2017 International Conference on Tools with Artificial Intelligence

Language Modelling for Collaborative Filtering: Application to Job Applicant Matching

Thomas Schmitt*‡, François Gonard†*‡, Philippe Caillou*, Michele Sebag*
*LRI, CNRS–INRIA–Univ. Paris-Sud, Université Paris-Saclay 91405 Orsay Cedex, France
†IRT SystemX, 8 avenue de la Vauve 91127 Palaiseau Cedex, France
Email: {firstname.lastname}@inria.fr
‡Equal contributions for first and second authors

***Abstract*—This paper addresses a collaborative retrieval problem, the recommendation of job ads to applicants. Specifically, two proprietary databases are considered. The first one focuses on the context of unskilled low-paid jobs/applicants; the second one focuses on highly qualified jobs/applicants. Each database includes the job ads and applicant resumes together with the collaborative filtering data recording the applicant clicks on job ads.**

The proposed approach, called LAJAM, focuses on the semi-cold start recommendation problem of recommending new job ads to known applicants. This setting is relevant to the temporary job sector, of increasing importance for current job markets. LAJAM learns a continuous language model on the job ad space, trained to comply with the collaborative filtering metrics. This language model, implemented as a neural net, can flexibly take into account heterogeneous additional information, e.g. related to the posting time and geolocation of the job ads.

The merits of the LAJAM approach are demonstrated comparatively to the state of the art on the thoroughly studied CiteULike database. The comparison of the public CiteULike database with the proprietary databases sheds some light on the specific difficulties of the job applicant matching problem.

I. INTRODUCTION

As many economic sectors have been impacted by data science, ranging from commerce to tourism and from travel to education, a question is whether and how the field of employment can also benefit from the conjunction of extensive data resources and machine learning algorithms. This paper focuses on the *frictional unemployment* phenomenon, manifested as significant numbers of job positions are unfilled, although there exist a significant number of unemployed people who are qualified for these jobs [1]. Frictional unemployment is tentatively explained from the cost and asymmetry of information for recruiters and applicants. The information issue at the root of frictional unemployment is tackled in this paper, aimed to automatically match job ads and resumes.

Several approaches have been proposed to facilitate the recruitment task, using diverse types of data to describe applicants: curriculum vitae [2], track record of past jobs [3] and/or profiles in social networks [4]. However, such resources might be poorly informative for those who need them the most. Unemployment is known to mainly affect both categories of unskilled people, and young people. In the category of unskilled young people, the unemployment rate is close to 40% (as of France in 2015). Yet, these people have a resume with moderately informative content, neither including any information about academic degrees or diplomas, nor on past jobs and track records. A job/applicant matcher, or JAM in the following, thus finds little signal in these resumes. When considering young highly skilled people, a JAM faces another kind of difficulty: an applicant might describe her expertise using rare words (e.g. the title of her PhD) whereas a job advertisement might describe the sought skills using an entirely different vocabulary[1]. Domain resources such as field-based ontologies can be used to enrich the documents and facilitate their matching, though they face some limitations due to the rapid pace of change in science and technology, and the emergence of new jobs.

This paper aims to learn a JAM, automatically recommending job ads to applicants. While the recommendation of job ads to applicants and the recommendation of resumes to recruiters are formally equivalent problems, both problems are different in practice and only the former recommendation problem is considered in the remainder of the paper. A specificity of the study is to consider two large-size proprietary databases. The first database concerns low-paid jobs and unskilled people. Indeed the low-paid job sector is less profitable than the high-paid job sector, and has been underlooked in the literature [2], [5], [6]; it also presents specific difficulties due to the low signal-to-noise in the resumes. The second dataset considers PhD-level resumes and industrial job ads targeted at PhDs. Each database includes the resume and job ad corpora, augmented with the collaborative filtering matrix reporting applicant clicks on job ads.

Two scientific questions are investigated in this paper. The first question is whether the relevance of a job ad to an applicant can only be determined from the (hidden) preferences of the applicant – in other words, whether job/resume matching is a pure collaborative filtering task [7], [8]. A second possibility will be investigated, suggesting that job/resume

T. Schmitt's PhD is funded by the Lidex *Institut de la Société Numérique* at Université Paris-Saclay. F. Gonard's PhD is funded by the French Program *Investissements d'Avenir*.

[1]For instance, a PhD title might read *Novel Copolyimide Membranes for pervaporation to be applied in the separation of aromatic/aliphatic mixtures* while a job ad relevant to the PhD expertise might ask for a researcher on *comparative genomics of different microorganisms in order to identify specific genes and islands involved in horizontal transfer.*

2375-0197/17/31.00 ©2017 IEEE
DOI 10.1109/ICTAI.2017.00186

matching should rather be viewed as a *machine translation* task: despite their different formulations, the job ads relevant to an applicant are said to be consistent if one can learn a metric on the job ad space accounting for the collaborative filtering matrix. The main contribution presented in this paper is a JAM system called LAJAM for *Language Model-based* JAM, that learns an *ad hoc* continuous language model on the top of the job ad space. This continuous language model, implemented as a neural net and trained to comply with the collaborative filtering matrix, brings some answers to the above questions.

The paper is organized as follows. Section II briefly reviews and discusses related work, considering the main three aspects relevant to the JAM problem: natural language processing, recommendation systems and metric learning. Section III describes the considered proprietary databases, the experiments conducted to assess their quality and the difficulty of the matching problem. Section IV presents a detailed overview of the proposed LAJAM system. Section V describes the experimental setting used to validate the approach. Section VI reports on LAJAM experimental results comparatively to state-of-art approaches [8]. The paper concludes with a discussion and some perspectives for further research.

II. Related Work

The field of automatic job/applicant matching (JAM) is hindered by the lack of extensive public resources, able to foster algorithm design and assessment on a grand scale, as was done for the field of computer vision by, e.g., the ImageNet dataset [9]. This lack of publicly available data resources is explained from the privacy requirements, and the difficulty of resume anonymization. To our best knowledge, the only large scale public JAM databases were proposed in the frame of the RecSys 2016 and RecSys 2017 challenges. The RecSys 2016 (respectively 2017) database includes over 1 million job ads (resp. 1.3 million), 780,000 resumes (resp. 1.4 million), and their interactions available for circa .01% of the resume/job pairs (resp. .001%) [6]. The anonymization was enforced through encoding resumes and job ads using 90,000 binary features (replacing raw text by IDs), besides perturbing the data by adding and removing clicks.

Early related work was rooted on Natural Language Processing (NLP), considering manually designed features [2] or bag-of-words representation [5], possibly enriched using domain resources such as job ontologies. On top of these representations, a similarity was designed or learned to estimate the relevance of a resume w.r.t. a job ad, possibly taking into account the resume structure through weighted similarities on the resume subparts [5].

The JAM problem features specific characteristics compared to "pure" NLP problems such as Information Retrieval (IR) [10] and Recognizing Textual Entailment (RTE) [11]. Indeed, a JAM could learn whether a resume is "relevant" to a job query, or whether the skills required by a job can be "entailed" by a resume; however, the resume and the job ad are of similar size, whereas a document is usually much more detailed than a query in IR (resp., than a statement in RTE). Another difference is that the notion of relevance or entailment could in principle be logically inferred. Quite the contrary, the match between a user and a recruiter depends on hidden information (e.g. the applicant's and recruiter's positive or negative biases). In other words, what people do (clicking on job ads or resumes) can hardly be logically inferred from what they say (the contents of the resumes and job ads).

The potential importance of individual preferences suggests that a JAM can better be handled in terms of recommendation and collaborative filtering [7]. Recommender systems come in three modes:

The most usual mode, referred to as **warm-start recommendation** and illustrated by the Netflix challenge [12], aims to recommend known items to known users. Note that the warm-start recommendation mode is hardly relevant to the JAM context, as an item (job ad) is no longer available after the position has been fulfilled.

The **full-cold start recommendation** mode aims to recommend new items to new users; this mode is the most usual one in the JAM context, where new applicants are looking for newly opened positions.

Finally, the **semi-cold start recommendation** mode aims to recommend new items to known users. This mode is relevant to the JAM context in the case of the temporary work sector, where applicants interact with the hiring agency on a regular basis and the current recommendations exploit the past interactions.

Full- and semi-cold start recommendation[2] are made possible by the side information describing the users and the items. In the JAM context, a job position (respectively an applicant) is associated with a job ad (resp., a resume) usually augmented with a job posting date and geolocation. The JAM problem thus defines a *collaborative retrieval* task [14].

As far as JAM tackles (semi)-cold start recommendation, approaches relying exclusively on the collaborative filtering matrix, e.g. based on matrix factorization [7], [14], on restricted Boltzman machines [15], or learning a continuous embedding of the items [13] are not applicable. Some intermediate representation capturing both the collaborative filtering information and the user/item description must thus be defined. In early approaches [16], [2], manually designed text features are used to recode the collaborative filtering matrix, defining the probability of a resume to be clicked upon by a recruiter, conditionally to the presence of such features in respectively the resume and the job ad. This more general representation of the collaborative filtering matrix thus allows one to both take into account the textual content-based features, and address the cold-start recommendation problem.

Overall, JAM systems involve two issues: finding representation(s) for resumes and/or job ads, and a distance on top of these representations, amenable to predict whether a (resume, job ad) pair is well-suited to each other. Related works along

[2]For some authors [13], "cold start" means that "some users come with a limited interaction history". In the remainder of this paper, semi-cold start refers to the recommendation of brand new items to known users.

these directions build upon the continuous language modelling, mapping words [17], [18], sentences or documents [19] onto vectors. A distance between documents follows from the NLP embedding into a Euclidean space, or can be defined using Word Movers Distance [20]. Such continuous embeddings can thus be used for (semi) cold start recommendation [21]. Another possibility builds upon supervised learning and ranking [22], [23], where the metric on the Euclidean space is optimized to maximize a classification or ranking criterion. Notably, Siamese neural networks [24], [25] optimize the input embedding in a continuous space w.r.t. the associated metric [26], [27]. Such approaches have been used to achieve job title normalization and classification [28] or learning sentence similarity [29]. In Information Retrieval, Deep Structured Semantic Mode (DSSM) optimizes the similarity in the latent space according to clickthrough data [30] (see also [31]). Multi-view DNN [32] further extends DSSM to non-Siamese architectures, where different types of information involved in the query/documents are associated with different embeddings.

A related approach is that of collaborative topic regression (CTR) [8]. For the sake of self-containedness, CTR is briefly described as it will be used as baseline method in the experiments (Section VI). The motivating application concerns the recommendation of scientific articles in an online scientific community framework; the collaborative filtering matrix indicates whether a user's library contains a given article. CTR combines ideas from collaborative filtering based on latent factor models, and content analysis based on probabilistic topic modelling and specifically Latent Dirichlet Allocation (LDA) [33]. Each LDA topic (e.g., "artificial intelligence") is characterized as a distribution on the vocabulary space. The j-th article is associated a latent description θ_j, with $\theta_j[k]$ the percentage of the k-th topic in the article (e.g., a given article might be described as 70% "artificial intelligence", 20% "high performance computing" and 10% "application"). Eventually, latent descriptions u_i of the user and $\theta_j + \epsilon_j$ of the article are optimized[3], such that $\langle u_i, \theta_j + \epsilon_j \rangle$ best fits the collaborative matrix $\mathcal{M}_{i,j}$,

$$(u^*, \epsilon^*) = \arg\min_{u,\epsilon} \left\{ \begin{array}{r} \sum_{i,j} c_{i,j} (\mathcal{M}_{i,j} - \langle u_i, \theta_j + \epsilon_j \rangle)^2 \\ +\lambda_u ||u||_2^2 + \lambda_v ||\epsilon||_2^2 \end{array} \right\}$$

with λ_u and λ_v respectively the weights of the regularization terms preventing CTR from overfitting the data, $c_{i,j} = 1$ if $\mathcal{M}_{i,j} = 1$ (the j-th document is in the i-th user's library), and .01 otherwise.

III. Exploratory analysis

This section describes the two anonymized large-scale and proprietary databases considered in this study, kindly provided by Web hiring agency Qapa[4] and non-profit organization ABG[5].

[3] ϵ_j is added, allowing the article latent description to diverge from its (fixed) topic proportion θ_j, to better fit the collaborative information.

[4] https://www.qapa.fr/

[5] http://www.intelliagence.fr/

A. Data

The Qapa database, focused on low-paid temporary jobs and unskilled applicants, gathers 5 million resumes, 4 million job ads and circa 11 million interactions over Jan. 2012- July 2015. An applicant is not required to provide a resume as they often do not have any. An applicant might access the Qapa platform using her mobile phone and writing a few (possibly colloquial) sentences; her description will also be enriched from her clicks on the job ads. The ABG database, focused on PhD applicants and job ads for PhDs in industry, gathers 10,000 job ads, 14,000 resumes and 68,000 clicks, recorded over the 2010-2015 period. To a job ad (resp. a resume) are usually attached a date and a city converted in geolocation (respectively, a date; the applicant geolocation is not available due to privacy constraints). Besides the job ad and resume corpora, each database involves the collaborative filtering matrix $\mathcal{M}$, with $\mathcal{M}_{i,j} = 1$ iff the i-th applicant clicked on the j-th job ad). Job ads and resumes are written in French, except for the ABG database where circa 30% of the job ads are in English (Table I). As shown on Fig. 1, job ads and resumes do not follow the same word distributions, neither for the unqualified job sector (Fig. 1 top left) nor for the qualified one (top right). This distribution gap explains why a direct matching between job ads and resumes does not yield good results (as shown by complementary experiments, omitted for brevity). Likewise, qualified vs unqualified resumes (respectively, job ads) follow different word distribution as shown in Fig. 1 bottom left (resp. bottom right).

The Qapa platform was drastically extended in April 2015, using another 2,000 skill features from the official French job ontology; applicants and recruiters can and do use these features to enrich their job ads or resumes. In the following, the excerpt of the Qapa database recorded over 2015 May-July (56,000 job ads and 31,000 resumes) is considered. An excerpt of the ABG data is considered, retaining job ads and resumes such that each applicant has clicked on at least two job ads and each job ad has been clicked upon by at least two applicants (10,400 job ads, 8,400 resumes).

	Qapa	ABG
# jobs in thousands (# wd)	56 (60)	10.4 (113)
# resumes in thousands (# wd)	31 (46)	8.4 (-)
# clicks in thousands (spars.)	226 (13.10^{-5})	63 (71.10^{-5})

TABLE I
Details of the Qapa and ABG excerpt databases, respectively recorded in May-July 2015 and Apr. 2010-Sept 2015

B. NLP Representations and Metrics

The primary representation of job ads and resumes is that of bags of words (using tf-idf to limit the impact of common words). Linear or non-linear dimensionality reduction techniques have been used to map job ads and resumes onto $\mathbb{R}^D$: Latent Semantic Analysis (LSA, with $D = 200$ singular vectors) [34], Latent Dirichlet Allocation (LDA, with $D = 200$ topics) [33], or sentence2vec [19]. The collaborative representation of the j-th job ad is the set of applicants clicking on

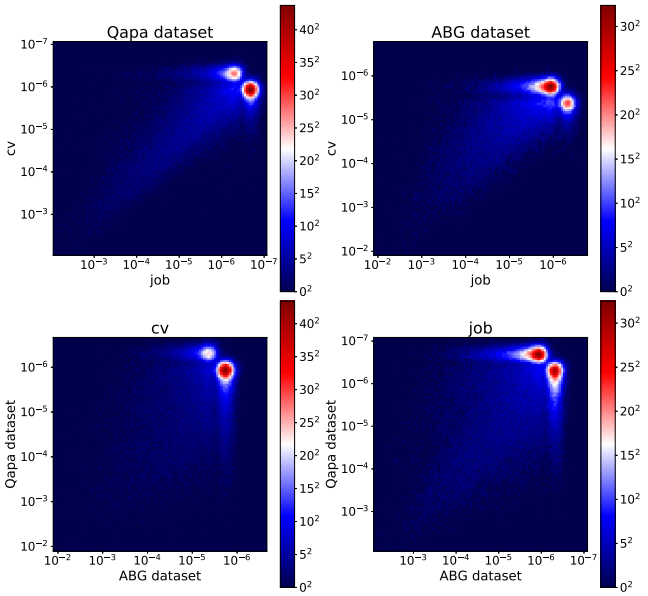

Fig. 1. Word distributions in Qapa and ABG databases in negative log scale. Top left: word is located at position x, y with x (resp. y) being the word frequency in Qapa jobs (resp. resumes). Top right: in ABG. Bottom left (resp. right): word frequency in Qapa resumes vs ABG resumes (resp. in Qapa jobs vs ABG jobs). The color indicates the distribution density (the redder, the more dense).

this job ad, noted $\mathcal{M}_{.,j}$. The similarity used together with the primary (respectively, reduced or collaborative) representation is the cosine similarity:

$$sim(u,v) = \frac{\langle u, v \rangle}{\|u\|_2 \|v\|_2}$$

C. Quality of the collaborative filtering matrix $\mathcal{M}$

Let m (respectively n) denote the number of users or applicants (resp. the number of items or job ads). The quality of the $m \times n$ collaborative filtering matrix $\mathcal{M}$ is assessed as follows. A lesioned matrix is defined by selecting 10% (i,j) pairs such that $\mathcal{M}_{i,j} = 1$ and setting them to 0 (removing at most 1 item per user). Mainstream recommendation approaches (see e.g. [35]) are used to re-estimate all $\mathcal{M}_{i,j}$ such that $\mathcal{M}_{i,j} = 0$, with

$$\widehat{\mathcal{M}}_{i,j} = \sum_{k=1}^{n} sim(j,k)\mathcal{M}_{i,k} \qquad (1)$$

For **content item-based recommendation**, $sim(j,k)$ stands for the cosine similarity among the initial or reduced representations of the j-th and k-th items. Only the reduced LSA representation is considered in this section (the other representations bring no improvement upon LSA); the associated recommendation will be referred to as LSA-based recommendation and denoted LSA-bl in the following.

For **$\mathcal{M}$-based recommendation**, $sim(j,k)$ stands for the cosine similarity among their collaborative representation, that is, $\mathcal{M}_{.,j}$ and $\mathcal{M}_{.,k}$.

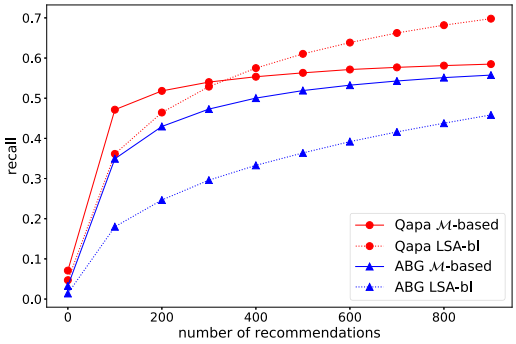

Fig. 2. $\mathcal{M}$-based and LSA-based recommendation in warm-start mode on Qapa and ABG: average recall per-user (fraction $y(T)$ of removed clicks recommended with rank lesser than T), averaged over 10 runs. LSA-based recommendation is dominated by $\mathcal{M}$-based recommendation, particularly so for the ABG database, which is blamed on the smaller size of the corpus.

D. Assessment of the databases

The quality of a database is estimated from the *recall@T* performance indicator, defined per user as the fraction of removed clicks ranked in the top-T $\widehat{\mathcal{M}}_{i,j}$ values, averaged over 10 runs. The performance indicators are measured for the LSA-based (with dimension 600) and $\mathcal{M}$-based recommendation approaches.

As displayed in Fig. 2, for 52% of Qapa (resp. 44% of ABG) applicants, the relevant recommendations appear in the top-200 (out of 56,000 job ads for Qapa and 14,000 for ABG); this ranking is considered to be acceptable by the Qapa and ABG experts. For a significant fraction of the applicants however, the first relevant recommendation is ranked below the top 10,000, being thus useless in a real-world setting. For both databases, the $\mathcal{M}$-based (item-based) similarity is more effective than the LSA one; a tentative interpretation is that the similarity of any two job ads is better captured from the applicant behaviors, than from the actual wording of the job ad (even after the dimensionality reduction achieved by LSA).

Surprisingly, a small world structure is observed on the job ads: defining two job ads to be adjacent if they have been clicked upon by a same applicant, it appears that any two Qapa (resp. ABG) job ads are at most 6-step (resp. 8-step) distant. We shall return to this remark in Section VII.

IV. Overview of LaJam

This section presents the proposed *Language Model-based* JAM (LAJAM) system, focusing on the semi-cold start problem of recommending brand new job ads to a known user. As said, this approach is suited to the temporary work sector, of prime importance in today's labor market[6].

Overall, LAJAM proceeds by i) learning a mapping from the job ad space onto $\mathbb{R}^d$, such that this mapping reflects the job similarity derived from matrix $\mathcal{M}$; ii) using this mapping together with $\mathcal{M}$ to rank the job ads for each applicant.

[6]As of 2017, 90% of the work contracts in France fall in the temporary work sector, for an average duration of one week.

A. Continuous language model for job ads

Letting (j,k) be a pair of job ads, boolean similarity $\text{sim}_{\mathcal{M}}(j,k)$ is set to 1 iff the j-th and k-th job ads have been clicked upon by at least one same applicant, and 0 otherwise[7]:

$$\text{sim}_{\mathcal{M}}(j,k) = \begin{cases} 1 & \text{if } \langle \mathcal{M}_{.,j}, \mathcal{M}_{.,k} \rangle > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The language model ϕ, represented by a parameter vector $W \in \mathbb{R}^L$, maps the initial or reduced description spaces (tf-idf, LSA or LDA) of the job ads onto $\mathbb{R}^d$, such that it induces a similarity noted sim_W compliant with the collaborative filtering similarity $\text{sim}_{\mathcal{M}}$:

$$\text{sim}_W(\mathbf{y}_j, \mathbf{y}_k) > \text{sim}_W(\mathbf{y}_j, \mathbf{y}_\ell) \text{ if } \text{sim}_{\mathcal{M}}(j,k) > \text{sim}_{\mathcal{M}}(j,\ell)$$

where $\mathbf{y}_j$ (resp., $\mathbf{y}_k$, $\mathbf{y}_\ell$) stands for the initial or reduced representation of the j-th (resp, k-th, ℓ-th) job ad, and $\text{sim}_W(\mathbf{y}_j, \mathbf{y}_k)$ is the cosine of vectors $\phi(\mathbf{y}_j)$, $\phi(\mathbf{y}_k)$. Mapping ϕ is learned using a Siamese neural architecture [24] (Fig. 3). Taking inspiration from [25], the elementary loss associated to any (j,k) job ad pair is the sum of two terms (to be minimized):

$$\begin{aligned} \mathcal{L}_W(j,k) = & \ \text{sim}_{\mathcal{M}}(j,k) \times (1 - \text{sim}_W(\mathbf{y}_j, \mathbf{y}_k)) \\ & + (1 - \text{sim}_{\mathcal{M}}(j,k)) \times \left[\text{sim}_W(\mathbf{y}_j, \mathbf{y}_k)\right]_+ \end{aligned} \quad (3)$$

where A_+ stands for $\max(A, 0)$. The first term in Eq. (3) is meant to maximize the cosine similarity of $\phi(\mathbf{y}_j)$ and $\phi(\mathbf{y}_k)$ for similar j and k. The second term is meant to push away the representations $\phi(\mathbf{y}_j)$ and $\phi(\mathbf{y}_k)$ of two dissimilar j and k job ads, thus preventing the discovery of trivial (constant) ϕ embeddings.

Overall, ϕ is trained to minimize the loss defined as:

$$\mathcal{L}(W) = \sum_j \sum_{k \in V(j)} \mathcal{L}_W(j,k) \quad (4)$$

where j ranges over all job ads and k ranges over a subset $V(j)$ (see below).

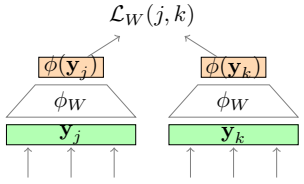

Fig. 3. Mapping ϕ from the job ad space onto $\mathbb{R}^d$ is trained using a Siamese neural net architecture, such that the sim_W (ϕ-based similarity) complies with $\text{sim}_{\mathcal{M}}$.

B. Discussion

In principle, the computational cost is quadratic in the number of job ads, hindering the computational tractability of the optimization problem (Eq. 4). The challenge is to both enforce the compliance of sim_W w.r.t. $\text{sim}_{\mathcal{M}}$, and the computational tractability of the approach. In practice, the dissimilar pairs outnumber the similar pairs by a factor 10^4 (on average, an Qapa applicant clicks on 7.4 job ads; a ABG applicant, on 4.3 job ads). The idea is thus to restrict the number of considered pairs (j,k) by imposing $k \in V(j)$, where $V(j)$ includes: i) all k such that $\text{sim}_{\mathcal{M}}(j,k) = 1$; and ii) K as many dissimilar job ads (with K a hyper-parameter set to 10 in the experiments), thus aggressively subsampling the dissimilar job ads.

Two subsampling options have been considered. The first one uniformly samples the dissimilar job ads, taking inspiration from negative sampling [17]. A large sized $V(j)$ is however required to be representative of the many dissimilar pairs, adversely affecting the computational cost. In the experiments, the size of $V(j)$ is set to 10 times the number of job ads similar to j. The second option consists of sampling the nearest-miss job ads w.r.t. ϕ, defined as the dissimilar job ads nearest to j according to the current mapping ϕ. This option will not be considered in the following: preliminary experiments show that, though it succeeds in pushing away the dissimilar job ads, these are replaced by other dissimilar job ads and the value of the loss only very slowly decreases along the training epochs.

C. LAJAM *Recommendation*

Eventually, the fit between the (known) i-th user and a new job $\mathbf{y}$ is computed by replacing the LSA- or $\mathcal{M}$-based similarity in Eq. (1) by the ϕ-based similarity:

$$\mathcal{F}(i, \mathbf{y}) = \sum_{j=1}^{n} \text{sim}_W(\mathbf{y}, \mathbf{y}_j) \mathcal{M}_{i,j}$$

with $\mathbf{y}$ (respectively $\mathbf{y}_j$) the initial or reduced representation of the new job (resp. the j-th job). The recommendations to the i-th user are ordered by decreasing value of $\mathcal{F}(i, \mathbf{y})$.

The computational complexity of the recommendation is optimized by limiting the search to the nearest neighbors of the current job ad (with logarithmic complexity in the number n of job ads) taking inspiration from [36].

V. EXPERIMENTAL SETTING

This section presents the experimental setting and the performance indicators used to comparatively assess LAJAM.

A. Databases

A JAM approach can best be evaluated on real-world databases. As discussed in Section II however, such databases are not public due to privacy constraints, hindering the reproducibility of the experimental evaluation and the fair assessment of the performances. For this reason, besides the proprietary Qapa and ABG databases (Section III), LAJAM is also evaluated on the public CiteULike database and compared to CTR using same folds and same experimental setting[8] [8]. CiteUlike (Section II) is concerned with two functionalities:

[7]Refined similarities, e.g., taking into account the number of applicants clicking on both job ads, have been considered. These are omitted in the rest of the paper as they make no significant difference in the experiments, due to the extreme sparsity of the collaborative filtering matrix.

[8]Data are available at http://www.citeulike.org/faq/data.adp. The public implementation of CTR, available at http://www.cs.cmu.edu/~chongw/citeulike/ is used, with 250 iterations.

warm-start recommendation and semi-cold start recommendation (recommending new articles to known users), referred to as "out-of-matrix" recommendation in [8]. The database includes 16,980 articles using a tf-idf description on a 8,000 words vocabulary. Only article titles and abstracts are considered, including 67 words on average. The collaborative filtering matrix $\mathcal{M}$ describes the subset of articles in the individual library of each of the 5,551 platform users. $\mathcal{M}$ sparsity is 22.10^{-4} (less sparse than Qapa and ABG by an order of magnitude).

B. Goal of experiments and performance indicators

The goal is to assess LAJAM performances in warm-start and semi-cold start mode. The performance indicators are the recall@20 and recall@100 (fraction of removed clicks ranked in the top-20 or top-100 recommendations). On CiteULike in semi-cold start mode, LAJAM is compared to CTR and two baselines: LSA-bl achieves content item-based recommendation (Eq. 1) using an LSA representation with cosine similarity; LDA-bl achieves content item-based recommendation using an LDA representation with dot product similarity.

On the Qapa and ABG databases, LAJAM is evaluated in warm-start and semi-cold start modes, comparatively to LSA-bl and LDA-bl (the latter is omitted for brevity as it is dominated by LSA-bl). In warm-start mode, the performance indicators are averaged on 10 runs, considering each a lesioned collaborative matrix built by randomly removing 10% of the clicks, subject to removing at most 1 click per user. In semi-cold start mode, the performance indicators are assessed along a standard 10 fold CV procedure (where each fold includes all users and 1/10th of the job ads).

C. Hyper-parameters

LSA-bl and LDA-bl use a 200-dim representation.

Two neuronal architectures have been considered for LAJAM. In all cases, the dimension D of the input is the tf-idf dimension (8,000 for CiteULike and 10,000 for Qapa and ABG). The shallow architecture is a D-200 network with $\tanh$ activations, initialized so as to reproduce the LSA representation. The deep architecture is a D-1,000-1,000-200 network, with 2 dense hidden layers and $\tanh$ activations. It is initialized so as to reproduce the LSA representation on the last layer. This architecture is selected based on a previous study [37]. The weights are augmented with a noise initialized as in [38] with multiplicative factor 10^{-3}. All architectures are implemented in Theano [39] and optimized using Adam [40].

For Qapa and ABG, the word description is augmented with the posting time and geolocation attached to each job ad. For fairness, LSA-bl and LDA-bl use an enriched similarity accounting for this extra information:

$$sim(j,k) \rightarrow sim(j,k) - \lambda_1 d_g(j,k) - \lambda_2 d_t(j,k) \quad (5)$$

where $d_g(j,k)$ (respectively $d_t(j,k)$) is the Euclidean distance between the geolocations (resp. the posting time) of the j-th and k-th job ads and the λ weights are selected by grid search to optimize the recall on the training set.

VI. EXPERIMENTAL VALIDATION

This Section reports on the experimental validation of LAJAM. All reported computational times are obtained on 16 CPU cores (Intel Xeon E7 @ 2.20GHz).

A. Comparative assessment on CiteULike

Table II reports the Recall@20 and Recall@100 on the CiteULike dataset in semi-cold start mode [8]. Surprisingly, CTR only slightly improves on LDA-bl; even more surprising is the fact that LSA-bl significantly dominates both LDA-bl and CTR.

A tentative interpretation for these facts goes as follows. Let Θ_i denote the sum, over all documents selected by the i-th user, of their LDA representation. LDA-bl recommends to the i-th user the documents with LDA representation θ and highest dot product $\langle \theta, \Theta_i \rangle$.

On the other hand, LSA-bl recommends to the i-th user the documents with LSA representation x and highest sum of $cosine(x, x_i)$ where x_i ranges over the LSA representation of the documents selected by the i-th user. With no loss of generality (regarding the LSA baseline recommendation) let us assume that all LSA representations have L_2 norm set to 1, and let Λ_i denote the sum of the x_i. The LSA baseline recommends to the i-th user the documents with LSA representation x and highest dot product $\langle x, \Lambda_i \rangle$.

In both cases, a "center of attraction" for the i-th user is defined from the data (Θ_i for LDA and Λ_i for LSA) and the recommended documents are those nearest to this center in L_2 norm (using a dot product). An important difference, beyond how the representations are defined, is that the document representations lie on the L_1 unit hypersphere in the LDA case, and on the L_2 unit hypersphere in the LSA case. After the celebrated discussion about the comparative effects of L_2 vs L_1 regularization [41], given a convex objective (here the distance to the center of attraction), the isolines thereof intersect an L_1 ball in a more sparse *and more instable* manner as for an L_2 ball. Said otherwise, small differences in the document representation have more impact in the LDA case than in the LSA case.

CTR improves on LDA-bl: it operates on the same search space and adjusts the "center of attraction" for the i-th user (replacing Θ_i by u_i). Likewise, LAJAM with a shallow architecture improves on LSA-bl: it operates on the same search space and adjusts the metric (modifying the LSA vectors). With a deep architecture however, LAJAM is dominated by both shallow-LAJAM and LSA-bl (though it still improves on CTR in a statistically significant manner).

B. Warm-start recommendation on Qapa and ABG

As shown on Fig. 4, shallow-LAJAM is dominated by the $\mathcal{M}$-based recommendation in the beginning of the Recall curves (for $p < 100$) and it dominates the baselines for $p > 100$. The improvement concerns circa 20% of the applicants, for whom the relevant recommendations by LAJAM are in the top 1,000 (as opposed to, below the 10,000 rank for LSA- and $\mathcal{M}$-based recommendation). Deep-LAJAM, slightly dominated

	R@20	R@100
LSA-bl	$\mathbf{.332 \pm .001}$	$.631 \pm .003$
LDA-bl	$.247 \pm .005$	$.584 \pm .005$
CTR	$.271 \pm .001$	$.587 \pm .001$
shallow LAJAM	$\mathbf{.327 \pm .004}$	$\mathbf{.652 \pm .005}$
deep LAJAM	$.279 \pm .005$	$.606 \pm .006$

TABLE II

SEMI-COLD START PERFORMANCE ON CITEULIKE: RECALL@20 AND RECALL@100 PERFORMANCES OF LSA-BL, LDA-BL, CTR AND LAJAM WITH SHALLOW AND DEEP NEURONAL ARCHITECTURES (PERFORMANCE AVERAGED USING A 5-FOLD CROSS-VALIDATION). THE TRAINING COMPUTATIONAL TIME IS 10 MIN FOR SHALLOW-LAJAM AND 20 MIN FOR DEEP-LAJAM.

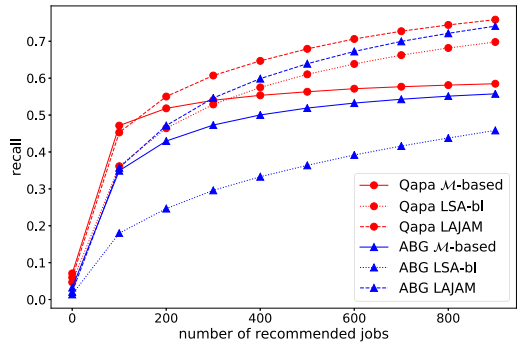

Fig. 4. Warm-start performance on Qapa (56,000 job ads) and ABG (10,000 job ads): Recall $y(p)$, percentage of recovered clicks in the top p recommendations, averaged out of 10 runs, for shallow LAJAM (dashed line) vs LSA-bl (dotted line) and the $\mathcal{M}$-based recommendation (plain line).

by shallow-LAJAM, has been omitted for readability. The fact that $\mathcal{M}$-based recommendation does not improve when p increases, and reaches a plateau with moderate performance suggests that a pure collaborative filtering approach is not appropriate to tackle the JAM problem. The computational time is ca. 7 hours (resp. 50 minutes) for shallow-LAJAM training and 4 minutes (resp. 12') for recommendation for all users for Qapa (resp. ABG). The training computational time of deep-LAJAM is twice that of shallow-LAJAM.

C. Semi cold-start recommendation on Qapa and ABG

Table III reports the performance of deep and shallow LAJAM on the Qapa and ABG databases, comparatively to LSA-bl. LDA-bl is omitted due to its lower performances[9]. Four settings are considered, where the job ad representation includes: i) the words only; ii) the words and the geolocation; iii) the words and the posting time; iv) the words, the geolocation and the posting time.

A first remark regards the utmost importance of the geolocation on Qapa, and of the posting time for ABG. This confirms the difference of the low vs highly qualified job sectors and the importance of letting the system learn how to take these information into account. Specifically, on the Qapa data the geolocation information yields an improvement of circa 12% for LSA-bl and 4% for LAJAM. In the meanwhile, the geolocation information does not yield any extra performance on the ABG dataset; a tentative interpretation for this fact is that PhDs and post-docs are used to geographic mobility. On the other hand, the time information yields an improvement of circa 30% on the ABG data (spanning over 5 years) and makes little difference on the Qapa dataset (spanning over 3 months).

Note that LSA-bl benefits from a considerable advantage, exploiting the geolocation and posting time distances with optimal weights (Eq. (5)). In contrast, LAJAM learns from scratch how to best exploit the geolocation and time information, expressed via 3 features among 10,000+ ones (the word features). A more informed LAJAM initialization will be considered in further work for a fair comparison with LSA-bl.

A second remark is that deep-LAJAM is always dominated by shallow LAJAM: this is blamed on the lack of regularization for the deep architecture, already noted on the CiteULike problem.

For the word-only setting, shallow-LAJAM very significantly outperforms the baseline on Qapa: by 9% regarding the Recall@20 and 5% regarding the Recall@100.

On ABG, shallow-LAJAM is dominated by LSA-bl in the word-only setting for the Recall@20 performance indicator, and slightly though significantly dominates the baseline for the Recall@100 indicator. Shallow-LAJAM likewise dominates the baseline in the word+geolocation setting.

VII. CONCLUSION AND PERSPECTIVES

The paper contribution focuses on the automatic matching of job ads and applicants, based on proprietary datasets illustrating both blue-collar and white-collar job markets. The proposed LAJAM approach succeeds in learning an *ad hoc* language model on the job ad space, accounting for the collaborative filtering metrics and flexibly exploiting extra information such as geolocations and posting times. The performances of LAJAM have been successfully assessed on the public CiteULike dataset, comparatively to CTR [8] and two baselines using content item-based recommendation on the top of an LSA (resp. an LDA) representation. A limitation of the approach is that deep-LAJAM currently lags behind shallow-LAJAM. On-going work is concerned with alleviating this limitation, through considering more data or regularizing the optimization objective.

A short-term perspective is to reconsider the training loss and the uniform sampling of the dissimilar job ad pairs. The current loss yields a contracting language model: if jobs i and j are similar, and j and k are similar (while i and k are dissimilar), the resulting language model tends to locate i and k close to each other by transitivity. In order to prevent the contraction of the language model space, one should enforce the selection of 2-step dissimilar job pairs (i, k) and push them away. In CiteULike, the job ads are structured as a small world of diameter 2.3; therefore most dissimilar job ad pairs are 2-step distant. Quite the contrary, in Qapa and ABG, job ads are structured as a small world with diameter circa respectively 6 for Qapa and 8 for ABG (section III): the uniform selection

[9]On Qapa, LDA-bl yields .26 for Recall@20 and .62 for Recall@100. On ABG, it yields .15 for Recall@20 and .41 for Recall@100 (words only).

Semi-cold start	Qapa					ABG				
	–	geoloc	time	geo-time	CPU	–	geoloc	time	geo-time	CPU
LSA-bl R@20	$.404 \pm .007$	$\mathbf{.636 \pm .003}$	$.439 \pm .007$	$\mathbf{.656 \pm .003}$	5 min	$\mathbf{.254 \pm .010}$	$.258 \pm .008$	$\mathbf{.574 \pm .008}$	$\mathbf{.579 \pm .009}$	5 min
LSA-bl R@100	$.694 \pm .005$	$\mathbf{.829 \pm .003}$	$.713 \pm .005$	$\mathbf{.835 \pm .002}$		$.522 \pm .012$	$.528 \pm .013$	$\mathbf{.814 \pm .007}$	$\mathbf{.817 \pm .007}$	
LAJAM (shallow) R@20	$\mathbf{.495 \pm .006}$	$.549 \pm .006$	$\mathbf{.523 \pm .005}$	$.558 \pm .006$	200min	$.226 \pm .008$	$\mathbf{.260 \pm .009}$	$.391 \pm .009$	$.392 \pm .011$	25min
LAJAM (shallow) R@100	$\mathbf{.743 \pm .007}$	$.784 \pm .005$	$\mathbf{.764 \pm .006}$	$.790 \pm .005$		$\mathbf{.544 \pm .008}$	$\mathbf{.599 \pm .006}$	$.761 \pm .009$	$.755 \pm .008$	
LAJAM (deep) R@20	$.475 \pm .007$	$.499 \pm .006$	$.483 \pm .006$	$.503 \pm .007$	400min	$.200 \pm .008$	$.199 \pm .008$	$.312 \pm .008$	$.336 \pm .009$	75 min
LAJAM (deep) R@100	$.725 \pm .007$	$.748 \pm .007$	$.731 \pm .006$	$.752 \pm .006$		$.493 \pm .010$	$.492 \pm .011$	$.705 \pm .007$	$.717 \pm .008$	

TABLE III

SEMI-COLD START SETTING: SHALLOW-, DEEP-LAJAM AND LSA-BL RECALL@20 AND RECALL@100 PERFORMANCES ON QAPA (5,600 JOB ADS) AND ABG (1,040 JOB ADS) (10-FOLD CV). THE TRAINING TIME IS REPORTED IN THE CPU COLUMN; THE RECOMMENDATION TIME (FOR ALL USERS) IS 27' FOR LSA-BL(RESP. 46' FOR SHALLOW-LAJAM) ON QAPA AND 6' FOR BOTH ON ABG.

of the dissimilar job ad pairs thus is unlikely to select 2-step distant dissimilar job ad pairs, and to efficiently prevent the contraction of the language model space.

A long-term research perspective is to exploit LAJAM in "what if" reasoning mode, and to identify the skills that a user should most preferably acquire in order to maximize his job opportunities in a given context: in other words, to extend LAJAM from the recommendation of jobs, to the recommendation of professional training.

Acknowledgments

The authors wish to warmly thank the Qapa and ABG CEOs, particularly Stéphanie Delestre, for many fruitful discussions; without their generous help this study would not have been possible.

REFERENCES

[1] D. T. Mortensen and C. A. Pissarides, "Job creation and job destruction in the theory of unemployment," *The Review of Economic Studies*, 1994.

[2] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, "Matching people and jobs: A bilateral recommendation approach," in *HICSS*, vol. 6. IEEE, 2006, p. 137.

[3] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, "Machine learned job recommendation," in *RecSys*. ACM, 2011, pp. 325–328.

[4] M. Tinghuai, Z. Jinjuan, T. Meili, T. Yuan, A.-D. Abdullah, A.-R. Mznah, and L. Sungyoung, "Social network and tag sources based augmenting collaborative recommender system," *IEICE Transactions on Information and Systems*, vol. 98, no. 4, pp. 902–910, 2015.

[5] E. Malherbe, M. Diaby, M. Cataldi, E. Viennet, and M.-A. Aufaure, "Field selection for job categorization and recommendation to social network users," in *Proc. of the IEEE/ACM International Conf. ASONAM*.

[6] T. Carpi, M. Edemanti, E. Kamberoski, E. Sacchi, P. Cremonesi, R. Pagano, and M. Quadrana, "Multi-stack ensemble for job recommendation," in *Proc. of the RecSys Challenge*. ACM, 2016, p. 8.

[7] Y. Koren and R. M. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, 2015, pp. 77–118.

[8] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *ACM SIGKDD*.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[10] S. Büttcher, C. L. A. Clarke, and G. V. Cormack, *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

[11] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *Machine learning challenges*, ser. LNCS. Springer, 2006, no. 3944, pp. 177–190.

[12] R. M. Bell and Y. Koren, "Lessons from the Netflix prize challenge," *ACM Sigkdd Explorations Newsletter*, vol. 9, no. 2, pp. 75–79, 2007.

[13] Y. Zhou and A. Nadaf, "Embedded collaborative filtering for" cold start" prediction," *arXiv preprint arXiv:1704.02552*, 2017.

[14] J. Weston, C. Wang, R. Weiss, and A. Berenzweig, "Latent collaborative retrieval," in *ICML*, 2012, pp. 9–16.

[15] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator." in *AISTATS*, vol. 1, 2011, p. 2.

[16] F. Färber, T. Weitzel, and T. Keim, "An automated recommendation approach to selection in personnel recruitment," *AMCIS*, 2003.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.

[18] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–43.

[19] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.

[20] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *ICML*, 2015, pp. 957–966.

[21] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *ACM SIGKDD*, 2015, pp. 1809–1818.

[22] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[23] C. J. Burges, "From RankNet to LambdaRank to LambdaMART: An overview," *Journal of Machine Learning Research*, 2010.

[24] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," *IJPRAI*, vol. 7, no. 04, pp. 669–688, 1993.

[25] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*. IEEE, 2005, pp. 539–546.

[26] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *CVPR*, 2014.

[27] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, 2015.

[28] P. Neculoiu, M. Versteegh, M. Rotaru, and T. B. Amsterdam, "Learning text similarity with Siamese recurrent networks," *ACL 2016*, 2016.

[29] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity." in *AAAI*, 2016, pp. 2786–2792.

[30] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *CIKM*.

[31] F. Zhu, J. Xie, and Y. Fang, "Learning cross-domain neural networks for sketch-based 3d shape retrieval," in *AAAI*, 2016, pp. 3683–3689.

[32] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *WWW*. ACM, 2015.

[33] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, 2003.

[34] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.

[36] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *J. Comput. Syst. Sci.*, 1973.

[37] T. Schmitt, P. Caillou, and M. Sebag, "Matching jobs and resumes: a deep collaborative filtering task," in *GCAI 2016*, vol. 41, 2016, pp. 124–137.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS*, vol. 9, 2010, pp. 249–256.

[39] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv preprints*. [Online]. Available: http://arxiv.org/abs/1605.02688

[40] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint*. [Online]. Available: https://arxiv.org/pdf/1412.6980

[41] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *ICML*, 2004, p. 78.