Soc. Netw. Anal. Min. (2014) 4:227
DOI 10.1007/s13278-014-0227-z

ORIGINAL ARTICLE

Exploration of methodologies to improve job recommender systems on social networks

Mamadou Diaby · Emmanuel Viennet · Tristan Launay

Received: 13 December 2013 / Revised: 15 July 2014 / Accepted: 1 August 2014

Abstract This paper presents content-based recommender systems which propose relevant jobs to Facebook and LinkedIn users. These systems have been developed at Work4, the Global Leader in Social and Mobile Recruiting. The profile of a social network user contains two types of data: user data and user friend data; furthermore, the profile of our users and the description of our jobs consist of text fields. The first experiments suggest that to predict the interests of users for jobs using basic similarity measures together with data collected by Work4 can be improved upon. The next experiments then propose a method to estimate the importance of users' and jobs' different fields in the task of job recommendation; taking into account these weights allow us to significantly improve the recommendations. The third part of this paper analyzes social recommendation approaches, validating the suitability for job recommendations for Facebook and LinkedIn users. The last experiments focus on machine learning algorithms to improve the obtained results with basic similarity measures. Support vector machines (SVM) shows that supervised learning procedure increases the performance of our content-based recommender systems; it yields best results in terms of AUC in comparison with other investigated methodologies such as Matrix Factorization and Collaborative Topic Regression.

M. Diaby (✉) · E. Viennet
Université Paris 13, Sorbonne Paris Cité, L2TI, 93430 Villetaneuse, France
e-mail: mamadou.diaby@univ-paris13.fr; mdiaby@work4labs.com

E. Viennet
e-mail: emmanuel.viennet@univ-paris13.fr

M. Diaby · T. Launay
R&D Department of Work4, 3 Rue Moncey, 75009 Paris, France

Keywords Job recommendation · Facebook · LinkedIn · Content-based recommender system · Social recommender systems · SVM

1 Introduction

Since the beginning of the 2000s, three famous social networks have been launched: LinkedIn in 2003, Facebook in 2004 and Twitter in 2006; each of them is counting millions or even billions of users around the world (Facebook 2014; LinkedIn 2014; Brain 2014). A social network site is often defined as a web-based service that allows users to construct profiles (public or semi-public), to share connections with other users and to view and traverse lists of connections made by others in the system (Boyd and Ellison 2008).

Personal information posted by users of a social network (which may involve personal description, posts, ratings, likes, but also social links) can be exploited by a recommender system in order to propose or advertise relevant items to them; as a result, social network data are becoming very attractive to many companies around the world. Recommender systems are often defined as software that elicit the interests or preferences of individual consumers for products, either explicitly or implicitly, and make recommendations accordingly (Xiao and Benbasat 2007). They have many practical applications (Linden et al. 2003; Bennett et al. 2007) that help users deal with information overload, reason why they have become an active research field for two decades.

This paper presents online recommender systems that extract social network users' interests for job offers and then make recommendations to them accordingly; it is focused on Facebook and LinkedIn. A variant of those recommender systems is used by Work4, the Global Leader in Social and Mobile Recruiting. Due to privacy concerns the proposed recommender systems only use the data that social network users explicitly granted access to; they also deal with noisy, missing and unstructured data from social network users and job descriptions.

The contributions of this paper are fourfold:

1. the first series of experiments on real-world data from Work4 reveals that Cosine similarity used to recommend jobs to social networks users (Sect. 4.2) gives results that can be improved;
2. we estimate the importance of each field of users and jobs in the task of job recommendation (Sect. 4.3) and the application of these importance improves the results with Cosine similarity (Sect. 4.4).
3. our third series of experiments show that the use of basic social recommendation methods failed to improve our results; however, the use of Rocchio's method to enrich users' vectors with their related jobs' data improves the results (Sects. 4.5 and 4.6).
4. our experiments on machine learning show that the use of support vector machine significantly improves our results (Sect. 4.7). This method outperforms two methods of the literature (Sect. 4.8): a collaborative filtering (CF) method and the collaborative topic regression (CTR) method proposed by Wang and Blei (2011).

The rest of this paper is organized as follows: we summarize work done on recommender systems over the last two decades in the Sect. 2, the Sects. 3 and 4, respectively, present the proposed recommender systems and the obtained results and we sum up and conclude in Sect. 5.

2 Related Work

Recommender systems help users deal with information overload by recommending items that would be of interest for them. There has been a lot of work done on designing recommender systems during the last two decades. Amazon.com (Linden et al. 2003) and Netflix (Bennett et al. 2007) are two popular applications of recommender systems.

2.1 Recommender systems

Recommender systems (Adomavicius and Tuzhilin 2005; Jannach et al. 2011) are mainly related to information retrieval (Baeza-Yates and Berthier 1999; Salton et al. 1975), machine learning (de Campos 2010; Salakhutdinov and Mnih 2008a), data mining (Séguela 2012; Han 1996) and other research fields beyond the scope of this study. Adomavicius and Tuzhilin (2005) 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; this is also known as top-k recommendations. For instance, let us 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 a rating-based recommender system in the rest of this document.

Recommender systems are generally classified into three or four categories (Bobadilla et al. 2013; Kazienko et al. 2011; Adomavicius and Tuzhilin 2005; Balabanovic and Shoham 1997): content-based methods, collaborative filtering, demographic filtering systems and hybrid approaches.

Content-based recommender systems (Lops et al. 2011; Pazzani and Billsus 2007) use the ratings that users gave to items in the past to define their profiles (Adomavicius and Tuzhilin 2005); if users have associated descriptions, this information is also taken into account. Content-based systems use the description of items to extract their profiles. In this paper, we develop content-based recommender systems that use both users' and jobs' textual descriptions to make recommendations.

In contrast to content-based systems, collaborative filtering methods use the opinion of a community of users similar to the active user to recommend items to him (Jannach et al. 2011; Salakhutdinov and Mnih 2008b; Lemire and Maclachlan 2005). Conversely, rating predictions of an item involves known ratings of similar users. Wang and Blei (2011) classified collaborative filtering methods into two main groups: matrix factorization methods and neighborhood methods. Neighborhood methods determine a set of most similar users to the active user and then determine the interest of the active user for items by aggregating the interests of the similar users. Matrix factorization (Salakhutdinov and Mnih 2008b; Wang and Blei 2011) is a method of latent factor models in which users and items are represented in low-dimensional space; the new representations of users and items are commonly computed by minimizing the regularized squared error.

Recommendations are made in demographic filtering systems by using users personal attributes (age, gender, income, country, survey responses, etc.) (Bobadilla et al. 2013; Kazienko et al. 2011; Gao et al. 2007).

Hybrid recommender systems combine two or more types of recommender systems into a single model; they generally yield better results than simple recommendation techniques (Adomavicius and Tuzhilin 2005) but are much more complex to design. In the particular case of the combination of a content-based system and a collaborative filtering technique, there are mainly four combinations (Adomavicius and Tuzhilin 2005):

1. aggregating the two recommendations of a content-based and collaborative filtering system (Claypool et al. 1999);
2. adding content-based characteristics to a collaborative filtering system: similarities between users or items are computed using content-based data (Balabanovic and Shoham 1997);
3. adding collaborative filtering to a content-based system: dimensionality reduction techniques on users' profiles obtained by combining content and collaborative data (Nicholas and Nicholas 1999);
4. developing a single unifying recommendation model that uses both content-based and collaborative data (-Wang and Blei 2011).

A new emerging category of recommender systems is social recommender systems which use both the active user's opinion and the opinions of his social connections (friends on a social network, users with behavior or interests similar to the active user's) to make recommendations to him. This category of recommender systems is gaining popularity with the rapid growth of social networks in recent years. In the literature, many methods have been developed to use both users' and their friends' data to make recommendations: combination of the matrix factorization and friendship links to make recommendations (Aranda et al. 2007), models that use the matrix factorization making sure that either the latent vectors of users are as close as possible to the weighted sum of those of their friends or the latent vectors of users are as close as possible to those of their friends individually (Ma et al. 2011). Currently, the results about social recommender systems seem mixed (Kantor 2009): some papers reported that social recommender systems are no more accurate than classic ones except in special cases (Golbeck 2006), while others argued that they yield better results than the classic ones (Groh and Ehmig 2007).

2.2 Data representation

In recommender systems, textual description of a document (user or item) is generally 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 weighting functions and the "bag-of-words" model and by filtering out unimportant terms for the task of recommendation. 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. There are many ways to filter out unimportant terms:

- define a list of stop words that will automatically be removed from the corpus. We can notice that a list of stop words depends on the problem one is solving; a list of all stop words for a specific task of recommendation is unknown most of time;
- filter out the very high and very low frequency terms (Séguela 2012), which requires to define two thresholds: one for high-frequency terms and one for low-frequency ones. To set these two thresholds, one need to conduct experiments on his datasets.
- another way is to filter out some grammatical categories of words which requires to determine the language and nature of words in the corpus; this makes this method slower than the previous two ones.

Weighting functions calculate the importance of a term for a given document; they are generally classified into three main categories (Séguela 2012): local functions, global 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; a most often mentioned local weighting function in the literature is the normalized Term Frequency (TF) defined as follows:

$$\mathrm{TF}_{t,d} = \frac{f_{t,d}}{\max_k f_{k,d}}, \tag{1}$$

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

Two other methods are the boolean weight (Bool) and Log Term Frequency (LTF) defined as follows:

$$\mathrm{Bool}_{t,d} = \begin{cases} 1 & \text{if } t \in d \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

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

Global weighting functions use the whole corpus (set of documents) to calculate the weight of a given term. The first global weighting function we can cite is the Inverse Document Frequency (IDF):

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

where N is the total number of documents in the corpus, n_t is the number of documents that contain the term t.

Another global weighting function is the Entropy defined as follows:

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

where $p_d^t = \frac{f_{t,d}}{\sum_k f_{t,k}}$ is the probability that the term t belongs to the document d and $f_{t,d}$ is defined in (1).

In the literature, combinations of a local weighting function and a global weighting function generally give better results than local weighting functions (Salton et al. 1975). TF–IDF is the most famous combination; it is defined as follows:

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

where t is a term and d is a document.

Another combination is Log-Entropy defined as follows:

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

We use TF–IDF as weighting function in our proposed recommender systems for two reasons: the comparison of different weighting functions shows that we obtain almost the same performance for different weighting functions and TF–IDF is known to yield good results in information retrieval (Salton et al. 1975).

2.3 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 while others are learnt models from underlying data using machine learning techniques.

Two well-known similarity measures (Adomavicius and Tuzhilin 2005; Jannach et al. 2011) are cosine similarity and Pearson correlation coefficient (PCC).

Cosine similarity is mostly used in content-based recommender systems: it yields better results in item–item filtering systems (Jannach et al. 2011). It measures the cosine of the angle between two vectors and is defined as follows:

$$\cos(u,v) = \frac{\sum_{k=1}^K u_k v_k}{\sqrt{\sum_{k=1}^K u_k^2}\sqrt{\sum_{k=1}^K v_k^2}} \tag{8}$$

where u and v are the vectors of users or items and K is the number of dimensions of u and v.

PCC is mainly used in collaborative filtering techniques and is defined as follows:

$$\mathrm{PCC}(u,v) = \frac{\sum_{k=1}^K (u_k - \bar{u})(v_k - \bar{v})}{\sqrt{\sum_{k=1}^K (u_k - \bar{u})^2}\sqrt{\sum_{k=1}^K (v_k - \bar{v})^2}} \tag{9}$$

where u and v are the vectors of users or items, $\bar{u}$ and $\bar{v}$ the mean values of u and v, respectively.

In the literature, we can meet other similarity functions like the mean squared difference (dissimilarity measure) (Shardanand and Maes 1995), the Gaussian and Exponential similarity functions (Séguela 2012) (based on the mean squared difference dissimilarity) and defined as follows:

$$\mathrm{Gaussian}(u,v) = \exp\left(-\frac{\sum_{k=1}^K (u_k - v_k)^2}{2\sigma^2}\right) \tag{10}$$

$$\mathrm{Exponential}(u,v) = \exp\left(-\frac{\sqrt{\sum_{k=1}^K (u_k - v_k)^2}}{\sigma}\right) \tag{11}$$

where σ is the parameter of the standard deviation to be set.

Classic similarity measures (Cosine similarity, PCC, ...) can work on some specific problems, but do not work on others: Cosine similarity yields better results in item-item filtering systems (Jannach et al. 2011) but in content-based recommender systems (see Sect. 2.1), if the user term space is not completely equal to the item term space, the computed similarities between users and items using Cosine similarity or PCC could be close to 0. In the literature, learnt similarity models from underlying data have been successfully used because they neatly fit the problems to be solved. Bayesian Networks (Pazzani and Billsus 1997), SVMs (Diaby et al. 2013; Joachims 1998) are two examples of methods used by researchers.

2.4 Performance metrics

For a reminder, we have two main groups of recommender systems (Adomavicius and Tuzhilin 2005): rating-based systems and preference-based filtering techniques. The group of a recommender system determines the set of performance metrics we can use to assess its performance.

A rating-based system is a predictive system (see Section 2.1); many performance metrics have been used to assess the performance of predictive systems Omary and Mtenzi (2010); among them we can cite the Precision, Recall, $F_\beta-$measure, RMSE (root mean square error) and MAE (mean absolute error).

The Precision refers to the capacity of a predictive system to be precise (in the prediction of different classes), while the Recall refers to its ability to find all elements of a specific class; they are defined as follows:

$$P(c) = \frac{\text{number of items correctly affected to c}}{\text{number of items affected to c}} \tag{12}$$

$$R(c) = \frac{\text{number of items correctly affected to c}}{\text{number of items that belong to c}} \tag{13}$$

where $P(c)$ and $R(c)$ are, respectively, the Precision and Recall for the class $c \in C$ and C is the set of classes.

A predictive system can have a high Recall with a low Precision or vice versa, that is why F_β (Rijsbergen 1979) has been designed to take into account the Recall and the Precision; F_1 is the most often mentioned in the literature. F_β for a class c is defined as follows:

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

The global Precision, Recall and F_β can be computed as the average or weighted sum of the performance of the different classes (Séguela 2012).

To compute the above performance metrics for a recommender system, we need to set a threshold: a given item is recommended to a given user if his interest for this item is greater than the threshold. Setting thresholds is sometimes tedious, that why we use the AUC-ROC (also known as AUC) as performance metric for our recommender systems; AUC-ROC is the area under the curve of a ROC (receiver operating characteristic) (Omary and Mtenzi 2010) obtained by plotting the TP rate (fraction of true positives) as a function of FP rate (fraction of false positives). It is used in binary classification tasks. The higher the AUC of a classifier is, the better the system is. We can notice that the minimum value of AUC is 0 while its maximum value is 1, but the AUC for a classifier that randomly assigns the different classes is close to 0.5. If the AUC of a system is below 0.5, one can inverse each of the predictions to obtain an AUC greater than 0.5.

The MAE and RMSE (Ma et al. 2011) are generally used in regression recommendation problems and they are defined as follows:

$$\mathrm{MAE} = \frac{1}{|\Gamma|}\sum_{i,j\in\Gamma} |r_{ij} - \hat{r}_{ij}| \tag{14}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\Gamma|}\sum_{i,j\in\Gamma} (r_{ij} - \hat{r}_{ij})^2} \tag{15}$$

where r_{ij} and $\hat{r}_{ij}$ are the original and predicted ratings, respectively and Γ is the test set; in regression problems r_{ij} and $\hat{r}_{ij}$ have continuous values while their values are discrete in classification problems.

In the top-K recommender systems also known as preference-based filtering recommender systems (in which the system computes a list of K items to be recommended to each user), we have some interesting metrics like MAP@K (Mean Average Precision) and the NDCG@K (Normalized Discount Cumulative Gain).

Sometimes in the literature we meet an adaptation of predictive systems' performance metrics to the top-K recommendations like the Recall@K (Recall for the top K items recommended to users) used in Wang and Blei (2011); one can also plot the Recall@K as a function of the number (K) of recommended items and eventually compute the area under this curve.

The MAP Aiolli (2013) metric has been used in the MSD (Million Song Dataset) challenge[1], it is defined as follows:

$$\mathrm{MAP@K} = \frac{1}{|U|}\sum_{u\in U}\frac{1}{K}\sum_{k=1}^{K}\frac{C_{uk}}{k} \times 1_{uk} \tag{16}$$

where U is the set of users and C_{uk} is the number of correct ranked in descending order of the ratings to the user u and

$$1_{uk} = \begin{cases} 1 & \text{if item at rank k is correct (for the user u)} \\ 0 & \text{otherwise} \end{cases}$$

The discount cumulative gain (DCG) is defined as follows:

$$\mathrm{DCG(b)@K} = \frac{1}{|U|}\sum_{u\in U}\sum_{k=1}^{K}\frac{r_{uk}}{\max(1, \log_b(k))} \tag{17}$$

where r_{uk} is the rating that the user u gave to the item at the position k in the ranked list of K items recommended to u.

The normalized discount cumulative gain (NDCG) (Ravikumar et al. 2011) is therefore defined as follows:

$$\mathrm{NDCG\ (b)@K} = \frac{\mathrm{DCG\ (b)@K}}{\mathrm{IdealDCG\ (b)@K}} \tag{18}$$

where the IdealDCG (b)@K is the DCG (b)@K for the ideal ranking of top K items (for each user, his top K items are ranked in descending order of the ratings he gave to them).

3 Social network-based job recommender systems

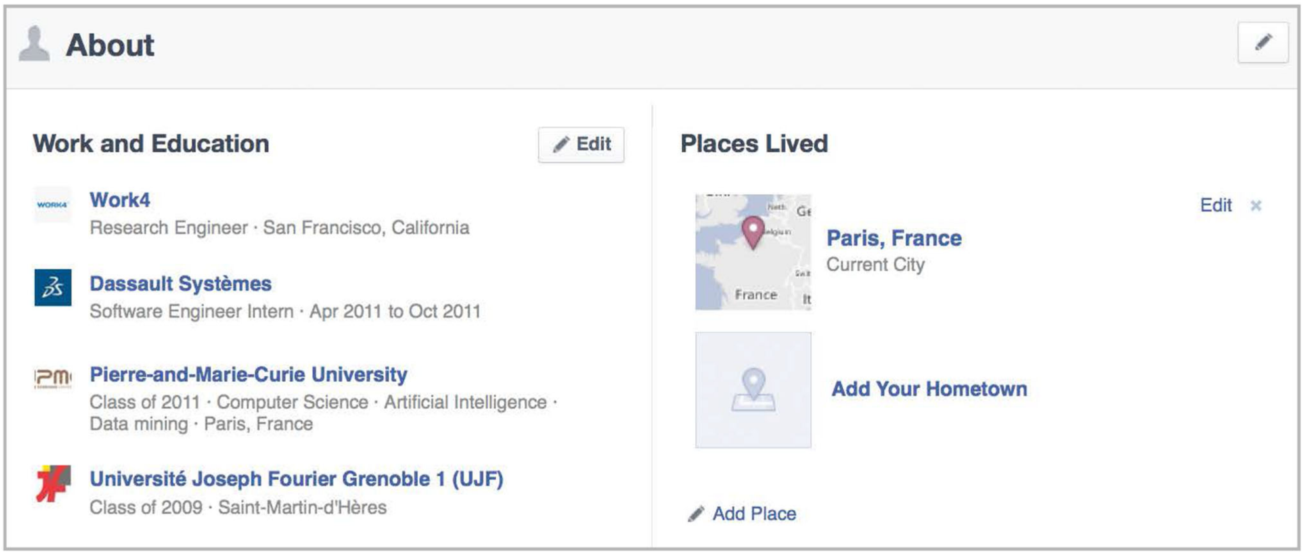
This section presents the description of our proposed job recommender systems: we first present the modeling of our social network users and job offers and then describe the proposed recommender systems.

3.1 Document modeling

The efficiency of a recommender system generally depends on how users and items are represented. Our social network users are defined by two types of data: user data and their social connection data.

- user data are both those that users post to social networks and those recorded while they were interacting with the platform. Publications, comments, likes, time spent reading or viewing resources are some examples of interaction data.

[1] http://www.kaggle.com/c/msdchallenge.


About

Work and Education

Work4
Research Engineer · San Francisco, California

Dassault Systèmes
Software Engineer Intern · Apr 2011 to Oct 2011

Pierre-and-Marie-Curie University
Class of 2011 · Computer Science · Artificial Intelligence · Data mining · Paris, France

Université Joseph Fourier Grenoble 1 (UJF)
Class of 2009 · Saint-Martin-d'Hères

Places Lived

Paris, France
Current City

Add Your Hometown

Add Place

Fig. 1 An example of a Facebook profile; we note three fields (work, education, places lived) with their sub-fields: company name, position, start date, end date, location, class of, college/university/school name, description and concentrations

- social connection data are the data of users' connections on a social network—for instance, the list of friends of social network user.

The proposed recommender systems (see Sects. 3.2, 3.4, 3.5 and 3.7) only use user data and the description of jobs to predict users' interests for jobs while the methods proposed in Sect. 3.6 also use the social connection data. Our Facebook users have authorized the Work4's applications to access 5 fields data: Work, Education, Quote, Bio, and Interests; LinkedIn users have only authorized 3 fields: Headline, Educations, Positions; our jobs descriptions have 3 fields: Title, Description, Responsibilities. Figures 1, 2 and 3, respectively show an example of a Facebook profile, a LinkedIn profile and a job description.

For each document (user or job), we extract a vector for each of its fields (which contain textual information) using the "bag-of-words" model and TF-IDF as weighting function; we also filter out stop words using lists defined by Work4. The vector of a document is computed as a weighted sum of the vectors of its different fields; each weight is the importance of the associated field. Figure 4 shows how we aggregate the vectors of different fields for Facebook users, LinkedIn users and Jobs.

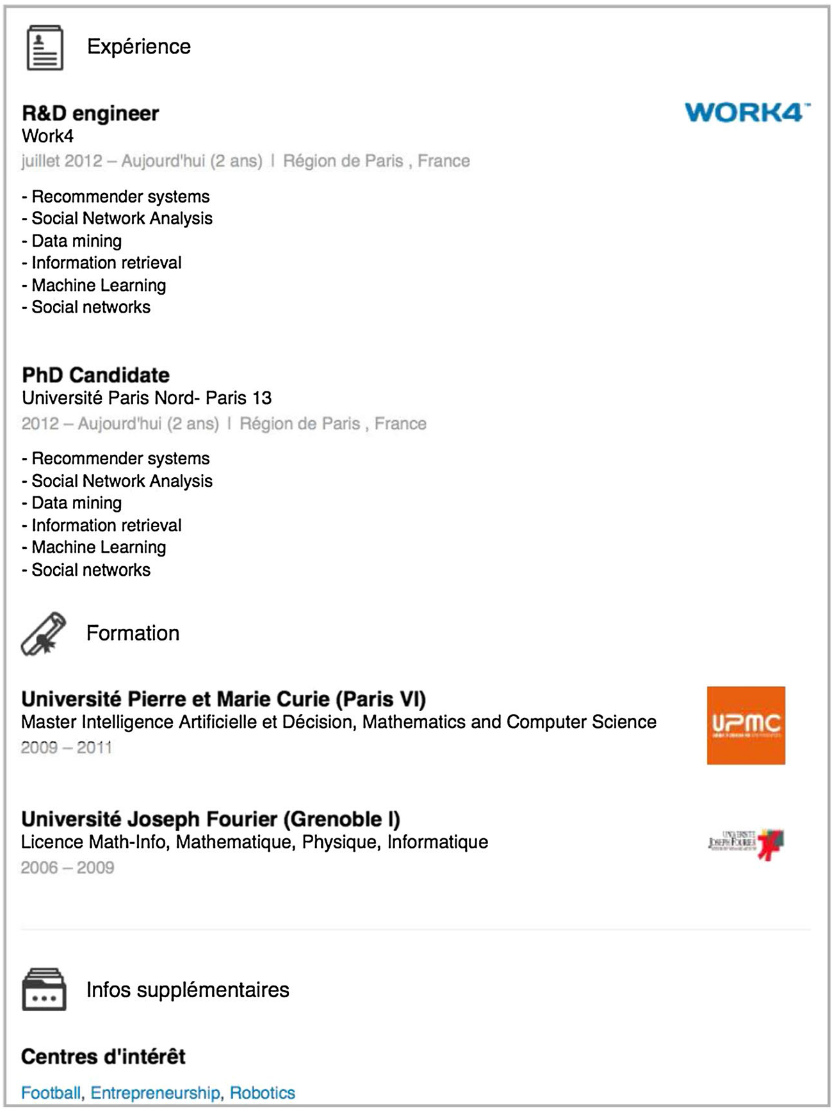
Users' term space is not completely equal to the jobs' term space: users generally use an informal vocabulary (contrasted to job descriptions): users' data contain many typos, abbreviations, some teen text terms, etc. To mitigate this issue, we filter out some terms (stop words) and we use the data contained in only some sub-fields of users' fields. Thus we only use the data in the:

- *position* and *description* sub-fields of the Facebook *Work* field,
- *concentrations* and *description* sub-fields of the Facebook *Education* field,
- Facebook *Quote*, *Bio* and *Interests* fields and LinkedIn *Headline* field (they have no sub-field),
- *position* and *description* sub-fields of the LinkedIn *Positions* field,
- *degree*, *fieldOfStudy*, *notes* and *activities* sub-fields of the LinkedIn *Educations* field.

It is also important to note that the vectors of users and jobs depend on their languages. We assign the language " Other " to all documents (users and jobs) whose languages are different from English and French since we are focused on only these two languages.

3.2 First recommender system: Engine-1

In Engine-1 the TF-IDF vectors of users and jobs are computed (following the method described in Sect. 3.1) by assuming that all the fields have the same importance ($\alpha_w^0 = \alpha_e^0 = \alpha_b^0 = \alpha_q^0 = \alpha_i^0 = 1$ and $\alpha_h^1 = \alpha_e^1 = \alpha_p^1 = 1$ and $\beta_t = \beta_d = \beta_r = 1$) on recommendation scores. We measure the interest of a user for a given job by computing the cosine similarity (8) of the user's vector and the vector of the job. One weakness of this system is that the assumption of all the fields have the same importance is probably false. If the users' term space is quite different from the jobs' term space, Engine-1 will fail to make proper recommendations; this could be another weakness of this system, but


Expérience

R&D engineer
Work4
juillet 2012 – Aujourd'hui (2 ans) | Région de Paris , France

- Recommender systems
- Social Network Analysis
- Data mining
- Information retrieval
- Machine Learning
- Social networks

PhD Candidate
Université Paris Nord- Paris 13
2012 – Aujourd'hui (2 ans) | Région de Paris , France

- Recommender systems
- Social Network Analysis
- Data mining
- Information retrieval
- Machine Learning
- Social networks

Formation

Université Pierre et Marie Curie (Paris VI)
Master Intelligence Artificielle et Décision, Mathematics and Computer Science
2009 – 2011

Université Joseph Fourier (Grenoble I)
Licence Math-Info, Mathematique, Physique, Informatique
2006 – 2009

Infos supplémentaires

Centres d'intérêt
Football, Entrepreneurship, Robotics

Fig. 2 An example of a LinkedIn profile with positions and educations fields and their sub-fields

we only use the data in some sub-fields to mitigate this issue (as described in Sect. 3.1).

3.3 Importance of fields

We know that it is unlikely that all the users' or jobs' fields have the same importance on the recommendation scores: some might be more important than others. In order to determine the importance of different fields of users and jobs, we propose the optimization problem below. 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:

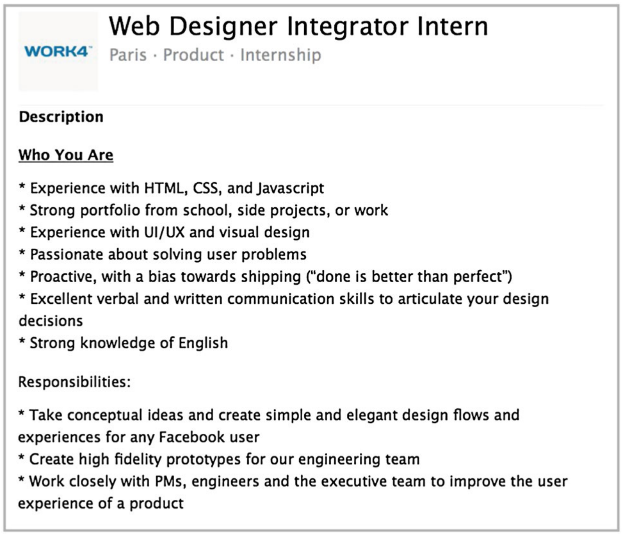
$$u(\alpha) = \sum_{f=1}^{f_u^0} \alpha_f^0 u_f^0 + \sum_{f=1}^{f_u^1} \alpha_f^1 u_f^1 \qquad (19)$$

where $\alpha = (\alpha^0, \alpha^1)$, $\alpha^0 = (\alpha_1^0, \cdots, \alpha_{f_u^0}^0)$ and $\alpha^1 = (\alpha_1^1, \cdots, \alpha_{f_u^1}^1)$ 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:

$$v(\beta) = \sum_{f=1}^{f_j} \beta_f v_f \qquad (20)$$

where $\beta = (\beta_1, \cdots, \beta_{f_j})$, 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.


Web Designer Integrator Intern

Paris · Product · Internship

Description

Who You Are

* Experience with HTML, CSS, and Javascript
* Strong portfolio from school, side projects, or work
* Experience with UI/UX and visual design
* Passionate about solving user problems
* Proactive, with a bias towards shipping ("done is better than perfect")
* Excellent verbal and written communication skills to articulate your design decisions
* Strong knowledge of English

Responsibilities:

* Take conceptual ideas and create simple and elegant design flows and experiences for any Facebook user
* Create high fidelity prototypes for our engineering team
* Work closely with PMs, engineers and the executive team to improve the user experience of a product

Fig. 3 An example of a job with the title, description and responsibilities fields

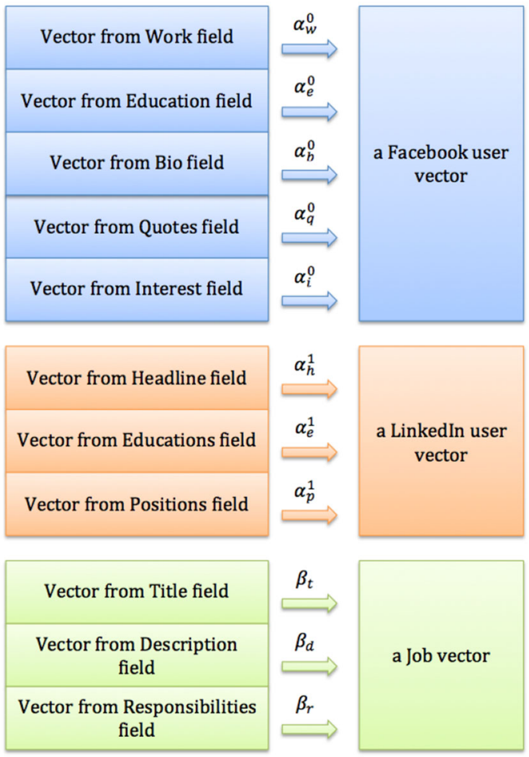


Fig. 4 Aggregation of vectors from different fields for Facebook users, LinkedIn users and Jobs; we can use different values for α^0, α^1 and β

The predicted similarity $\hat{y}(u, v, \alpha, \beta)$ between a user u and a job v using the importance of fields α and β is computed as follows:

$$\hat{y}(u, v, \alpha, \beta) = \cos(u(\alpha), v(\beta)) \tag{21}$$

where cos is defined by the Eq. (8).

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

$$E_\Gamma(\alpha, \beta, c_0, c_1) = \frac{1}{|\Gamma|} \sum_{(u,v,y)\in\Gamma} c_y \cdot (y - \hat{y}(u, v, \alpha, \beta))^2. \tag{22}$$

Each entry of Γ is a 3-tuple (u, v, y) where u, v and $y \in \{0, 1\}$ represent a user, a job and their label, respectively, 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 $\hat{y}(u, v, \lambda_1\alpha, \lambda_2\beta) = \hat{y}(u, v, \alpha, \beta)$, which means that if the optimization problem has a solution, it is not unique. Therefore, we solved a constrained optimization problem to reduce the number of solutions; the constraints are: $\|\alpha^0\| = 1$, $\|\alpha^1\| = 1$ and $\|\beta\| = 1$.

3.4 Second recommender system: Engine-2

In Engine-2, we have used the optimal importance (α^{0*}, α^{1*}, β^*) of users' and jobs' fields (see Sect. 4.3) when computing the vectors of documents as weighted sum of vectors of their fields (following the method described in Sect. 3.1). The interest of a user for a given job is then measured by computing the cosine similarity (8) of the user's vector and the vector the job.

3.5 Third recommender system: Engine-3

We also explored the use of trained statistical models: SVMs (Support Vector Machines). SVMs (Cortes and Vapnik 1995) are known to yield good performance in text categorization (Joachims 1998). Using libSVM (Chang and Lin 2011), we therefore apply this supervised learning procedure to our job recommendation problem; this leads to our third recommender system Engine-3.

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

$$I_{\mathrm{SVM}}(u, v) = (w(u_1, v_1), \ldots, w(u_T, v_T))$$

where T is the total number of terms in the dataset (or the total number of selected terms if one selects the most important terms), u and v are, respectively, the TF-IDF vectors of a user and job obtained by using the importance (α^{0*}, α^{1*}, β^*) of different fields computed in the Sect. 4.3 and $w(x, y)$ is a monotonic function.

We then define w as follows:

$$w(x, y) = \log(1 + cantor(\lfloor x \rceil, \lfloor y \rceil))$$

where $\lfloor x \rceil$ is the nearest integer to x and *cantor* is the Cantor pairing function defined by:

$$cantor(x, y) = \frac{1}{2}(x + y)(x + y + 1) + y$$

where x and y are integers.

The Cover's theorem (Cover 1965) states that data are more likely to be linearly separable in high dimension. We have a very high-dimensional problem (see Sect. 4.1), so we decide to use a linear model (linear SVM): simpler, quicker and avoid the over-fitting problem in general.

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

3.6 Social recommender systems: Engine-4a, Engine-4b, Engine-4c

Since our Facebook users do not completely fill the fields that are interesting in job recommendation task (see Table 2), we experiment the use of the data of the friends of Facebook and LinkedIn users to partially recover missing information in order to improve our recommendations. The use of both users' and their friends' data to make recommendations is known in the literature as social recommendation (Ma et al. 2011; Kantor 2009; Aranda et al. 2007).

Firstly, we propose a generic pseudo algorithm for social recommendation (see the pseudo Algorithm 1).

Pseudo algorithm 1: Generic pseudo algorithm for social recommendations on social networks.

```
Data: uData: Users data,
      fData: Friends data,
      jData: Jobs data,
      $\alpha$: importance of users data,
      $\theta_{\text{classmates}}$: threshold for classmates,
      $\theta_{\text{ex-colleagues}}$: threshold for (ex-)colleagues
Result: Social recommendations
1  foreach User ∈ uData do
2      Find the clusters of his friends densely connected
       using community detection and/or clustering methods
       on fData;
3      foreach Cluster of friends do
4          Extract statistics about colleges/universities and
           companies using the sub-fields of Facebook and
           LinkedIn users' profiles;
5      end
6      Using these statistics, select the relevant clusters
       (communities of (ex-)colleagues and (ex-)classmates)
       using the thresholds $\theta_{\text{ex-colleagues}}$ and $\theta_{\text{classmates}}$;
7      foreach Job ∈ jData do
8          Compute the interest of User for Job using both
           his data (with the weight $\alpha$) and the data from
           relevant communities of friends (with the weight
           $(1 - \alpha)$);
9      end
10 end
```

Unfortunately, due to privacy concerns, we did not obtain enough friendship links to directly apply the pseudo Algorithm 1: our statistics show that a large majority of users (more than 80 %) have less than 10 friends whose profiles data are available (users are not willing to share information about their friends). As a result, we finally use a simplified version of the pseudo Algorithm 1 which uses for each user his data and all the data of his friends: the global interest of a user u for a job v is computed as a linear combination of the interest of u for v and the average interests of his friends for v. We define 3 basic social recommendation methods: Engine-4a, Engine-4b and Engine-4c. The enriched score of a user u for a job v is computed in Engine-4a following the Eq. (23), in Engine-4b following the Eq. (24) and in Engine-4c following the Eq. (25).

$$\text{social score}(u, v) = \cos(\alpha u + (1 - \alpha)\overline{u_F}, v) \tag{23}$$

$$\text{social score}(u, v) = \alpha \cos(u, v) + (1 - \alpha)\cos(\overline{u_F}, v) \tag{24}$$

$$\text{social score}(u, v) = \alpha \cos(u, v) + \frac{(1 - \alpha)}{|F|}\sum_{f \in F} \cos(u^f, v) \tag{25}$$

where u is the original vector of the user, $\overline{u_F}$ is the average vector of the user's friends, v is a vector of a job, F is the set friends of the active user, u^f is the vector of the friend f and $\alpha \in [0, 1]$ is the importance of user's data.

In Engine-4a, Engine-4b and Engine-4c, the TF-IDF vectors of users and jobs are obtained by using the importance (α^{0*}, α^{1*}, β^*) of different fields computed in the Sect. 4.3.

3.7 Relevance feedback: Engine-5

We also used a slightly modified version of the famous Rocchio's formula (Rocchio 1971) to enrich our users' vectors with the vectors of jobs linked (relevant and non relevant jobs) to them:

$$u' = \max(0, (a.u + b\overline{u_J^+} - c\overline{u_J^-})) \tag{26}$$

where u' is the enriched vector of the active user, u is the original vector of the user, $\overline{u_J^+}$ is the average vector of jobs that match the profile of the user, $\overline{u_J^-}$ is the average vector of jobs that do not match the profile of the user and a, b, c are the parameters of the algorithm. Here the TF-IDF vectors of users and jobs are also obtained by using the importance (α^{0*}, α^{1*}, β^*) of different fields computed in the Sect. 4.3.

4 Results

This section presents the experiments that we conducted while developing the proposed recommender systems (see

Sects. 3.2, 3.3, 3.4, 3.5, 3.6 and 3.7). The first part is dedicated to the description of our datasets while the second part presents the importance of different fields and the results of different proposed recommender systems.

We use AUC-ROC (see Sect. 2.4) as performance metric for our different experiments and we compute 95% empirical confidence intervals [(2.5 %quantile, 97.5 % quantile)] using bootstrapping (100 runs per experiment).

4.1 Description of datasets

We evaluate the performance of our job recommender systems (see Sects. 3.2, 3.4, 3.5, 3.6 and 3.7) on 6 datasets collected by the company Work4. Each entry in our 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; label 1 denotes a matching between the user and the job while the label is 0 when the job does not correspond to the user. Here are the descriptions of the 6 collected datasets:

1. Candidate: users can use Work4's applications to apply to jobs, we assume that users only apply to the jobs that are relevant for them (label =1); this dataset contains applications' data.
2. Feedback: contains the feedback from users of Work4's applications.
3. Random: this dataset contains couples of users–jobs that have been randomly drawn from Work4's databases and manually annotated.
4. Review: it contains recommendations made by Work4's systems that have been manually validated by two different teams of the company.
5. Validation: it contains recommendations made by Work4's systems that have been manually validated by only one team of the company.
6. ALL: a sixth dataset has been artificially created: it is the union of the 5 previous datasets.

As explained above, the entries in Review and Validation datasets are obtained by manually annotating some recommendations made by our systems. To annotate a recommendation of a job to a user, the annotators have all the available information about the job (title, company, industry, ...) and the information the user has explicitly authorized our applications to access to (education background, work history, age, etc.) and they can annotate the recommendation as follows:

- 1: the user matches the job,
- 0: the user does not match the job; in this case, they can justify their decision by:
  - ∘ experience mismatch: user's and job's required experience do not match,
  - ∘ languages mismatch: user's and job's languages do not match,
  - ∘ countries mismatch: user's and job's countries do not match.

  Recommending an internship job to a person who has currently a full-time position or who has been graduated for years is generally considered by the annotators as an experience mismatch.
- -1: cannot decide if the user matches the job or not; these entries are not used in this study.

The proposed Engines (see Sect. 3) in this study cannot detect some of these mismatches like experience mismatch but Work4 is using advanced versions of Engines that can detect these subtle mismatches.

We assumed in Candidate dataset that users only apply to jobs that are relevant for them; this assumption could be false in some situations. The following examples show some situations in which our assumption is not correct:

- people who are actively looking for a job can apply to several jobs at same time even if they do not match their profiles.
- people who are changing careers can apply to jobs that do not match their profiles.

It is worth noting that each job is associated to a job page which generally represents a page of a company (in this context, the pages are sets of job offers published by the same company). Hence jobs from the same job page are generally similar since they are likely from the same company. Table 1 shows the basic statistics from our datasets while Table 2 shows the percentage of empty fields in each dataset. We can notice that most social networks users do not fill completely the corresponding forms; this problem is more severe for Facebook than LinkedIn profiles. Our recommender systems must thus deal with incomplete (and very noisy) data.

After filtering out stop words using lists defined by Work4, we obtained a dictionary with 218,533 terms (English terms, French terms and other languages' terms); we focus on English and French. We use the TF-IDF scores to select the most interesting terms in users' and jobs' profiles as done in Wang and Blei (2011); Blei and Lafferty (2009); Figure 5 shows the cumulative weights of users' and jobs' terms.

4.2 Engine-1

Figure 6 shows the results of Engine-1 on different datasets for all users, Facebook users and LinkedIn users; we can notice that the results on Feedback dataset are different from the others but not significant (see the associated confidence intervals). We also notice that results are bad (AUC <0.5) for Facebook users and all users on ALL

Table 1 Basic statistics from our datasets: number of instances, proportion of label 0/1 and instances linked to Facebook/LinkedIn users

	Datasets					
	ALL	Candidate	Feedback	Random	Review	Validation
Total number of instances	203,990	30,179	246	2,898	32,441	138,226
Proportion						
Label 0	0.79	0.00	0.35	0.99	0.76	0.96
Label 1	0.21	1.00	0.65	0.01	0.24	0.04
Facebook	0.38	0.97	0.25	0.76	0.33	0.25
LinkedIn	0.62	0.03	0.75	0.24	0.67	0.75

ALL dataset is the union of the 5 other datasets. We assumed that users only applied to jobs that match their profiles in Candidate dataset: this dataset contains only labels 1, so it cannot be directly used for the AUC metric

Table 2 Percentage of empty fields in our datasets; in bold, fields empty at more than 50 %

Fields	Datasets					
	ALL	Candidate	Feedback	Random	Review	Validation
Facebook						
Bio	**66.5**	**81.3**	**61.8**	**85.0**	**69.4**	48.4
Education	**81.6**	**88.2**	**52.7**	**81.4**	**78.3**	**73.9**
Interests	**76.9**	**79.1**	**90.9**	**94.5**	**79.7**	**72.7**
Quotes	**93.6**	**87.7**	**92.7**	**95.9**	**97.5**	**98.8**
Work	**60.8**	**73.3**	23.6	**80.1**	45.5	46.2
LinkedIn						
Educations	14.1	12.9	8.4	12.5	13.1	14.0
Headline	0.3	1.6	0.0	0.3	0.4	0.2
Positions	0.2	3.8	0.0	1.0	0.2	0.1
Jobs						
Description	0.0	0.0	0.0	0.5	0.0	0.0
Responsibilities	**57.6**	**52.8**	**78.6**	**74.7**	**65.4**	**82.0**
Title	0.0	0.0	0.0	0.0	0.0	0.1

Our Facebook users do not completely fill the fields that are interesting for our systems. It is worth noting that the percentage of empty *Responsibilities* fields for jobs is very high, this due to the fact that this field information is sometimes merged with the *Description* field one

dataset; these bad results are due to the fact that users' terms space is not exactly equal to jobs' terms space despite the fact that we selected the most interesting sub-fields for social networks users. The Sect. 5 explains in details this out-performance. If we consider the AUC-ROC scores for Facebook users and LinkedIn users separately, we notice that we have better results for LinkedIn users (see Fig. 6): LinkedIn users' terms space seems closer to jobs' terms space than Facebook one.

4.3 Importance of fields

To compute the importance of users and jobs fields on the recommendation scores, we solve the constrained optimization problem stated in the Sect. 3.3 using the function *minimize* (from Scipy.optimize module) with the method SLSQP (Sequential Least Squares Programming). We set the costs c_0 and c_1 as in Anand et al. (2010): $c_0 = \frac{1}{n_0}$ and $c_1 = \frac{1}{n_1}$ where n_0 and n_1 are the number of entries with label 0 and label 1 in the training sample, respectively.

Figure 7 shows the importance of Facebook users', LinkedIn users' and jobs' fields with their confidence intervals, respectively. It also suggests that the important fields in the task of job recommendation are:

- *Work* field for Facebook users;
- *Headline* field for LinkedIn users;
- *Title* field for jobs.

These results seem to make sense since the field *work* contains useful information to determine the interests of Facebook users for jobs, the field *Headline* sums up LinkedIn users careers and the field *title* contains needed

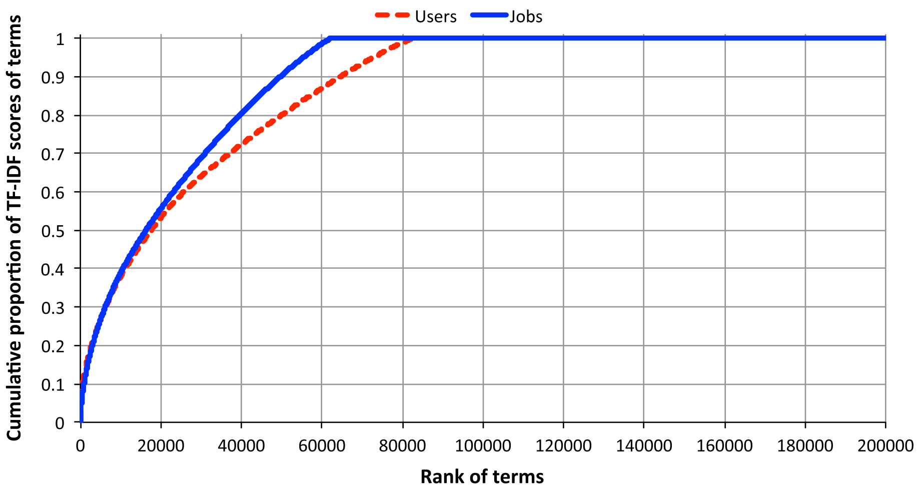


Fig. 5 Cumulative TF–IDF weights of users' and jobs' terms; this figure is used to select the most informative terms using TF–IDF as informativeness criterion

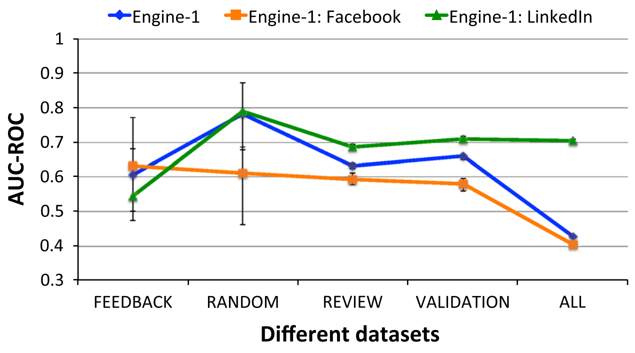


Fig. 6 Performance (AUC-ROC) of Engine-1 for Facebook users, LinkedIn users and all the users. We obtain an AUC <0.5 on ALL dataset (which includes entries in Candidate dataset) for Facebook and all users, this is due to the poor performance of Engine-1 in predicting applications of users to jobs (see Sect. 5 for further details). For AUCs below 0.5, we can inverse all the predictions to obtain AUCs >0.5

information about a given job to globally determine if it is relevant or not.

4.4 Comparison between Engine-1 and Engine-2

Figure 8 depicts the performance of Engine-2 for Facebook and LinkedIn users; we note that we obtain higher performance for LinkedIn users than for Facebook users as in Engine-1 (see Sect. 4.2).

We compare Engine-1 to Engine-2 to see if the application of the importance of users' and jobs' fields improves or not our results. Figure 9 shows that the application of the importance of fields (computed in Sect. 4.3) significantly improves our results on all of our datasets except the Random dataset, but the results on this dataset are not significant (see the associated confidence intervals). The performance of Engine-2 is better than Engine-1's one but is not still sufficient for us.

4.5 Social recommender systems: Engine-4a, Engine-4b, Engine-4c

We compare our 3 basic social recommender systems to Engine-2: we vary the importance α of users' data from 0 to 1 to see how this variation impacts the results. Table 3 shows that all the proposed social recommender systems fail to give better results than Engine-2 for any value of $\alpha < 1$; these results suggest that not all the friends of a social network user have the same preferences for jobs as him; to improve recommendations, we should only use the data of friends that are similar to the active user, not the data of all his friends. The main issue in our context is the fact that we cannot accurately compute similarity between users (whose profiles are almost empty) and their friends.

4.6 Application of the relevance feedback: Engine-5

To compare Engine-2 to Engine-5 which uses the relevance feedback, we vary the proportion of datasets to be used as feedback sets from 0 to 0.6; for each user in feedback sets, we enrich his vector with the vectors of his linked jobs as explained in Sect. 3.7; we also set $a = b = c = 1$. Table 4 shows that the use relevance feedback drastically improves our results; this is very interesting and shows that if we have a sufficient set of feedback for a user (jobs that match or not the user), we can make accurate job recommendations to him. Unfortunately this has currently no direct application at Work4 since our users generally do not give any feedback about the jobs we recommend to them.

4.7 Comparison between Engine-2 and Engine-3

To learn our SVM linear model, we set the costs of different classes c_0 and c_1 as previously (see Sect. 4.3). In our context, the ideal procedure of splitting our datasets into

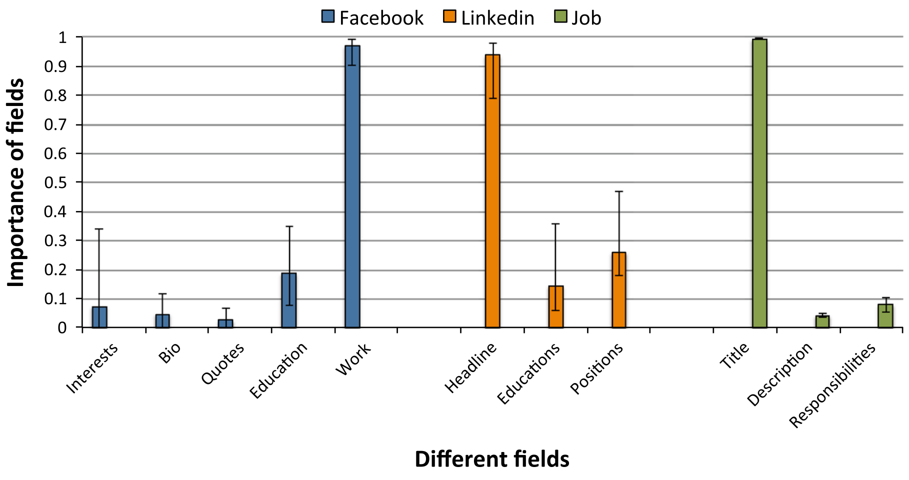


Fig. 7 ALL dataset—Importance $\alpha^0 \in [-1, +1]^5$, $\alpha^1 \in [-1, +1]^3$ $\beta \in [-1, +1]^3$ of Facebook users, LinkedIn users and jobs fields, respectively; the higher the importance is, the most important the associated field is in the task job recommendation

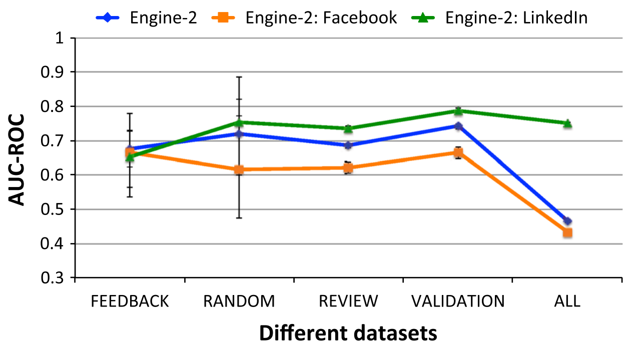


Fig. 8 Performance (AUC-ROC) of Engine-2 for Facebook users, LinkedIn users and all the users. We obtain an AUC <0.5 on ALL dataset (which includes entries in Candidate dataset) for Facebook and all users, this is due to the poor performance of Engine-2 in predicting applications of users to jobs (see Sect. 5 for further details). For AUCs below 0.5, we can inverse all the predictions to obtain AUCs >0.5

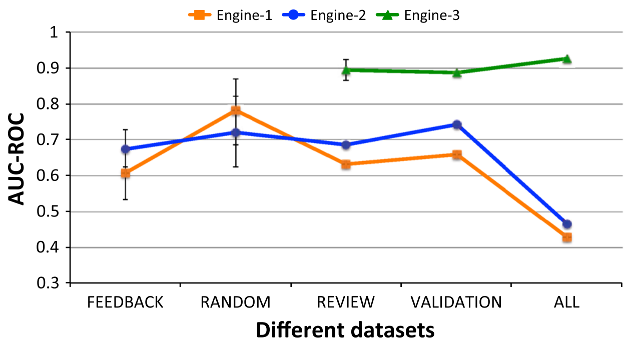


Fig. 9 Comparison between Engine-1, Engine-2 and Engine-3 for each of our datasets; for Engine-3, we use a tenfold cross-validation on the 3 biggest datasets (review, validation and ALL) due to the fact that the two smallest datasets have not enough entries to obtain significant results using machine learning

training sets and test sets could be: learn a model from labeled data linked to a set job pages and then apply the learnt model to make recommendations on new job pages; this involves many varied job pages in the used training sets but unfortunately we have several huge job pages (job pages linked to many entries) in our datasets that makes difficult the use of this procedure. If we split a dataset into a training and test sets using the above procedure of splitting, we have 2 scenarios (due to the huge job pages in our datasets):

1. the training set contains exclusively huge job pages: we learn a model on few pages that cannot yield good results on new pages;
2. the training set contains exclusively pages with few linked examples: the learnt model cannot also yield good on specific huge pages.

Finally, we use an alternative method based the tenfold cross-validation: we randomly split the active dataset into 10 disjoint sets making sure that each set contains about 10 % of each job page; for each fold, we use 1 set as test set and 9 sets as training set. This procedure of splitting datasets into training and test sets could bias the results since jobs from the same page are similar but has two applications at Work4:

1. learn a global model (what we do in this paper) using all our datasets that we will use for new clients to make first recommendations; we can notice that this global model could yield bad results for some clients with specific job pages; for instance it will probably make bad recommendations for clients whose categories of jobs were not in the datasets used to learn the model.
2. learn a local model for each client that has enough feedback (labeled data from the client's teams or from Work4's teams) to learn a model; this local model will neatly fit the client data and will make better recommendations than the global model for client.

Figure 9 shows the comparison between Engine-2 and Engine-3 using tenfold cross-validation on our 3 biggest datasets (ALL, Validation and Review). We can see that

Table 3 Social recommendation: comparison between Engine-2, Engine-4a, Engine-4b and Engine-4c in terms of AUC; for a reminder, users' social vectors/scores = α users vector + $(1-\alpha)$ users' friends vectors/scores

Methods	Importance α of users' data (from 0 to 1)					
	0	0.2	0.4	0.6	0.8	1.0
ALL						
Engine-2	**0.47 ± 0.01**					
Engine-4a	0.30 ± 0.01	0.38 ± 0.01	0.39 ± 0.01	0.41 ± 0.01	0.43 ± 0.01	**0.47 ± 0.01**
Engine-4b	0.30 ± 0.01	0.33 ± 0.01	0.37 ± 0.01	0.40 ± 0.01	0.43 ± 0.01	**0.47 ± 0.01**
Engine-4c	0.30 ± 0.01	0.34 ± 0.01	0.37 ± 0.01	0.41 ± 0.01	0.43 ± 0.01	**0.47 ± 0.01**
Validation						
Engine-2	**0.74 ± 0.01**					
Engine-4a	0.53 ± 0.01	0.63 ± 0.02	0.67 ± 0.01	0.70 ± 0.01	0.73 ± 0.02	**0.74 ± 0.01**
Engine-4b	0.53 ± 0.01	0.59 ± 0.02	0.65 ± 0.01	0.71 ± 0.01	0.74 ± 0.02	**0.74 ± 0.01**
Engine-4c	0.54 ± 0.02	0.61 ± 0.01	0.67 ± 0.01	0.72 ± 0.02	**0.74 ± 0.01**	**0.74 ± 0.01**
Review						
Engine-2	**0.69 ± 0.01**					
Engine-4a	0.52 ± 0.01	0.61 ± 0.01	0.63 ± 0.01	0.66 ± 0.01	0.68 ± 0.01	**0.69 ± 0.01**
Engine-4b	0.52 ± 0.01	0.57 ± 0.01	0.61 ± 0.01	0.65 ± 0.01	0.67 ± 0.01	**0.69 ± 0.01**
Engine-4c	0.53 ± 0.01	0.59 ± 0.01	0.63 ± 0.01	0.66 ± 0.01	0.67 ± 0.01	**0.69 ± 0.01**

We can note that we obtain the highest AUCs for $\alpha = 1$ (no use of social data) for social recommendation methods. Section 5 explains in details why some AUCs are below 0.5

Table 4 Relevance Feedback: comparison between Engine-2 and Engine-5 in terms of AUC

Methods	Proportion α of the datasets used as Feedback sets			
	0	0.2	0.4	0.6
ALL				
Engine-2	0.47 ± **0.01**			
Engine-5	0.47 ± 0.01	**0.92 ± 0.01**	**0.95 ± 0.01**	**0.97 ± 0.00**
Validation				
Engine-2	0.74 ± **0.01**			
Engine-5	0.74 ± 0.01	**0.95 ± 0.01**	**0.98 ± 0.01**	**0.99 ± 0.00**
Review				
Engine-2	0.69 ± **0.01**			
Engine-5	0.69 ± 0.01	**0.95 ± 0.01**	**0.97 ± 0.01**	**0.99 ± 0.01**

We can note that the use of relevance feedback drastically increases the performance of our recommendation engines

Engine-3 outperforms Engine-2 on these datasets; these results show that it is possible to learn from our data a similarity function that yields better results than cosine similarity.

4.8 Comparison between Engine-2 and Engine-3 and two methods of the literature

In this section we compare Engine-2 and Engine-3 to two methods of the literature. The first method is a simple Collaborative Filtering (CF) based on matrix factorization and the second method is the Collaborative Topic Regression (CTR) proposed by Wang and Blei (2011). We use the code provided by Wang and Blei (2011) to compare our methods to CF and CTR. For the CTR, we use the 25,000 terms for users and 25,000 terms for jobs with the higher TF-IDF weights (this represents more than 60% of total inertia for users and jobs; see Fig. 5). Here are the parameters we use for CTR and CF:

- the confidence parameter for the label 1: $a = 1$;
- the confidence parameter for the label 0: $b = 0.01$ (the label 0 can be interpreted into two ways: don't match or don't know);
- the number of latent dimensions num_factors = 50;
- the max number of iterations max_iter = 50.

Table 5 Comparison between Engine-2, Engine-3, CF and CTR. CF and CTR are the methods proposed and implemented by Wang and Blei (2011) using tenfold cross-validation

Methods	Different datasets		
	ALL	Validation	Review
Engine-2	0.47 ± 0.01	0.74 ± 0.01	0.69 ± 0.01
Engine-3	**0.93 ± 0.01**	**0.89 ± 0.01**	**0.89 ± 0.01**
CF	0.67 ± 0.03	0.73 ± 0.03	0.84 ± 0.03
CTR	0.80 ± 0.02	0.78 ± 0.04	0.88 ± 0.01

Table 5 shows that Engine-3 outperforms CTR on our three biggest datasets. Not surprisingly we obtain better results for CTR than for CF: CTR is an improvement of CF (Wang and Blei 2011); we can also note that CF outperforms our Engine-2. The main weakness of CTR is the fact that it only uses the data of relevant jobs for users while our Engine-3 use both the data of relevant and non relevant jobs for users.

5 Discussion and future work

We conducted many experiments on real-world data provided by Work4 in order to test and improve our proposed recommender systems. We used AUC as performance metric for our different Engines.

The first series of experiments concludes that Facebook users and LinkedIn users seem to have a vocabulary (set of terms in their profiles) a bit different from the vocabulary of jobs descriptions. However, LinkedIn users vocabulary seems closer to the vocabulary of jobs than Facebook one (see Fig. 6). On ALL dataset, some Engines obtain an AUC below 0.5, this could be explained by the combination of the following facts:

- ALL dataset includes all entries in Candidate dataset [applications of users to jobs (label=1)],
- 97 % of entries in Candidate dataset is coming from Facebook users (see Table 1),
- Users' vocabulary is quite different from the vocabulary of jobs (this problem is more severe for Facebook than LinkedIn profiles),
- and Facebook users' profiles are less filled than LinkedIn ones (see Table 2).

To sum up, the poor performance of some Engines in predicting applications of users to jobs (entries with label 1) leads them to obtain an AUC below 0.5.

Not surprisingly, we find out that the most relevant fields for job recommendations are *Work* (for Facebook user), *Headline* (for LinkedIn users) and the *Title* for jobs. 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 annotators of Work4 manually reviewed entries in Review and Validation datasets (see Sect. 4.1); they probably used information in these fields to make their reviews.

The application of the importance of different fields (see Fig. 7) (Engine-2) significantly improves the results (compared to Engine-1), but the results could be improved.

Unfortunately, the use of basic methods of social recommendation (Engine-4) failed to improve our results (compared to Engine-2) and we could not use complex methods of social recommendation due to the nature of our data. However, the use of relevance feedback drastically improves the results; this is very interesting and shows that we can improve the performance of heuristic-based job recommender systems by enriching users' profiles using their feedback.

Our linear SVM-based recommender system (Engine-3) outperforms the cosine-based methods (Engine-1 and Engine-2); it also outperforms the CTR and CF proposed by Wang and Blei (2011).

We have assumed in this paper that users only apply to jobs relevant for them (in Candidate dataset), this assumption could be partially false for many reasons (users were testing Work4 applications when applied to jobs, etc.); we will try to precisely measure the validity of this assumption in our future work.

Users' term space seems to be not completely equal to the jobs' term space, to solve this issue, we recently used an ontology, O*NET-SOC taxonomy[2] (that defines the set of occupations across the world of work) to develop a new taxonomy-based vector model for social networks users and jobs' descriptions suited to the task of job recommendation.

We are applying this vector model to predict the audience of jobs posted by social network users. We are also developing a much more sophisticated vector model for social networks users and jobs that is based on ontologies and will contains users' experience/experience required by jobs for different occupations, their keywords and their related occupations. In this model, we will be using stemming, lemmatization and new lists of stop words to better preprocess users' and jobs' textual documents.[3]

Acknowledgments This work is supported by Work4, ANRT[3] (the French National Research and Technology Association), the French FUI Project AMMICO and the project Open Food System. The authors thank Guillaume Leseur, software architect at Work4, Benjamin Combourieu and all the Work4*Engines* team. We thank all the

[2] http://www.onetcenter.org/taxonomy.html.

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

anonymous reviewers who spent their time and energy reviewing this paper, thanks for your reviews and helpful comments and remarks.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6): 734–749

Aiolli F (2013) Efficient top-n recommendation for very large scale binary rated datasets. In: Proceedings of the 7th ACM Conference on Recommender Systems, ser. RecSys '13. ACM, New York, NY, USA, pp 273–280.

Anand A, Pugalenthi G, Fogel GB, Suganthan PN (2010) An approach for classification of highly imbalanced data using weighting and undersampling. Amino Acids 39(5):1385–91

Aranda J, Givoni I, Handcock J, Tarlow D (2007) An online social network-based recommendation system. Department of Computer Science-University of Toronto, Computer Sciences Technical Report

Balabanovic M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Comm. ACM 40(3):66–72

Baeza-Yates RA, Berthier R-N (1999) Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA

Bennett J, Lanning S, Netflix N (2007) The netflix prize. In: In KDD Cup and Workshop in conjunction with KDD

Blei DM, Lafferty JD (2009) Topic models. In: Srivastava AN, Sahami M (Eds) CRC Press

Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl Based Systems 46:109–132

Boyd DM, Ellison NB (2008) Social network sites: definition, history, and scholarship. J Comput Mediat Commun 13(1):210–230

Brain S (2014). Available at http://www.statisticbrain.com/twitter-statistics/

Chang C-C, Lin C-J (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol 2(3), pp 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm

Claypool M, Gokhale A, Miranda T, Murnikov P, Netes D, Sartin M (1999) Combining content-based and collaborative filters in an online newspaper

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297

Cover TM (1965) Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. Electron Comput IEEE Trans, vol. EC-14, no. 3, pp 326–334

de Campos LM, Fernández-Luna JM, Huete JF, Rueda-Morales MA (2010) Combining content-based and collaborative recommendations: a hybrid approach based on bayesian networks. Int J Approx Reason 51(7):785–799

Diaby M, Viennet E, Launay T (2013) Toward the next generation of recruitment tools: an online social network-based job recommender system. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining ASONAM 2013, pp 821–828.

Facebook (2014) Available at http://newsroom.fb.com/company-info/

Gao F, Xing C, Du X, Wang S (2007) Personalized service system based on hybrid filtering for digital library. Tsinghua Sci Technol 12(1):1–8

Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In: Proceedings of the 4th International Conference on Trust Management, ser. iTrust'06. Springer, Berlin, Heidelberg, pp 93–104.

Groh G, Ehmig C (2007) Recommendations in taste related domains: collaborative filtering vs. social filtering. In: Proceedings of the 2007 International ACM Conference on Supporting Group Work, ser. GROUP '07. ACM, New York, NY, USA, pp 127–136.

Han J (1996) Data mining techniques. SIGMOD Rec 25(2): 545

Jannach D, Zanker M, Felfernig A, Friedrich G (2011) Recommender systems: an introduction. Cambridge University Press, Cambridge

Joachims T (1998) Text categorization with suport vector machines: learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning, ser. ECML '98. Springer, London, UK, pp 137–142.

Kantor PB (2009) Recommender systems handbook. Springer, New York; London

Kazienko P, Musiał K, Kajdanowicz T (2011) Multidimensional social network in the social recommender system, vol. 41, no. 4, pp. 746–759

Lemire D, Maclachlan A (2005) Slope one predictors for online rating-based collaborative filtering. In: Proceedings of SIAM Data Mining (SDM'05)

Linden G, Smith B, York J (2003) Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80

LinkedIn (2014). Available at http://press.linkedin.com/about

Lops P, de Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor PB (Eds) Recommender systems handbook. Springer, Berlin, pp 73–105.

Ma H, Zhou D, Liu C, Lyu MR, King I (2011) Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on Web search and data mining, ser. WSDM '11. ACM, New York, NY, USA, pp 287–296.

Nicholas ISC, Nicholas CK (1999) Combining content and collaboration in text filtering. In: Proceedings of the IJCAI 99 Workshop on machine learning for information filtering, pp 86–91.

Omary Z, Mtenzi F (2010) Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. Int J Infon (IJI) 3

Pazzani MJ, Billsus D (2007) The adaptive web. In: Brusilovsky P, Kobsa A, Nejdl W (Eds) Content-based Recommendation Systems. Springer, Berlin, Heidelberg, pp 325–341.

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

Ravikumar P, Tewari A, Yang E (2011) On ndcg consistency of listwise ranking methods. Available at http://www.cs.utexas.edu/users/ai-lab/?RTY11

Rijsbergen CJV (1979) Information Retrieval, 2nd edn. Butterworth-Heinemann, Newton, MA, USA

Rocchio JJ (1971) Relevance feedback in information retrieval. In: Salton G (Ed) The SMART retrieval system: experiments in automatic document processing, ser. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs NJ, ch. 14, pp 313–323.

Salakhutdinov R, Mnih A (2008a) Bayesian probabilistic matrix factorization using markov chain monte carlo

Salakhutdinov R, Mnih A (2008b) Probabilistic matrix factorization

Salton G, Wang A, Yang C (1975) A vector space model for information retrieval. J Am Soc Inf Sci 18(11):613–620

Séguela J (2012) 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

Shardanand U, Maes P (1995) Social information filtering: algorithms for automating “word of mouth”. In: Proceedings of the SIGCHI Conference on human factors in computing systems, ser. CHI ’95. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp 210–217.

Wang C, Blei DM (2011) 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, pp 448–456.

Xiao B, Benbasat I (2007) 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, pp 137–209.

Mamadou Diaby is currently a Ph.D. candidate at the Laboratoire de Traitement et Transport de l’Information (L2TI) of Université Paris 13 (France) and an R&D engineer at Work4, the Global Leader in Social and Mobile Recruiting. He is working on the Engines project to improve the performance of Work4’s recommender systems.

Emmanuel Viennet received his Ph.D. in computer science in 1992 from Universtité Paris 11 (France). He is currently a professor at Université Paris 13 and the Ph.D. supervisor of Mamadou Diaby. His research focuses on data mining methods for social networks and multimedia data analysis. He collaborates with the company Work4 on the Engines project, an R&D project which aims at developing the next generation of recruitment tools in social networks.

Tristan Launay received his Ph.D. in statistics and Bayesian methods in 2012 from Université de Nantes (France). He worked as an R&D engineer and a statistician on the Engines project at Work4.