initial value of standard deviation parameter does not allow to improve prediction accuracy.

We choose the best parameters in gaussian and exponential functions, and compare text representation methods for $S2$ and $S3$ approaches in figure 4. In the semi-supervised system, TF and LSA representations are equivalent whereas CA representation provides slightly lower quality. $\overline{MAE}$ is quite stable from 30 neighbors, except for estimation by mean on the neighborhood. In the non supervised system, all representation methods allow to reach a similar prediction quality, but LSA and CA methods are more stable with the number of neighbors. In the rest of the paper, we choose TF representation because it provides slightly better results for $S1$ and $S2$ approaches. However, we notice that LSA has allowed to reduce dimensionality and to preserve prediction accuracy.

The three approaches are finally compared in figure 5. We retain $max(d) - d$ similarity measure for the non-supervised hybrid system, and gaussian similarity for the semi-supervised hybrid system. The three approaches are better than average recommender, but the semi-supervised system allows to reach the lowest $\overline{MAE}$.

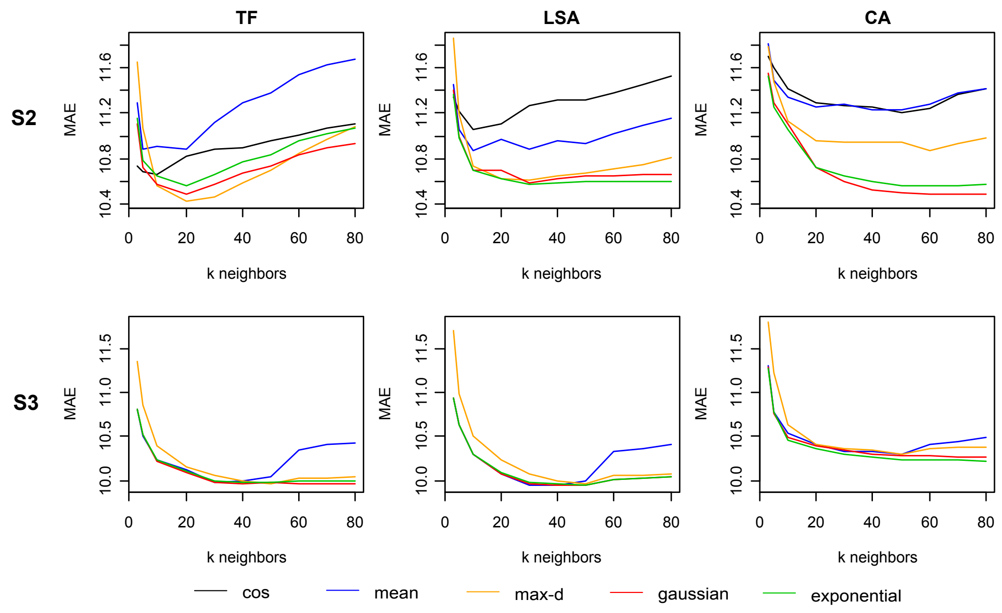

FIG. 4 – $\overline{MAE}$ *with* $S2$ *and* $S3$ *systems, according to the representation method and similarity measure.*

4.1.2 Improving job posting description

We now add structured variables to the set of descriptive variables. $S2$ approach is based on textual similarity and can't support the adding of qualitative and quantitative variables.

We lead the same experiments as previously to identify the best standard deviation parameters in gaussian and exponential similarity functions: $\sigma_g^{opt} = \frac{1}{3}\sigma_d$ and $\sigma_e^{opt} = \frac{1}{3}\sigma_d$. Results for $S1$ and $S2$ approaches with or without the adding of job descriptive variables are presented in figure 6. Curve pattern is similar but adding job posting characteristics has improved the quality of predictive algorithm.

4.1.3 Relevance feedback

We now consider the sample of reposted jobs. These jobs have already been posted on one or several job boards by the past, and our objective is to exploit previously observed returns to improve the quality of prediction on the sample of reposts. In relevance feedback (RF) approach, learning is made on all other postings described with the enriched vector (see section 3.3.3), including first postings associated with reposts. Approach without relevance feedback requires only textual features and other descriptive variables. As previously, enriched descriptive vector can only be used in supervised and semi-supervised approaches, $S1$ and $S3$. Figure 7 shows the comparison of results for the two approaches, with or without the use of relevance feedback. Thanks to relevance feedback, the prediction for a job repost will be more accurate ($\overline{MAE}$ is reduced by 1.5).

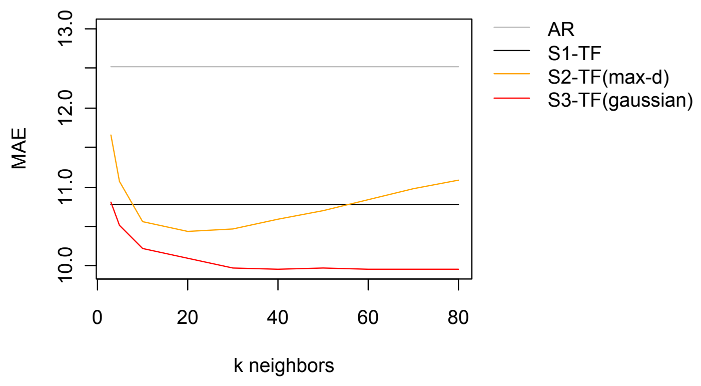

FIG. 5 – *Comparison of best algorithms.*

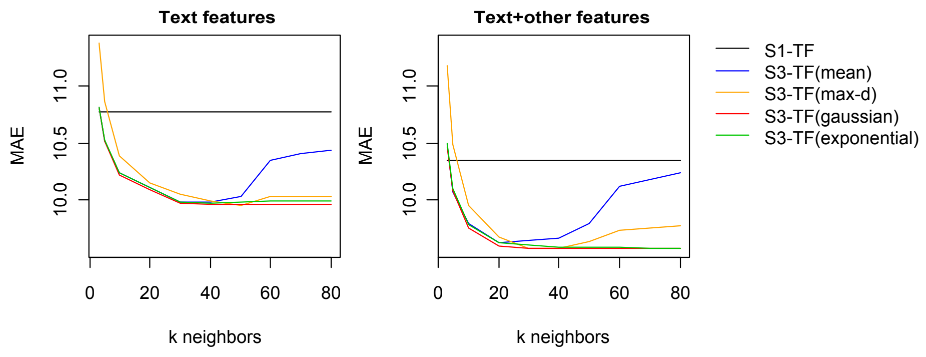

FIG. 6 – *Prediction improvement by adding descriptive variables.*

4.2 Discussion

Strengths and weaknesses of approaches proposed are summarized in table 3. PLS-R implies a linear relation between components and dependent variable, whereas other approaches are nonlinear with an estimation on the job posting neighborhood. In addition, the risk of overfitting is higher in PLS-R (because of the high number of input variables) than in other approaches. We appreciate the ability of $S1$ and $S3$ systems to provide interpretation tools of variable impact during the PLS modeling step. These systems also allow to give more important weigths to relevant features regarding to return estimation. On the contrary, $S2$ system does not allow neither explainability nor control on the weights of posting features.

5 Conclusions and future work

In this article, we have introduced three different approaches to predict job posting returns on job boards. We propose an innovative application of recommender systems by introducing

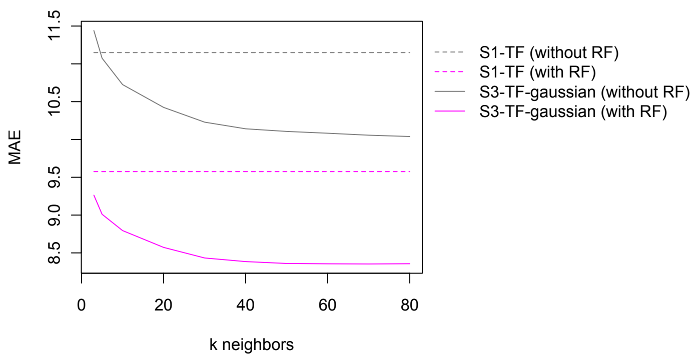

FIG. 7 – *Prediction improvement by relevance feedback.*

Recommendation approach	**Linearity constraint**	**Overfitting risk**	**Feature interpreting**	**Feature weight fitting**
PLS-R (S1)	yes	yes	yes	yes
Non-supervised hybrid (S2)	no	no	no	no
Semi-supervised hybrid (S3)	no	low	yes	yes

TAB. 3 – *Strengths and weaknesses of approaches.*

two hybrid systems adapted to the new item problem. In the main case, systems are only based on posting features (no use of rating knowledge). The semi-supervised hybrid system outperforms other approaches by combining the understanding of posting feature impact on job board return and the usage of collaborative knowledge. This system also allows the adding of descriptive data to improve prediction accuracy and ensure algorithm stability with the number of neighbors. In this approach, TF and LSI representation methods provide similar results, just as $max(d) - d$, gaussian and exponential similarity functions. In case of repost, learning on an enriched descriptive vector allows to improve algorithm efficiency.

Since our recommender system is dependent from a content-based approach, we need a sufficient number of postings on a job board to understand returns and give reliable recommendations. In a future work, we will extend this research to the case of new job boards (new users), with the introducing of job board features and the understanding of their impact on posting performance.

References

Abdi, H. (2010). Partial least square regression, projection on latent structure regression, PLS-regression. *Wiley Interdisciplinary Reviews: Computational Statistics 26*, 97–106.

Adomavicius, G. and A. Tuzhilin (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering 17(6)*, 734–749.

Baeza-Yates, R. and B. Riberto-Neto (1999). *Modern information retrieval.* New York: ACM Press.

Balabanovic, M. and Y. Shoham (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM 40(3)*, 66–72.

Blei, D. M., A. Ng, and M. I. Jordan (2003). Latent dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

Deerwester, S., S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 391–407.

Gehler, P., A. Holub, and M. Weilling (2006). The rate adapting poisson model for information retrieval and object recognition. In *ICML'06: Proceedings of the 23rd International Conference on Machine Learning*, pp. 337–344.

Helland, I. S. (1990). PLS regression and statistical models. *Scandinavian Journal of Statistics 17*, 97–114.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, pp. 50–57.

Hoskuldsson, A. (1988). PLS regression methods. *Journal of Chemometrics 2*, 211–228.

Kessler, R. (2009). *Traitement automatique d'informations appliqué aux ressources humaines*. Thèse de doctorat, Université d'Avignon et des Pays de Vaucluse.

Konstan, J. A., B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM 40(3)*, 77–87.

Lebart, L., A. Salem, and L. Berry (1998). *Exploring Textual Data*. Kluwer.

Martin, D. I. and M. W. Berry (2007). Mathematical foundations behind latent semantic analysis. In *Handbook of Latent Semantic Analysis*, pp. 35–55.

Mooney, R. J. and L. Roy (1999). Content-based book recommending using learning for text categorization. In *Proceedings of ACM SIGIR'99 Workshop Recommender Systems: Algorithms and Evaluation*.

Pazzani, M. and D. Billsus (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning 27*, 313–331.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley.

Salton, G., A. Wong, and C. S. Yang (1975). A vector space model for automatic indexing. *Communications of the ACM 18*, 613–620.

Schein, A. I., A. Popescul, L. H. Ungar, and D. Pennock (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference*.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pp. 44–49.

Séguéla, J. and G. Saporta (2011). A comparison between latent semantic analysis and correspondence analysis. In *CARME'11: International conference on Correspondence Analysis and Related Methods*.

Shardanand, U. (1994). Social information filtering for music recommendation. Master's thesis, Massachusetts Institute of Technology.

Shardanand, U. and P. Maes (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of Conference on Human Factors in Computing Systems*.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. In P. R. Krishnaiaah (Ed.), *Multivariate Analysis*, pp. 391–420.

Wold, S., C. Albano, W. J. D. III, K. Esbensen, S. Hellberg, E. Johansson, and H. Sjostrom (1983a). Pattern recognition: Finding and using regularities in multivariate data. In *Proceedings of IUFOST Conference: Food Research and Data Analysis*, pp. 147–188.

Wold, S., H. Martens, and H. Wold (1983b). The multivariate calibration problem in chemistry solved by the PLS method. In *Proceedings of the Conference on Matrix Pencils*, pp. 286–293.