Toward the Next Generation of Recruitment Tools: An Online Social Network-based Job Recommender System

Mamadou Diaby *†, Emmanuel Viennet *, Tristan Launay †
* Université Paris 13, Sorbonne Paris Cité, L2TI, F-93430, Villetaneuse, France
{mamadou.diaby, emmanuel.viennet}@univ-paris13.fr
† Work4, 3 Rue Moncey, 75009, Paris, France
tlaunay@work4labs.com

***Abstract*—This paper presents a content-based recommender system which proposes jobs to Facebook and LinkedIn users. A variant of this recommender system is currently used by Work4, a San Francisco-based software company that offers Facebook recruitment solutions.**

Work4 is the world leader in social recruitment technology; to use its applications, Facebook or LinkedIn users explicitly grant access to some parts of their data, and they are presented with the jobs whose descriptions are matching their profiles the most.

The profile of a user contains two types of data: interactions data (user's own data) and social connections data (user's friends data). Furthermore the users profiles and the description of jobs are divided into several parts called fields.

Our experiments suggest that to predict the users interests for jobs, using basic similarity measures together with their interactions data collected by Work4 can be improved upon.

The second part of this study presents a method to estimate the importance of each field of users and jobs in the task of job recommendation.

Finally, the third part is devoted to the use of a machine learning algorithm in order to improve the results obtained with similarity measures: we trained a linear SVM (Support Vector Machines). Our results show that using this supervised learning procedure increases the performance of our content-based recommender system.

Keywords— *Content-based Recommender System*, *Social Networks*, *Social Recruiting*, *Support Vector Machines*.

I. Introduction

The rapid growth of social networks in recent years has developed a new business: the trade of social networks users data. These social networks data are becoming important for many companies around the world and are often used to determine social networks users interests for items in order to propose or advertise items to them.

[1] defined social network sites as web-based services that allow users to construct profile (public or semi-public), to share connections with other users and to view and traverse lists of connections made by others in the system.

The personal information posted by users of a social network (which may involve personal description, posts, ratings, but also social links) can be exploited by a recommender system. [2] defined recommender systems as softwares that elicit the interests or preferences of individual consumers for products, either explicitly or implicitly, and make recommendations accordingly. Recommender systems have many pratical applications [3, 4] that help users deal with information overload, reason why they have become an active research field for two decades.

This paper presents an online social network-based recommender system that extracts users interests for jobs and then make recommendations to them accordingly. It is focused on two very popular social networks Facebook and LinkedIn, each counting millions of active users [5, 6]. A variant of this recommender system is used by Work4, the world leader in social recruitment technology. Due to privacy concerns the proposed recommender system only uses the data that social networks users explicitly granted access to.

The contributions of this paper are three-fold:

1) the first series of experiments on real world data from Work4 revealed that classic similarity measures used together with interactions data (see Section III-A) of social networks users give results that can be improved;
2) the importance of each field of users and jobs in the task of job recommendation has been estimated in the second series of experiments;
3) and finally, the last series of experiments results suggested that using supervised learning yields better results than similarity measures.

The rest of this paper is organized as follows: we summarized work done on recommender systems over the last two decades in section II, the sections III and IV present the proposed recommender system and the obtained results respectively and we sum up in section V

II. Related Work

Recommender systems help users deal with data overload by recommending to them items that they would like. There has been a lot of work done on designing recommender systems during the last two decades. Amazon.com [3] and Netflix [4] are two popular applications of recommender systems.

ASONAM'13, August 25-29, 2013, Niagara, Ontario, CAN

A. Recommender Systems

Recommender systems [7, 8] are mainly related to information retrieval [9, 10], machine learning [11, 12] and data mining [13]. [7] classified recommender systems into two main groups: rating-based systems and preference-based filtering techniques.

Rating-based recommender systems focus on predicting the absolute values of ratings that individual users would give to the unseen items. For instance, someone who rated the movies "Star Wars" $9/10$ and "The Matrix" $7/10$ would rate "X-Men Origins" $6/10$.

Preference-based filtering techniques predict the correct relative order of items for a given user. For instance, let assume the following preferences for a given user: iPad 3 $\succ$ Galaxy S III $\succ$ Galaxy tab 2; using the features of items and the opinions of other users, a preference-based system can predict that after iPhone 5 release, the user's new preferences would be: iPhone 5 $\succ$ iPad 3 $\succ$ Galaxy S III $\succ$ Galaxy tab 2.

This paper is focused on rating-based recommender systems, therefore the term *recommender system* refers to rating-based recommender system in the rest of this document.

Rating-based recommender systems are generally classified into three categories [7, 14]: content-based methods, collaborative filtering and hybrid approaches but [15] classified them into four categories: the above-cited three categories plus demographic filtering systems.

- Content-based recommender systems use either the ratings that users gave to items in the past or their associated description to define their preferences [16].
- Recommendations are made in demographic filtering systems using users personal information (age, gender, income, etc.): data provided during registrations or extracted from purchasing histories, survey responses, etc [15].
- In contrast to content-based systems, collaborative filtering approaches use the opinion of community of similar users to recommend a given item to a given user [8, 17, 18]. Conversely, rating predictions of an item involve known ratings of similar users.
- A hybrid recommender system combines a content-based system and a collaborative filtering technique into a single model. There are mainly four combinations [7]:
  1) aggregating the two recommendations of a content-based and collaborative filtering system [19];
  2) adding content-based characteristics to a collaborative filtering system: similarities between users or items are computed using content-based data [14];
  3) adding collaborative filtering to a content-based system: dimensionality reduction techniques on users profiles obtained by combining content and collaborative data to create a collaborative view [20];
  4) developing a single unifying recommendation model that uses both content-based and collaborative data [21].

B. Data representation

In recommender systems, textual description of a document (user or item) is mainly represented as a vector in which each component has a value that represents the importance of the associated term for the document. This vector is generally constructed using a weighting function and the "bag-of-words" model and by filtering out unimportant terms. The main assumption of the "bag-of-words" model is that the relative order of terms in a document has a minor importance in text categorization or classification tasks. A weighting function calculates the importance of a term for a given document. There are many ways to filter out unimportant terms:

- define a list of stopwords that will automatically be removed from the corpus;
- filter out the very highest and lowest frequencies words;
- another way is to filter out some grammatical categories.

[13] classified weighting functions into three main categories: local weighting functions, global weighting functions and the combinations of local and global weighting functions.

Local weighting functions only calculate the weight of a given term inside a given document. The boolean weight (Bool), normalized Term Frequency (TF) and Log Term Frequency (LTF) are the well-known methods; they are defined as follows:

$$\text{Bool}_{t,d} = \begin{cases} 1 & \text{if } t \in d \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\text{TF}_{t,d} = \frac{f_{t,d}}{\max_k f_{k,d}} \tag{2}$$

$$\text{LTF}_{t,d} = \log(1 + f_{t,d}) \tag{3}$$

where $f_{t,d}$ is the frequency of the term t in the document d.

Global weighting functions use the whole corpus (set of documents) to calculate the weight of a given term. The Inverse Document Frequency (IDF) and normalized Entropy (Entropy) which are two well-known global weighting functions, are defined as follows:

$$\text{IDF}_t = 1 + \log\left(\frac{N}{n_t}\right) \tag{4}$$

$$\text{Entropy}_t = 1 + \sum_d \frac{p_d^t \log(p_d^t)}{\log(N)} \tag{5}$$

where N is the total number of documents in the corpus, n_t is the number of documents that contain the term t, $p_d^t = \frac{f_{t,d}}{\sum_k f_{t,k}}$ is the probability that the term t belongs to the document d and $f_{t,d}$ is defined in (2).

In the literature, combinations of a local weighting function and a global weighting function give better results than local weighting functions [10]. TF-IDF and Log-Entropy are two well-known combinations and they are defined as follows:

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \text{IDF}_t \tag{6}$$

$$\text{Log-Entropy}_{t,d} = \text{LTF}_{t,d} \times \text{Entropy}_t \tag{7}$$

where t is a term and d is a document.

C. Similarity functions

Recommender systems use many various similarity functions to compute similarity between users, between items or between users and items: some similarity functions are heuristic and others are learnt models from underlying data using machine learning techniques. Two well-known similarity measures [7, 8] are cosine similarity and Pearson Correlation Coefficient (PCC).

Cosine similarity measures the cosine of the angle between two vectors; it is defined as follows:

$$\cos(u, v) = \frac{\sum_k u_k v_k}{\sqrt{\sum_k u_k^2}\sqrt{\sum_k v_k^2}} \tag{8}$$

It is mostly used in content-based recommender systems. As for PCC, it is mainly used in Collaborative Filtering techniques.

Classic similarity measures work on some specific problems but do not work on others: cosine similarity yields better results in item-item filtering systems [8] but in content-based recommender systems (see Section II-A), if the set of terms used by users is different from the set of terms used by jobs, the computed similarities between users and jobs using cosine similarity or PCC are almost always 0. In the literature, learnt similarity models from underlying data have been successfully used since they neatly fit the problems to be solved. Bayesian Networks [22] and SVMs [23] are two popular methods.

D. Performance measures

Many performance measures have been used to assess the performance of predictive systems (classifiers, recommender systems, etc.) [24]: precision, recall, F_β-measure.

The recall r_c and precision p_c for a class $c \in C$ are defined as follows:

$$r_c = \frac{n_c^*}{n_c} \tag{9}$$

$$p_c = \frac{n_c^*}{n_c'} \tag{10}$$

where C is the set of classes, n_c is the number of items in the class c, n_c' is the number of items affected to the class c and n_c^* is the number of items correctly affected to the class c.

The recall R and precision P for all the classes C are defined as follows:

$$R = \frac{\sum_{c \in C} r_c}{\mathrm{Card}(C)} \tag{11}$$

$$P = \frac{\sum_{c \in C} p_c}{\mathrm{Card}(C)} \tag{12}$$

F_β [25] is designed to take into account the recall and the precision; F_1 is the most often mentioned in the literature. F_β is defined as follows:

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{\beta^2 \times P + R} \tag{13}$$

Another popular performance measure is the AUC (Area Under the Curve) of a ROC (Receiver Operating Characteristic) curve [24] known as AUC-ROC. It is obtained by plotting the TP rate (fraction of true positives) as a function of FP rate (fraction of false positives).

III. A Social Network-based Job Recommender System

This section presents the proposed recommender system. It is divided into three main parts: the first part is about the modelling of documents (users and jobs) that we have used, the second part is dedicated to the proposed heuristic-based approach and finally the last part proposes a supervised learning method to improve our recommender system.

A. Document Modelling

The efficiency of a recommender system depends on how users and items have been represented. Social networks users are defined by two types of data: interactions data and social connections data.

- interactions data are both those that users post to the social networks and those recorded while users interacting with the system. Publications, comments, likes, time spent reading or viewing resources are some examples of interactions data.
- Social connections data are those from users social connections. List of friends can be cited as an example.

The proposed recommender systems (see Sections III-B and III-C) only use users interactions data and jobs descriptions to predict users interests for jobs.

Facebook users authorized the Work4 applications to access 5 fields data: Work history, Education history, Quote, About me, and Interests. LinkedIn users only authorized 3 fields: Headline, Education, Positions. LinkedIn Education and Positions fields are almost equivalent to Facebook Education history and Work history fields respectively. Work4 jobs descriptions have 3 fields: Title, Description, Responsibilities.

The vector that represents the profile of a given document (a user or job) is constructed as follows:

- in the basic method (see Section III-B1), texts from different fields of a document are concatenated, the concatenated texts are transformed into a vector using a weighting function. All fields are assumed to have the same importance on recommendation scores;
- in the generalized method (see Section III-B3), texts are concatenated taking into account the importance of the field from which they are. The transformation from concatenated texts to vectors is done using the same weighting function as in the basic method.

B. Heuristic-based method: cosine similarity

First of all, the well-known weighting functions (Bool (1), TF (2), LTF (3), TF-IDF (6) and Log-Entropy (7)) have been compared using cosine similarity (8). From the results that we obtained (see Table III), it is worth noting that TF, LTF and TF-IDF used together with cosine similarity give almost the

same results, hence TF-IDF (6) and cosine similarity (8) have been chosen as weighting function and similarity measure for the heuristic-based methods.

1) Basic heuristic-based method: The baseline method is simple: each user and job is represented as a vector using the "bag-of-words" method with TF-IDF (6) as weighting function (see Section III-A). Here, all users' and jobs' fields are assumed to have the same importance. The interest of a user for a given job is measured by computing the cosine similarity (8) of their vectors.

2) Importance of fields: Obviously, we know that not all the users or jobs fields have the same importance on the recommendation scores: some are more important than others. In order to determine the importance of users fields and jobs fields, the following optimization problem has been proposed.

The vector $u(\alpha)$ of a user u using the importance vector α of users fields is defined as the weighted sum of his fields vectors; formally it is defined as follows:

$$u(\alpha) = \sum_{f=1}^{f_u^0} \alpha_f^0 u_f^0 + \sum_{f=1}^{f_u^1} \alpha_f^1 u_f^1 \tag{14}$$

where $\alpha = (\alpha^0, \alpha^1)$, $\alpha^0 = (\alpha_1^0, \ldots, \alpha_{f_u^0}^0)$ and $\alpha^1 = (\alpha_1^1, \ldots, \alpha_{f_u^1}^1)$ are respectively the importance of Facebook and LinkedIn users fields, f_u^0 and f_u^1 are respectively the numbers of Facebook and LinkedIn users fields in the training set, u_f^0 and u_f^1 are respectively the vectors of the Facebook and LinkedIn field f for the user u and finally α_f^0 and α_f^1 are the importance of the Facebook and LinkedIn field f.

The vector $v(\beta)$ of a job v using the importance vector β of jobs fields is defined as the weighted sum of its fields vectors; formally it is defined as follows:

$$v(\beta) = \sum_{f=1}^{f_j} \beta_f v_f \tag{15}$$

where $\beta = (\beta_1, \ldots, \beta_{f_j})$, f_j is the number of jobs fields in the training set, v_f is the vector of the field f for the job v and β_f is the importance of the field f.

The predicted similarity $\hat{y}_{uv}(\alpha, \beta)$ between a user u and a job v using the importance of fields α and β is computed as the $\cos(u(\alpha), v(\beta))$ (see eq. (8)).

To learn the optimal α and β on the dataset Γ, we optimize the Weighted Mean Squared Error $E_\Gamma(\alpha, \beta, c_0, c_1)$ defined as follows:

$$E_\Gamma(\alpha, \beta, c_0, c_1) = \frac{1}{|\Gamma|} \sum_{(u,v,y)\in\Gamma} c_y \cdot (y - \hat{y}_{uv}(\alpha, \beta))^2 \tag{16}$$

Each entry of Γ is a 3-tuple (u, v, y) where u, v represent a user and a job respectively, $y \in \{0, 1\}$ is their associated label, c_0 and c_1 are the costs of the class 0 and the class 1 respectively.

For $\lambda_1 > 0$ and $\lambda_2 > 0$, we can notice that $\hat{y}_{uv}(\lambda_1\alpha, \lambda_2\beta) = \hat{y}_{uv}(\alpha, \beta)$, which means that if the optimization problem has a solution, it is not unique. Therefore we solved an constrained optimization problem to reduce the number of solutions; the constraints are: $\|\alpha^0\| = 1$, $\|\alpha^1\| = 1$ and $\|\beta\| = 1$.

3) Generalized method: The generalized recommender system is similar to the basic one (see Section III-A), the only difference is that it uses the importance of fields (see the results in the Section IV-B2) while aggregating the texts from different fields.

C. Supervised learning: SVM

After generalizing the heuristic-based recommender system (see section III-B3), we explored the use of trained statistical models: SVMs (Support Vector Machines). SVMs [26] are known to yield good performance in text categorization [23]. Using libSVM [27], we therefore apply this supervised learning procedure to our problem.

The input vectors I_{SVM} of the SVM are stated as follows:

$$I_{\text{SVM}}(u, v) = (u_1, \ldots, u_K, v_1, \ldots, v_K) \tag{17}$$

where K is the total number of terms in the dataset and u and v are the TF-IDF vectors of a user and job respectively.

Since we have a very high-dimensional problem and data are likely linearly separable in high dimension, we used a linear model (linear SVM).

In order to efficiently handle unbalanced datasets, we used different costs for the two classes: c_0 and c_1 for the class 0 (label = 0) and the class 1 (label = 1) respectively.

IV. Experiments

This section presents the experiments that we conducted while developing the proposed recommender systems (see Sections III-B and III-C). The first part is dedicated to the description of the datasets, the results of the basic method are presented in the second part, the third and fourth parts are respectively dedicated to the importance of users and jobs fields in the task of job recommendation and the results of supervised learning procedure.

AUC-ROC (see Section II-D) has been used as performance measure for the experiments. The 95% confidence intervals ($[2.5\%$ quantile, 97.5% quantile$]$) have been computed.

We called Bernoulli method (algorithm), the naive classifier that randomly assigns a class to an entry in the test set using the proportion of positive examples (label = 1) and negative ones (label = 0): its expected AUC-ROC score is 0.5.

A. Description of datasets

We evaluated the performance of the proposed job recommender systems (see Sections III-B and III-C) on three datasets extracted from Work4 test data. Each entry in the three datasets is a 3-tuple (u, v, y) where u and v are the vectors of a given user and job respectively and $y \in \{0, 1\}$ is their associated label. Here are the description of the collected three datasets:

1) Candidates Dataset: users can use Work4 applications to apply to jobs. Work4 records the associated data (user and job chosen). We assume that users only apply to the jobs that match with them, so this dataset contains $15,625$ positive examples (label = 1);

2) Random Dataset: This dataset contains couples of users and jobs that have been randomly drawn from Work4 databases and manually annotated. It contains $3,394$ entries;
3) Feedback Dataset: Feedback from Work4 applications users are stored; this leads to the third dataset: Feedback Dataset with $6,650$ entries. We have mainly used two methods to collect feedback:
   a) Work4 manually samples computed matches to assess their quality, and accumulates the data over time.
   b) When a Facebook or LinkedIn user receives an email from our recommender system saying that some of his friends would be perfect for a given job, he can approve or reject the matches. Its feedback, when provided, is included into databases.

A fourth dataset called "All Dataset" has been artificially created: it is the union of three previous datasets and contains $25,669$ entries.

It is important to note that each job is associated to a job page which generally represents a page of a company. Hence jobs from the same job page are generally similar since they are likely from the same company.

Table I shows the percentage of positive entries (label = 1) and negative entries (label = 0) in each dataset. We can see that Random Dataset is highly unbalanced but it is the only dataset whose proportion label 0 / label 1 is close to the real distribution.

	Labels	
Datasets	Label 0	Label 1
Random	98.4	1.6
Feedback	59.3	40.7
Candidates	0.0	100.0
All	28.4	71.6

Table I: Percentage of label 0 (negative examples) and label 1 (positive examples) in each dataset.

After filtering out stopwords using a list of stopwords, we obtained a dictionary with $108,609$ terms.

Table II shows the percentage of empty fields in each dataset. It is worth noting that a lot of these fields are empty: most social networks users do not fill completely the corresponding forms. This problem is more severe in Facebook than in LinkedIn profiles. Our system must thus deal with incomplete (and very noisy) data.

B. Results

1) Heuristic-based model: Experiments have been conducted to compare weighting functions using cosine similarity on the datasets.

Table III shows the performance of cosine similarity with various weighting functions. It reveals that none of the weighting functions used with cosine similarity yields an AUC-ROC score > 0.5 (score of the Bernoulli method) on the All Dataset. This is a bad result but it is possibly due to the fact that users terms space is different from jobs terms space. To check this hypothesis, AUC-ROC scores have been computed for both Facebook users and LinkedIn users (see Figure 1).

		Datasets		
	Fields	Random	Feedback	Candidates
Facebook	Work history	**77.3**	7.4	**58.6**
	Education history	**74.7**	43.5	**64.7**
	Quotes	**95.4**	**99.0**	**98.8**
	About me	**93.2**	**90.4**	**98.0**
	Interests	**86.8**	**68.8**	**72.6**
LinkedIn	Headline	0.3	0.4	1.5
	Educations	14.0	14.0	12.8
	Positions	1.5	0.3	4.1
Jobs	Title	0.0	0.0	0.0
	Description	0.0	0.0	0.0
	Responsibilities	**72.4**	**91.9**	43.8

Table II: Percentage of empty fields in the three datasets; in bold, fields empty at more than 50%.

	Datasets		
Weighting functions	Random	Feedback	Candidates
Bool	0.781	0.642	0.421
TF	0.779	**0.733**	**0.428**
LTF	**0.784**	0.709	0.426
TF-IDF	0.780	0.716	0.422
Log-Entropy	0.780	0.693	0.421

Table III: Performance (AUC-ROC) of cosine similarity with classic weighting functions, in bold the best results for each dataset.

Figure 1 shows that cosine similarity yields better results for LinkedIn users than for Facebook users: LinkedIn users terms space seems closer to jobs terms space than Facebook one.

2) Importance of fields: To compute the importance of users and jobs fields on the recommendation scores, the constrained optimization problem stated in the Section III-B2 has been solved using the function *minimize* (from Scipy.optimize module) with the method SLSQP (Sequential Least Squares Programming). The costs c_0 and c_1 are set as in [28] $c_0 = \frac{1}{n_0}$ and $c_1 = \frac{1}{n_1}$ where n_0 and n_1 are the number of entries with

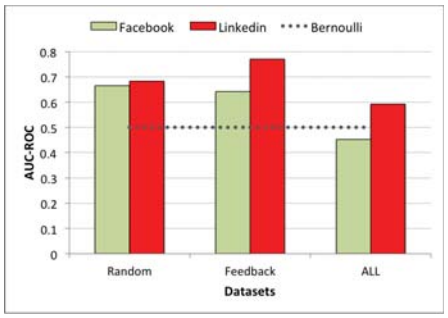

Figure 1: Performance (AUC-ROC) of cosine similarity with TF-IDF: Facebook users versus Linkedin users.

label 0 and label 1 in the training sample respectively.

Bootstrapping (400 runs) has been used to estimate the confidence intervals of the importance of fields. Figures 2, 3 and 4 show the importances of Facebook users', LinkedIn users' and jobs' fields with their confidence intervals respectively. They also suggest that the important fields in the task of job recommendation are:

- Work history field for Facebook users;
- Headline field for LinkedIn users;
- and in Title field for jobs.

The results reveal how Feedback Dataset has been collected: at first sight people only compare the job title to users Work history (Facebook) or Headline (LinkedIn) to send feedback because the Work history (see Figure 2), Headline (see Figure 3) and Title (see Figure 4) importance for Feedback Dataset are greater than those for the other datasets.

Random Dataset results are different from the others: probably due to the fact that it is highly unbalanced.

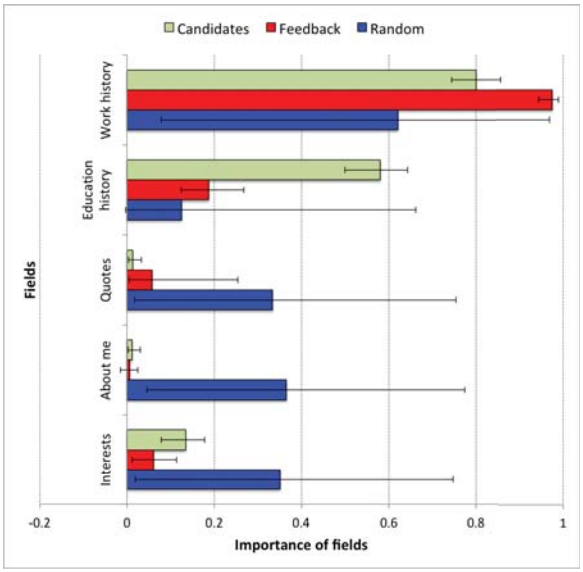

Figure 2: Importance $\alpha^0 \in [-1,+1]^5$ of Facebook users fields: the higher the importance is, the most important the associated field is in the task job recommendation.

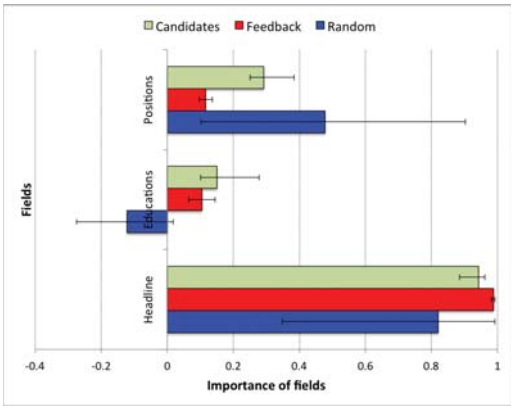

Figure 3: Importance $\alpha^1 \in [-1,+1]^3$ of LinkedIn users fields: the higher the importance is, the most important the associated field is in the task job recommendation.

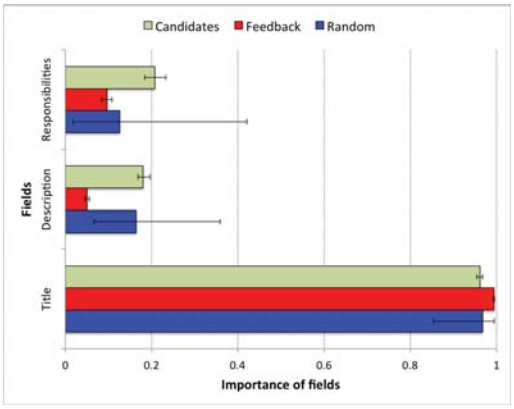

Figure 4: Importance $\beta \in [-1,+1]^3$ of jobs fields: the higher the importance is, the most important the associated field is in the task job recommendation.

The AUCs-ROC (see Table IV) of the generalized method (see Section III-B3) are only slightly greater than those of the basic method (see Section III-B1): possibly due to the fact that many Facebook users fields are empty (see Table II).

3) Linear SVM versus cosine similarity: The following results are from the experiments that we have conducted in order to compare SVM to cosine similarity on our datasets. The confidence intervals are estimated using bootstrapping (100 runs by experiment).

The costs of classes c_0 and c_1 are set as previously (see Section IV-B2). The datasets are randomly split into training set (66%) and test set (34%) using the two following methods:

- method 1: randomly draw a sample (whose size represents 66% of the dataset) and consider it as the training set; the rest of the dataset is considered as the test set;
- method 2: randomly draw a page of jobs, when a page is drawn, all the entries in the dataset whose job is on this page are considered as entries in the training. When the size of the training set is the closest to 66% of the dataset size, the rest of the dataset is considered as the test set.

	Cosine similarity	
Datasets	Basic method	Generalized method
Random	0.707	0.710 (**+0.003**)
Feedback	0.693	0.695 (**+0.002**)
All	0.472	0.483 (**+0.011**)

Table IV: Comparison between basic and generalized cosine similarity AUC-ROC.

Figure 5 shows the comparison between linear SVM models and cosine similarity using the method 1 of splitting. We can see that linear SVM outperforms cosine similarity on every dataset especially on the All Dataset. The difference between the AUC-ROC scores of cosine similarity and linear SVM is high: data in training set and test set are possibly too correlated because jobs on the same page are similar since a page of jobs is generally associated to a company. To avoid this problem, the method 2 has been used (see Figure 6).

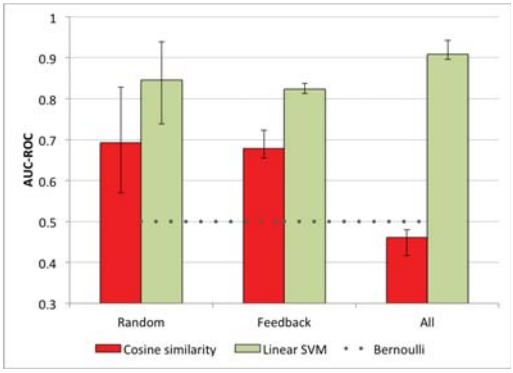

Figure 5: AUC-ROC for the method 1: linear SVM vs cosine similarity.

Figure 6 shows the comparison between linear SVM models and cosine similarity using the method 2 of splitting. The results show that:

- the jobs from Random and Feedback datasets are probably too small to cover all the possible diversity. The models learnt from these datasets do not generalize well to a new job page.
- on All Dataset, the learnt models outperform the cosine similarity: since it is a big dataset, we have a lot of various jobs pages in training sets.

The results here are globally lower than those of the method 1 (see Figure 5) but it seems that the more entries in training set we have, the higher the performance of the learnt models are.

V. Discussion and Future Work

Many experiments have been conducted on real world data from Work4 in order to test the proposed recommender system.

The first series of experiments concluded that Facebook users and LinkedIn users seem to have a vocabulary (set of terms in users profiles) different from the vocabulary of jobs descriptions. However LinkedIn users vocabulary seems closer to the vocabulary of jobs than Facebook one (see Figure 1).

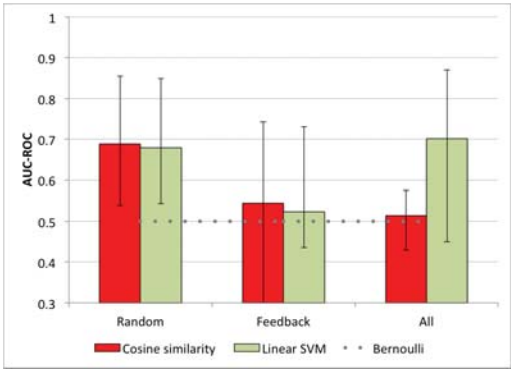

Figure 6: AUC-ROC for the method 2: linear SVM vs cosine similarity.

Not surprisingly, the most relevant fields for users are "Work history" (Facebook) and "Headline" (LinkedIn). On jobs, the is "Title" brings the most important information. For instance, given a Facebook or LinkedIn user and a job description, if one tries to tell whether this user matches with this job, one will probably first compare the user's work history (Facebook) or headline (LinkedIn) field information to the title of the job.

The computed fields importance used to generalize our basic model only slightly increased the AUC-ROC scores. This is possibly due to the fact many Facebook fields are almost empty (see Table II). Another reason could be that AUC-ROC is not the right performance measure for our problem. We will try other performance measures (see Section II-D) in our future work.

Linear SVM yields better results but needs to be trained on lots of jobs pages to be able to make accurate predictions on unseen jobs pages.

We have assumed in this paper that users only apply to jobs that match with them (Candidates Dataset), this assumption could be partially false for many reasons (users were testing Work4 applications when applied to jobs, etc.). We will partially manually annotate entries in Candidates Dataset in our future work.

Statistics from our datasets (see Table II) showed that a vast majority of Facebook users fields are almost empty. This has raised a big problem: we cannot accurately make recommendations to users whose profiles are almost empty using the proposed recommender system. Consequently, we are currently working on a social recommender system which uses both user's interests and his similar friends' interests to recommend jobs to him. The use of social users social connections data could mitigate the problem of Facebook users with few terms in their profiles which would lead to more accurate recommendations: recommendations are made using not only users interests but their similar friends interests for jobs.

VI. Acknowledgements

This work is supported by Work4 and ANRT[1], the French National Research and Technology Association. The authors thank Guillaume Leseur, software architect at Work4, Benjamin Combourieu, Anne-Claire Haury and all the Work4 *Engines* team.

References

[1] D. M. Boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2008.

[2] B. Xiao and I. Benbasat, "E-commerce product recommendation agents: use, characteristics, and impact," in *MIS Quarterly*, vol. 31, no. 1. Society for Information Management and The Management Information Systems Research Center Minneapolis, MN, USA, March 2007, pp. 137–209.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[4] J. Bennett, S. Lanning, and N. Netflix, "The netflix prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[5] "Facebook," Feb. 2013. [Online]. Available: http://newsroom.fb.com/Key-Facts

[6] May 2012. [Online]. Available: http://www.go-gulf.com/blog/social-networking-user/

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[8] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. Cambridge University Press, 2011.

[9] R. A. Baeza-Yates and R.-N. Berthier, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[10] G. Salton, A. Wang, and C. Yang, "A vector space model for information retrieval," *Journal of the American Society for Information Science*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[11] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks." *Int. J. Approx. Reasoning*, vol. 51, no. 7, pp. 785–799, 2010.

[12] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," 2008.

[13] J. Séguela, "Fouille de données textuelles et systèmes de recommandation appliqués aux offres d'emploi diffusées sur le web," Ph.D. dissertation, Conservatoire National des Arts et Métiers (CNAM), Paris, France, May 2012.

[14] M. Balabanovic and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Comm. ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997.

[15] P. Kazienko, K. Musiał, and T. Kajdanowicz, "Multidimensional Social Network in the Social Recommender System," vol. 41, no. 4, pp. 746–759, Jul. 2011.

[16] J. J. Rocchio, "Relevance feedback in information retrieval," in *The SMART Retrieval System: Experiments in Automatic Document Processing*, ser. Prentice-Hall Series in Automatic Computation, G. Salton, Ed. Prentice-Hall, Englewood Cliffs NJ, 1971, ch. 14, pp. 313–323.

[17] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of SIAM Data Mining (SDM'05)*, 2005. [Online]. Available: http://www.daniel-lemire.com/fr/documents/publications/lemiremaclachlan_sdm05.pdf

[18] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," 2008.

[19] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," 1999.

[20] I. S. C. Nicholas and C. K. Nicholas, "Combining content and collaboration in text filtering," in *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, 1999, pp. 86–91.

[21] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 448–456.

[22] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine Learning*, vol. 27, pp. 313–331, 1997.

[23] T. Joachims, "Text categorization with suport vector machines: Learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning*, ser. ECML '98. London, UK, UK: Springer-Verlag, 1998, pp. 137–142.

[24] 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)*, vol. 3, Sept. 2010.

[25] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[26] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[27] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[28] A. Anand, G. Pugalenthi, G. B. Fogel, and P. N. Suganthan, "An approach for classification of highly imbalanced data using weighting and undersampling." *Amino Acids*, vol. 39, no. 5, pp. 1385–91, 2010.

[1] ANRT: Association Nationale de la Recherche et de la Technologie.