# Methods for Job Recommandation on Social Networks

Mamadou Diaby

N° attribué par la bibliothèque

# MÉTHODES POUR LA RECOMMANDATION D'OFFRES D'EMPLOI DANS LES RÉSEAUX SOCIAUX

**Mamadou DIABY**

**Directeur de thèse** : Emmanuel VIENNET

**Encadrants à Work4**: Guillaume LESEUR et Dr. Tristan LAUNAY

**Spécialité**: Informatique

présentée le **04 Juin 2015**

au Laboratoire de Traitement et Transport de l'Information (L2TI)
École doctorale Galilée
Université Paris 13, Sorbonne Paris Cité

**pour l'obtention du grade de Docteur de l'Université Paris 13**

devant le jury composé de :

Prof Emmanuel VIENNET, Paris 13/L2TI, directeur de thèse
Prof Marie-Aude AUFAURE, Ecole Centrale Paris/MAS, rapporteur
Prof Jean-François BOULICAUT, INSA Lyon/LIRIS, rapporteur
Dr Cécile BOTHOREL, Télécom Bretagne/LUSSI, examinateur
Benjamin COMBOURIEU, Work4/département R&D, examinateur
Prof Anissa MOKRAOUI, Paris 13/L2TI, examinateur

Villetaneuse, France, 2015

N° Assigned by the Library

# METHODS FOR JOB RECOMMENDATION ON SOCIAL NETWORKS

**Mamadou DIABY**

**PhD Advisor**: Emmanuel VIENNET

**Supervisors at Work4**: Guillaume LESEUR and Dr. Tristan LAUNAY

**A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in the subject of Applied Mathematics**

Presented on **June 04th, 2015**

At Laboratoire de Traitement et Transport de l'Information (L2TI)
École doctorale Galilée
Université Paris 13, Sorbonne Paris Cité

Before a jury composed of:

Prof Emmanuel VIENNET, Paris 13/L2TI, PhD advisor
Prof Marie-Aude AUFAURE, Ecole Centrale Paris/MAS, Rapporteur
Prof Jean-François BOULICAUT, INSA Lyon/LIRIS, Rapporteur
Dr. Cécile BOTHOREL, Télécom Bretagne/LUSSI, Reviewer
Benjamin COMBOURIEU, Work4/R&D department, Reviewer
Prof Anissa MOKRAOUI, Paris 13/L2TI, Reviewer

Villetaneuse, France, 2015

I, Mamadou DIABY, declare that this thesis titled, "*Méthodes pour la recommandation d'offres d'emploi dans les réseaux sociaux*" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University using data provided by the company Work4.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: May 25, 2015

Live as if you were to die tomorrow. Learn as if you were to live forever.
— Mahatma Gandhi

I would like to dedicate this work to my family - Boulaye Diaby (father), Fatoumata Diaby (mother), Oumar Diaby (brother) and Fatoumata Diaby (sister). Thanks for your unconditional love, support and encouragement throughout these so long years.

# Remerciements

Mes premiers remerciements vont à mon directeur de thèse, le professeur Emmanuel VIENNET, qui m'a accompagné tout au long de ces années. Ses conseils, son encadrement rigoureux et ses qualités scientifiques et humaines ont permis l'aboutissement de cette thèse. Je manque de mots pour lui dire combien j'ai aimé travailler avec lui, combien il a été un directeur de thèse formidable.

Je remercie chaleureusement Marie-Aude AUFAURE, Jean-François BOULICAUT, Cécile BOTHOREL, Benjamin COMBOURIEU et Anissa MOKRAOUI d'avoir accepté d'être membres de mon jury de thèse.

Stéphane Le VIET, en tant que fondateur de Work4, vous m'avez fait confiance début 2012 pour mener à bien ce projet de recherche. De cette aventure à Work4, je sors grandi tant sur le plan professionnel que sur le plan humain. Merci de votre confiance !

Tristan LAUNAY, tu as supervisé à Work4 cette thèse de doctorat pendant quinze mois, au cours desquels tu m'as appris à être rigoureux, à être exigent avec soi-même, à être perfectionniste et à toujours chercher à se surpasser. Ton expérience de thèse CIFRE m'a beaucoup servi et tu as pesé de tout ton poids pour faire mieux accepter les contraintes d'une thèse CIFRE au sein d'une entreprise n'ayant pas la culture de la recherche. Merci pour tout !

Benjamin COMBOURIEU, tu as été mon manager durant ces trois dernières années, tu as toujours cherché à concilier les contraintes académiques et industrielles. Merci pour tout ce que tu as fait pour que cette thèse se passe dans les meilleures conditions malgré toutes les difficultés inhérentes à la nature startup de Work4.

Guillaume, Léopold, Anne-claire et Emmanuel Malherbe, je ne vous ai pas oublié, je tiens à vous remercier pour votre soutien, pour vos conseils et pour vos innombrables coups de main spontanés. J'ai eu de la chance de travailler avec vous et j'ai beaucoup appris de chacun de vous.
Adrien (my Lord!), Matteo, Ouadia, Peter, Laurent, Ulysse et l'autre Adrien, merci pour votre bonne humeur; vous avez su me donner le sourire dans les moments les plus difficiles. Vous êtes les meilleurs !

Thierry, Nidhal, Amine, le bureau E202 du L2TI restera à jamais le nôtre. Je me rappellerai toujours de nos interminables discussions sur nos thèses respectives, de nos longues soirées

passées ensemble à murir nos idées et à peaufiner nos articles. Daniel, Blaise, Patricia, Raphaël et Ivan, nous avons formé avec fierté la team “recommandation et analyse des réseaux sociaux” du L2TI. Nous avons connu des moments de stress intenses pour boucler des papiers dont je garde un très bon souvenir car ces moments nous ont permis de mieux nous connaître. Je vous remercie tous de m’avoir accompagné durant ces années de thèse et de galère !

*Villetaneuse, France, 25 mai 2015* Mamadou DIABY

# Abstract

We are entering a new era of data mining in which the main challenge is the storing and processing of massive data: this is leading to a new promising research and industry field called Big data. Data are currently a new raw material coveted by businesses of all sizes and all sectors. They allow organizations to analyze, understand, model and explain phenomena such as the behavior of their users or customers. Some companies like Google, Facebook, LinkedIn and Twitter are using user data to determine their preferences in order to make targeted advertisements to increase their revenues.
This thesis has been carried out in collaboration between the laboratory L2TI[1] and Work4[2], a French-American startup that offers Facebook recruitment solutions. Its main objective was the development of systems recommending relevant jobs to social network users; the developed systems have been used to advertise job positions on social networks.
After studying the literature about recommender systems, information retrieval, data mining and machine learning, we modeled social users using data they posted on their profiles, those of their social relationships together with the bag-of-words and ontology-based models. We measure the interests of users for jobs using both heuristics and models based on machine learning. The development of efficient job recommender systems involved to tackle the problem of categorization and summarization of user profiles and job descriptions.
After developing job recommender systems on social networks, we developed a set of systems called Work4Oracle that predict the audience (number of clicks) of job advertisements posted on Facebook, LinkedIn or Twitter. The analysis of the results of Work4Oracle allows us to find and quantify factors impacting the popularity of job ads posted on social networks, these results have been compared to those of the literature of Human Resource Management.
All our proposed systems deal with privacy preservation by only using the data that social network users explicitly allowed to access to; they also deal with noisy and missing data of social network users and have been validated on real-world data provided by Work4.

**Keywords**: Social network analysis, Job recommender systems, Audience of job ads, Data mining, Facebook, LinkedIn, Twitter

[1] Laboratoire de Traitement et Transport de l'Information (http://www-l2ti.univ-paris13.fr).
[2] http://www.work4labs.com

# Résumé

Nous sommes à l'aube d'une nouvelle ère du data mining, celle du stockage, traitement, analyse et exploitation des données massives que l'on appelle Big Data. Les données sont devenues une nouvelle matière première, très prisée par des entreprises de tout type et de toute taille à travers le monde ; elles permettent d'analyser, de comprendre, de modéliser et d'expliquer certains phénomènes comme le comportement et les préférences des utilisateurs ou clients d'une entreprise donnée. La compréhension des préférences des utilisateurs et des clients d'une entreprise permet de leur proposer de la publicité ciblée afin d'augmenter les ventes et la satisfaction des clients et ainsi pouvoir améliorer les revenues de l'entreprise, ce que les géants du Web comme Google, Facebook, LinkedIn et Twitter ont bien compris.

Cette thèse de doctorat a été réalisée dans le cadre d'une convention CIFRE[1] entre le laboratoire L2TI[2] de l'université Paris 13 et la start-up franco-américaine Work4[3] qui développe des applications de recrutement sur Facebook. Son objectif principal était la mise au point d'un ensemble d'algorithmes et méthodes pour proposer aux utilisateurs des réseaux sociaux les offres d'emploi les plus pertinentes.

Le développement de nos algorithmes de recommandation a nécessité de surmonter de nombreuses difficultés telles que le préservation de la vie privée des utilisateurs des réseaux sociaux, le traitement des données bruitées et incomplètes des utilisateurs et des offres d'emploi, la difficulté de traitement des données multi-langues et, plus généralement, la difficulté d'extraire automatiquement[4] les offres d'emploi pertinentes pour un utilisateur donné parmi un ensemble d'offres d'emploi. Les systèmes développés durant cette thèse sont principalement basés sur les techniques de systèmes de recommandation, de recherche documentaire, de fouille de données et d'apprentissage artificiel ; ils ont été validés sur des jeux de données réels collectés par l'entreprise Work4.

Dans le cadre de cette étude, les utilisateurs d'un réseau social sont liés à trois types entités : les offres d'emploi qui leur sont pertinentes, les autres utilisateurs du réseau social auxquels ils se sont liés d'amitié et les données personnelles qu'ils ont publiées sur leurs profils. Les profils des utilisateurs des réseaux sociaux et la description de nos offres d'emploi sont constitués de plusieurs champs contenant des informations textuelles.

Après une étude de la littérature, nous avons modélisé les utilisateurs et offres d'emploi en utilisant les approches « sac de mots » et les modèles basés sur les ontologies. Nous avons

[1] Conventions Industrielles de Formation par la REcherche.

[2] Laboratoire de Traitement et Transport de l'Information (http://www-l2ti.univ-paris13.fr).

[3] http://www.work4labs.com

[4] En utilisant des algorithmes et des modèles mathématiques.

estimé les importances des différents champs (des utilisateurs et des offres d'emploi) dans le processus de recommandation d'offres d'emploi et avons estimé la pertinence des offres d'emploi pour les utilisateurs en utilisant à la fois des heuristiques et des modèles basés sur les techniques d'apprentissage artificiel.
Nous avons terminé cette thèse par le développement d'un ensemble de systèmes prédictifs appelés Work4Oracle capables d'estimer l'audience (nombre de clics) qu'obtiendrait une offre d'emploi postée sur Facebook, LinkedIn ou Twitter. Ces systèmes combinent à la fois les techniques de systèmes de recommandation et les méthodes d'apprentissage artificiel. Les résultats nous ont permis de quantifier les facteurs qui influent sur l'audience (donc l'attractivité) des offres d'emploi postées sur les réseaux sociaux.
Toutes les données utilisées par nos différents algorithmes ont été explicitement autorisées par les utilisateurs des réseaux sociaux correspondants afin de respecter leur vie privée.

**Mots clefs** : Analyse des réseaux sociaux, Systèmes de recommandation d'offres d'emploi, Audience des offres d'emploi, Fouille de données, Facebook, LinkedIn, Twitter

# Contents

# List of Figures

# List of Tables

# 1 Introduction

*"We're entering a new world in which data may be more important than software."*

- Tim O'Reilly, Founder and CEO of O'Reilly Media

**Contents**



## 1.1 Background and motivation

The world is increasingly becoming more digital and connected and, as a result, an ever increasing amount of data is being generated by individuals and companies alike [Tapscott, 2008]. Many of our daily actions are stored in the databases of one or more organizations (the websites one browses, the products/items one buys or likes, the public transportation people use, the persons one calls, the duration and content of calls, emails people send or receive, etc.). Collecting, storing and processing huge amounts of data is leading to a new promising research and industry field known as Big Data [Bollier and Firestone, 2010; Mayer-Schönberger and Cukier, 2013]. Data allow companies to analyze, understand and model the behavior of their users and customers and then make targeted advertisements to increase their incomes, benefits and customer satisfaction. Data are transforming the businesses. Based on this observation, many companies around the world are collecting huge amounts of data. Among them, we have Facebook, one of the most popular social networks in the world nowadays; according to the company's official statistics, it counts more than a billion of users around the world in 2015 [Facebook, 2015]. Two other current popular social networks are LinkedIn and Twitter, counting hundreds of millions of users [LinkedIn, 2015; Twitter, 2015].

A social network site can be defined as a web-based service [Boyd and Ellison, 2008] 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. Information posted by social network users on their profiles such as personal description, education history, work history, posts, likes, ratings given to items and the information about their social relations (friends, followers, ...) on the social network can be exploited by a recommender system to infer their interests for items. Recommender systems are often defined as software that elicit the interests of individual consumers for products, either explicitly or implicitly, and make recommendations accordingly [Xiao and Benbasat, 2007]. They have been popularized by Amazon.com[1] and Netflix[2] and have many practical applications [Bennett et al., 2007; Linden et al., 2003] that help users deal with information overload, and thus they have become an active research field for two decades.

Social network data are becoming important for many companies around the world and are often used to determine the interest of social network users for items in order to propose or advertise items to them. Based on this observation, Work4[3] (see the presentation in the chapter 2), a French-American software company has been founded in 2010 on a simple idea "*make every Facebook user a recruiter and a candidate*". It offers Facebook recruitment solutions. This thesis has been carried out in collaboration between Work4 and L2TI (Laboratoire de Traitement et de Transport de l'Information, a French computer science laboratory). It had two main objectives:

1. Define and develop a set of algorithms for job recommendation on social networks, mainly on Facebook and LinkedIn.
2. Define and develop a set of algorithms to predict the audience of job ads posted on social networks and find out the factors impacting these audience in order to optimize job ad campaigns (on Facebook, LinkedIn and Twitter).

Recommending jobs to social network users is a very complex task involving the combination of different techniques from recommender systems, natural language processing (job descriptions are written is a natural language), automatic text processing, data mining and machine learning. It requires to deal with missing, noisy data:

- Users generally use some social networks (like Facebook and Twitter) for fun, not for a professional purpose. As a result, they generally do not completely fill the fields of their profile pages that are interesting for the task of job recommendation (*Education* and *Work history* fields for instance). Some users publish fake information on these social networks which makes very difficult the task of recommending jobs to them.
- On Facebook, friends of a user are professionally heterogeneous (in other words, users have generally many friends whose professional skills are different from theirs), this

[1] http://www.amazon.com
[2] https://www.netflix.com
[3] http://www.work4labs.com

makes difficult the use of friends' data to improve users' profiles in the task of job recommendation.

- LinkedIn is professionally-oriented but due to privacy concern, we can only access data in some specific fields, this limits us to take a full advantage of LinkedIn data.
- Some job descriptions are poorly written and lack of context about their related companies and industries. These job descriptions are difficult to understand and process even for a human. Figure 1.1 shows an example of a poorly written job description in our databases, the automatic detection of poorly written job descriptions is sometimes difficult.

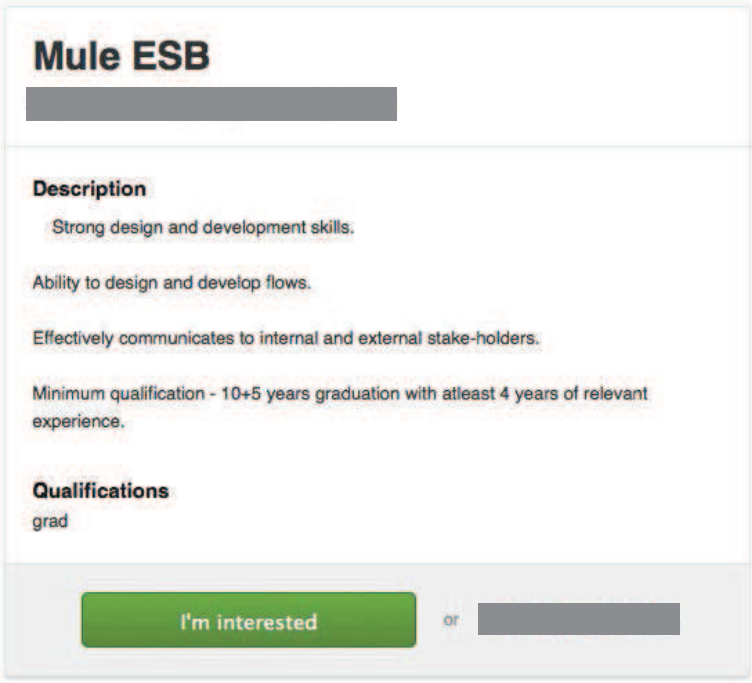



Figure 1.1 – An example of a poorly written job description. One can notice that such a job description is difficult to match with social network users, even for a human.

- We have a very sparse user-job matrix, containing the information about the interest of users for jobs. The sparsity rate is greater than 99.8% in our collected datasets: we are facing a *cold-start recommendation problem* (see section 3.1) in which recommendations are made based on few available information about users' preferences.

Since we are dealing with missing, noisy data, we need to develop methods that are robust to noises and missing data.

One of the main concerns about using social networks for recommendation is the fact that we have to deal with privacy preservation. We can only use data that users explicitly share. Our Facebook users have authorized the Work4's applications to access data in 5 fields (see

section 2.2): *Work*, *Education*, *Quote*, *Bio*, and *Interests*. LinkedIn users have only authorized 3 fields (see section 2.2): *Headline*, *Educations*, *Positions*. Our job descriptions have 3 fields: *Title*, *Description*, *Responsibilities*. Figures 1.3, 1.4 and 1.5 respectively show an example of a Facebook profile, a LinkedIn profile and a job description while Figure 1.2 depicts the modeling of our job recommendation process.

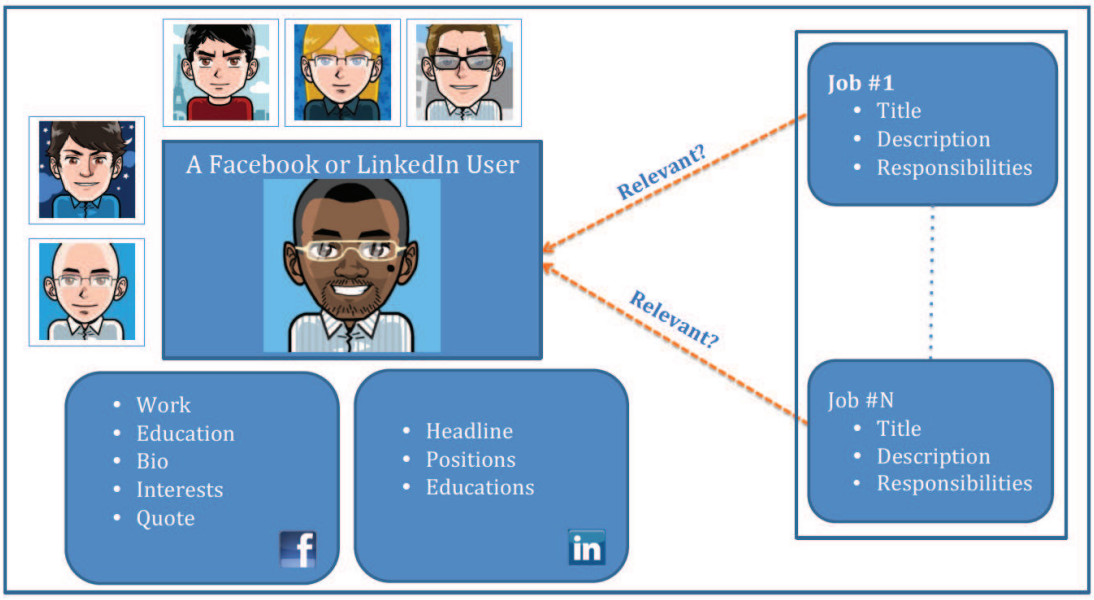



Figure 1.2 – Modeling of the process of recommending jobs to social network users. A Facebook or LinkedIn user can be linked to other social network users (friends, social connections, etc.); his profile is composed with several fields generally containing textual information.

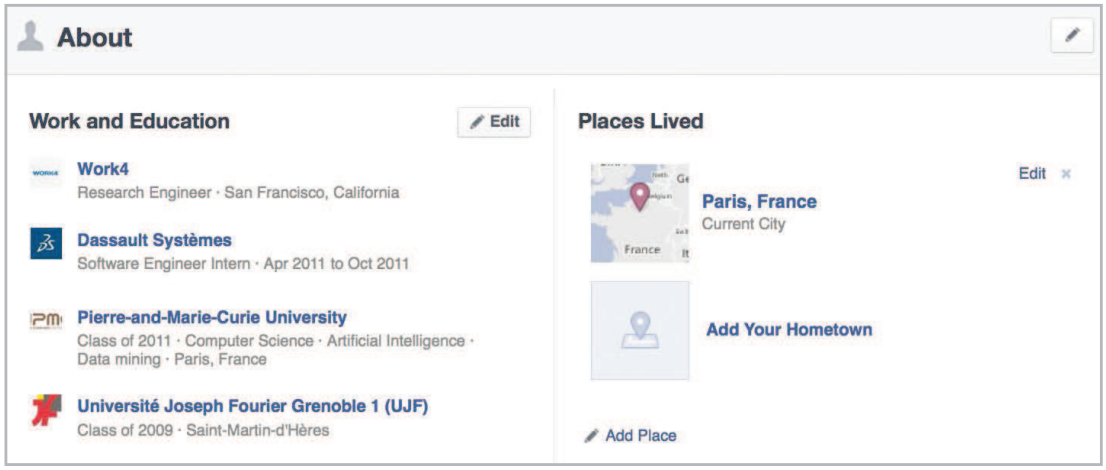



Figure 1.3 – An example of a Facebook profile. We note three fields (*Work*, *Education*, *Places lived*) with their sub-fields: *company name*, *position*, *start date*, *end date*, *location*, *class of*, *college/university/school name*, *description* and *concentrations*.

Matching a job with a user (in general) is a multidimensional problem involving to deal with

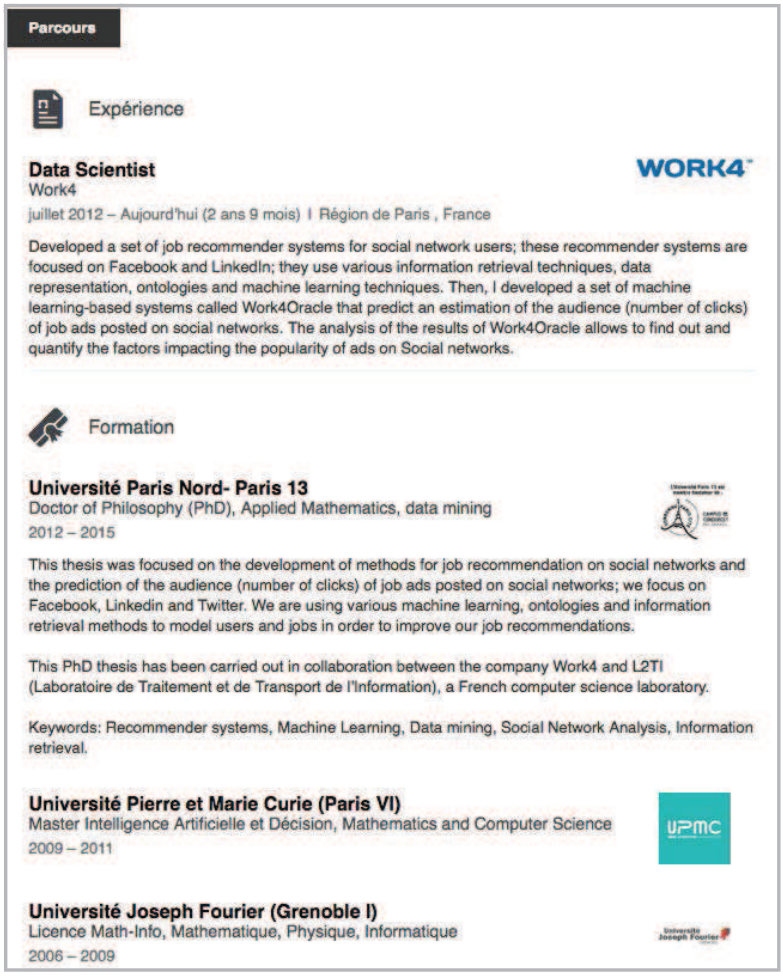



Figure 1.4 – An example of a LinkedIn profile with *Positions* and *Educations* fields and their sub-fields.

several layers of matching:

- Education of users **vs.** Education required by jobs.
- Experience of users **vs.** Experience required by jobs.
- Skills of users **vs.** Skills required by jobs.
- Users' spoken languages **vs.** Languages required of jobs.
- Locations preferred by users **vs.** Locations of jobs.
- Users' preferred industries **vs.** Industries of jobs.
- Salary that users want **vs.** Salary of jobs.


Figure 1.5 – An example of a job with the *Title*, *Description* and *Responsibilities* fields.

- Characteristics of users' ideal companies **vs.** Characteristics of companies offering jobs. We know that some users prefer working in big companies while others like small ones or even startups.

We also have to deal with the temporal aspect of job recommendation to users: the profiles of users are dynamic. The following example shows the evolution of the profile of a user: *software engineer* → *senior software engineer* → *manager* → *CTO*. In this study, we only focus on some dimensions since we have not all the data required to deal with all the dimensions mentioned above.

In this thesis, we also tackle the problem of estimating the audience of job ads posted on social networks and finding out the hidden factors impacting their audience. Being able to predict the audience of job ads allows us to help customers to optimize their job ad campaigns. This task is very complex too and requires the use of techniques from recommender systems, machine learning, human resource management and data mining. Here, we also deal with the problem of noisy, missing data of social network users and job descriptions, the problem of privacy preservation and the problem of multiple criteria job matching (mentioned above).

## 1.2 Contributions of this thesis

After studying the literature about recommender systems (see the paper [Bernardes et al., 2014]), data mining, social network analysis, artificial intelligence, machine learning and

information retrieval, we proposed several models for social network users, jobs and job ads. We used the different proposed models for social network users and jobs to develop a set of algorithms for job recommendation on social networks: some developed recommender systems are based on heuristic similarity functions while others use models learnt from our data using machine learning algorithms.

After developing job recommender systems, we proposed a modeling of job advertisements, and then designed and studied a set of systems that predict the audience of job ads posted on social networks: they are mainly based machine learning, techniques for recommender systems and our proposed vector models for job ads. The analysis of the results of proposed systems allows us to find out and quantify the hidden factors impacting the audience of job ads on social networks.

All our experiments have been conducted on real-world datasets collected by the company Work4 and their results allow us to obtain following contributions:

1. Our first series of experiments reveal that heuristic similarities used to recommend jobs to social network users give results that could be improved (especially for Facebook users). We estimate the importance of each field of users and jobs in the task of job recommendation and show that selecting variables (fields) according to their importance allows to slightly and significantly improve the quality of our job recommendation to social network users. These results have been published in [Diaby and Viennet, 2014a; Diaby et al., 2013, 2014].

2. Then, we show that the use of basic social recommendation methods (use of friends' data to enrich users' profiles) failed to improve our results, however the use of Rocchio's method to enrich users' vectors with related jobs' data (*Relevance Feedback*) improves the results. Our experiments on machine learning show that the use of Support Vector Machine (SVM) significantly improves the quality of our job recommendation. This method outperforms two state-of-the-art recommendation techniques: a Collaborative Filtering method and a hybrid recommender system (Collaborative Topic Regression) [Wang and Blei, 2011]. The conclusions of these studies have been published in [Diaby et al., 2014].

3. We also propose a new representation of social network users and job descriptions based on the taxonomy O*NET (see appendix B), suited to the task of job recommendation; this representation is an alternative to TF-IDF. The proposed representation model is an efficient dimensionality reduction method in the task of job recommendation. We then generalized this vector model to a multilayer vector model including additional features like abilities, skills and interests. We published these results in [Diaby and Viennet, 2014b, 2015c].

4. We design and study a series of decision-making systems called Work4Oracle that estimate the audience of job advertisements posted on Facebook, LinkedIn and Twitter.

To develop these systems, we propose a generic model for job ads on social networks. The analysis of the results of Work4Oracle allows us to find out the hidden factors impacting the audience of job ads on social networks. These studies led to two papers: one accepted for publication [Diaby and Viennet, 2015a] and the other under review [Diaby and Viennet, 2015a,b].

## 1.3 Organization of the manuscript

We present in the next chapter (chapter 2) the company Work4 and its products, this chapter depicts the context of social media-based recruitment and its related problems and describes the different social networks on which Work4's applications are based.

The chapter 3 presents the state-of-the-art of recommender systems: we describe different categories of recommender systems (content-based systems, collaborative filtering, hybrid systems, social and trust recommendation engines) and discuss their strengths and weaknesses.

The chapter 4 provides a state-of-the-art of information retrieval and data mining and knowledge discovery in databases, it presents data representation models and machine learning algorithms we used to develop our different systems.

The chapter 5 presents our proposed models for social network users and job descriptions and the proposed job recommender systems. It also compares our methods to two state-of-the-art recommendation techniques and discuss the strengths and weaknesses of studied systems.

We present in the chapter 6 our developed systems to predict the audience of job ads posted on social networks and an analysis of their results. This chapter discusses the different factors impacting the audience of job ads on social networks.

We finally conclude the work done in this thesis in the chapter 7 by discussing the strengths and weaknesses of our proposed methods and presenting the future directions.

# 2 Presentation of the company Work4

*"We're making everyone a recruiter, everyone a candidate."*

- Stéphane Le Viet, founder and CEO of Work4

**Contents**



## 2.1 Introduction

This thesis has been carried out in collaboration with Work4[1] and L2TI[2] as presented in the previous chapter. It was focused on the development of job recommender systems on social networks and systems predicting the performance of job ads posted on Facebook, LinkedIn and Twitter.

[1] http://www.work4labs.com
[2] Laboratoire de Traitement et Transport de l'Information (http://www-l2ti.univ-paris13.fr)

Work4 is a software company founded in 2010 in Paris by Stéphane Le Viet, it has currently two offices: one in Paris (France) and the other in San Francisco (California, USA). The company offers Facebook recruitment solutions and its vision is "Make every Facebook user a recruiter and candidate". During the 3 years of this thesis, Work4 has developed several recruitment tools (based on Facebook, LinkedIn and Twitter) including **Work4us**, **Smart Sort**, **Smart Share**, **ERP**, **Other Jobs**, **Job Card**, **SJS-Recruiter**, **GraphSearch-Recruiter**, **Social Referrals**. This chapter describes the different social networks on which the Work4's recruitment tools are based, the way our data are collected and how the privacy are managed on these social networks in the section 2.2; the section 2.3 presents the different recruitment tools developed by Work4 during these last years.

## 2.2 Social networks on which Work4's applications are based

The (web) applications developed by Work4 are mainly based on Facebook[3], so they need a Facebook account to be set up and managed. Hadley defines a (web) application as a HTTP-based application (software) whose interactions are amenable to machine processing.

The first time a user, client or customer uses a Work4's application, he needs to connect with his Facebook account, then he is asked to authorize the application to access the information in some parts (fields) of his Facebook profile as shown by Figure 2.1. For all our applications, we need at least *Facebook basic permissions access*.

*It is important to note that our applications only use the data that social network users have explicitly authorized to access since without explicit authorizations, we cannot access the data in their profiles.*

In 2012 and 2013, when our data have been collected, we have used *Facebook basic permissions access* which allowed to access information in **Work**, **Education**, **Quote**, **Bio** and **Interests** fields of users' profiles. It also allowed to access friend list of users and information like the first and last name of users. Recently (in 2014), Facebook has changed its privacy settings[4] (see Figure 2.1). As a result, *basic permissions access* are no more sufficient to access information in the fields of Facebook user profiles that are interesting for the task of job recommendation.

After authorizing the application to access the information in some fields of his Facebook profile, the application asks the user, client or customer to link a LinkedIn profile to obtain better job recommendation (see Figure 2.2a). Adding a LinkedIn profile allows the application to access the information in 3 fields: **Headline**, **Educations**, **Positions**. Users can skip this step since it is not mandatory. LinkedIn[5] is more professionally-oriented, as a result, its data might be much more interesting for job recommendation than Facebook ones.

[3] https://www.facebook.com
[4] https://developers.facebook.com/docs/facebook-login/permissions/v2.2
[5] https://www.linkedin.com

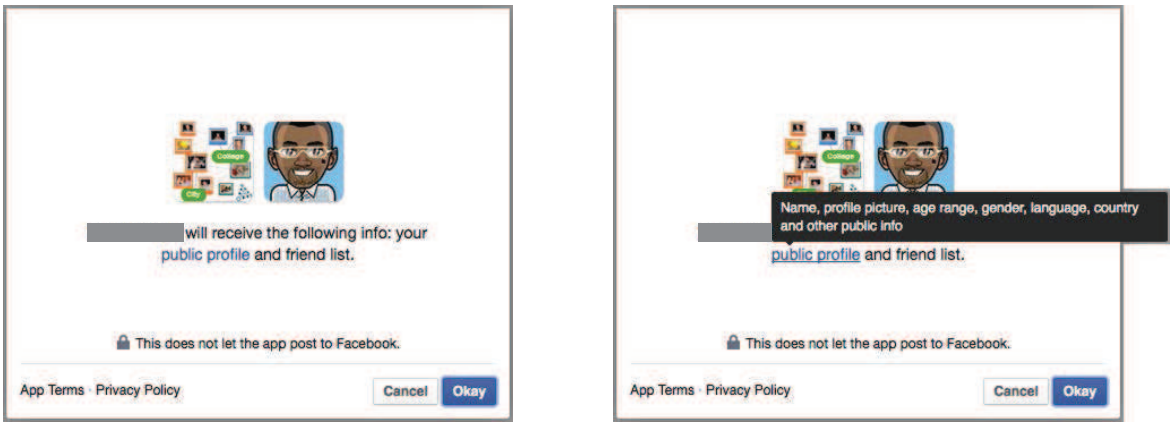
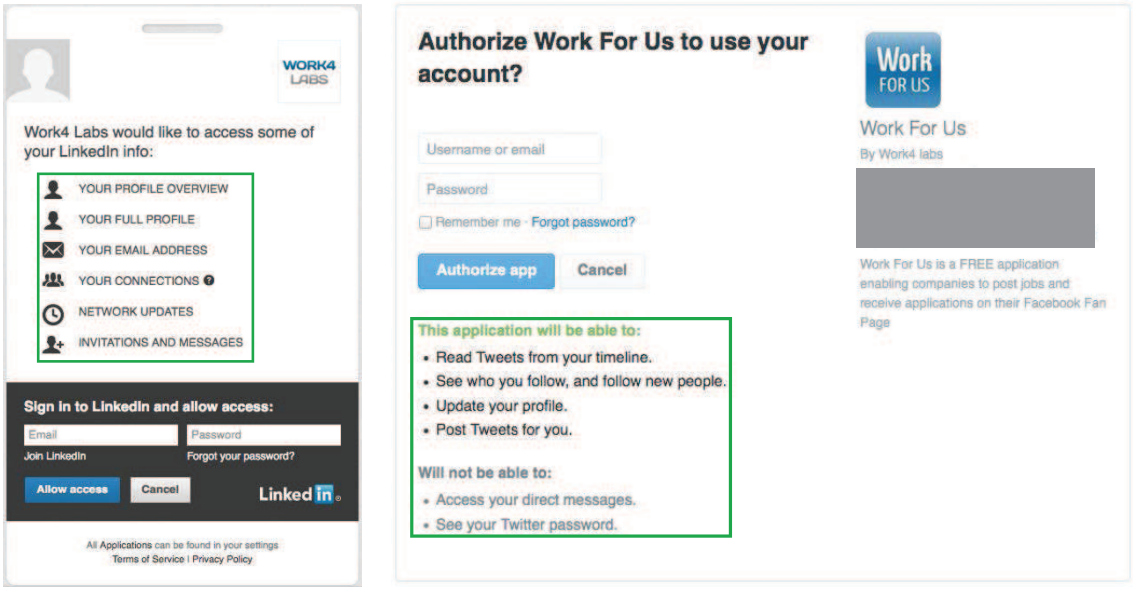



Figure 2.1 – An application requesting a "Basic permissions (public profile)" access to a Facebook profile in March 2015.

Some applications developed by Work4 like *Job Card* (see section 2.3.6), *SJS-Recruiter*, *Social Referrals* (see section 2.3.8) use Twitter[6]: these applications ask users to link their Twitter accounts as shown by Figure 2.2b. This step is not mandatory, so can be skipped too.




(a) LinkedIn. (b) Twitter.

Figure 2.2 – An example of an application requesting permissions to access LinkedIn and Twitter profiles.

After successfully adding his social network profiles, the application creates in our databases, one or more entries (corresponding to the user) which contain:

- The user's Facebook data (Work data, Education, Friend list, ...).

[6] https://twitter.com

- His LinkedIn data (Headline, Educations, Connections, ...) if he has successfully added his LinkedIn account.
- His Twitter data (Followers, ...) if he has successfully linked his Twitter account.

Our customers/clients use the application *Work4us* (see section 2.3.1) to publish their jobs on Facebook which allows us to collect the data related to jobs. The description of Work4 jobs has 3 fields containing textual information: **Title**, **Description**, **Responsibilities**.

## 2.3 Social network-based recruitment tools developed by Work4

This section presents some of the recruitment tools developed by the company Work4 during these last years.

### 2.3.1 Work4us and Work4ads

Work4Us is the core product of Work4, it allows companies to create their own career pages directly on Facebook and publish their job offers on the created pages. Facebook users can thus see, browse companies' jobs and even apply directly on Facebook. Work4 also developed Work4Ads, a tool allowing to create ad campaigns on Facebook with a budget and a targeting system. Figure 2.3 shows the main interface of these two tools, namely Work4Us and Work4Ads.

### 2.3.2 Smart Sort

As presented previously, Work4Us allows companies to create their careers pages (on which they can publish their jobs) on Facebook. The Smart Sort tool is based on Work4us, it allows users to sort jobs on a career page from the most relevant to the less relevant jobs to their Facebook profiles. Figure 2.4 shows an example of a career page using the application Smart Sort.

### 2.3.3 Smart Share

Like the Smart Sort tool previously presented, Smart Share is also based on Work4Us, it allows users to find the top 3 friends who match the most the description of a job on a Facebook career page: user can then share the job with his 3 friends. Using Smart Share, Facebook users can link their LinkedIn profile(s), which allows us to have LinkedIn users in our databases. Figure 2.5 shows an example of using the Smart Share tool to find friends matching the description of a job.

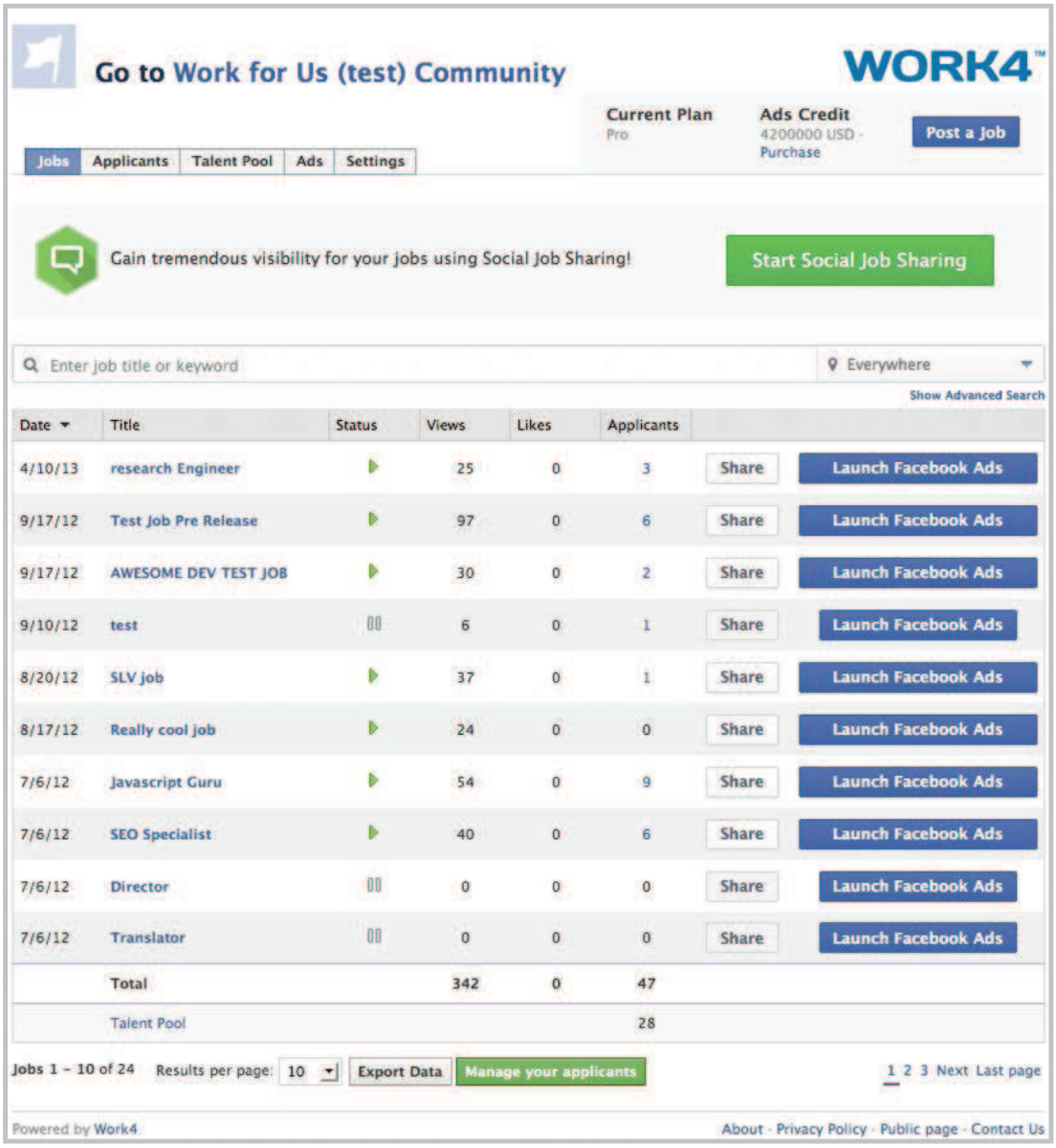



Figure 2.3 – Main interface of Work4Us and Work4Ads that respectively allows companies to create their Facebook career pages and make ad campaigns on Facebook.

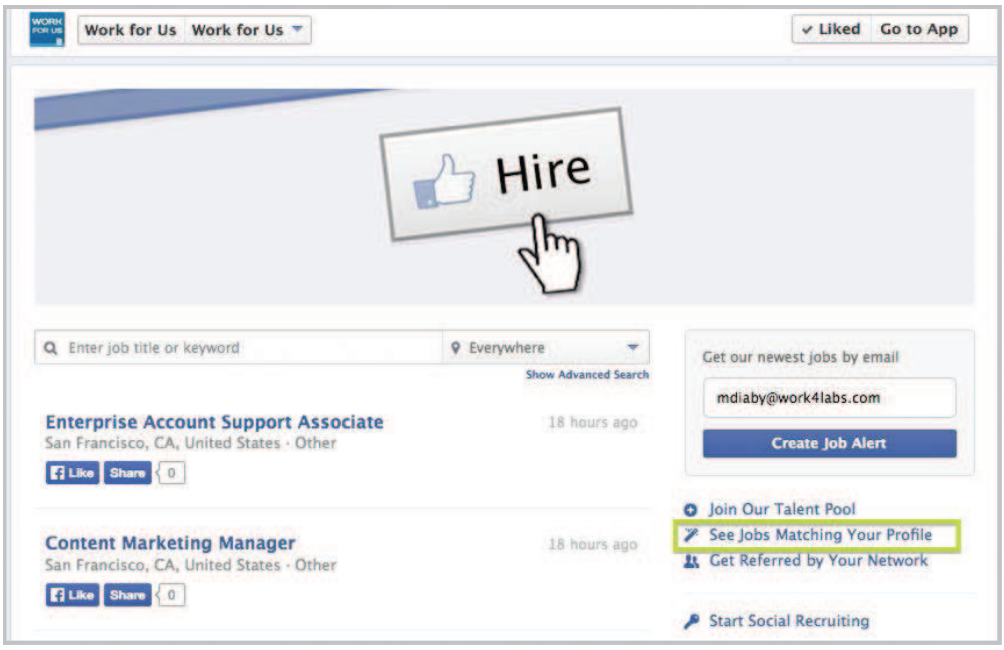



Figure 2.4 – An example of a career page using the Smart Sort tool.


Figure 2.5 – An example of matching friends with a job description using the Smart Share tool.

### 2.3.4 ERP

ERP is a referral tool for HRs: employees of a given company can register their social network profiles (Facebook and LinkedIn) in the ERP system. The system then extracts a list of jobs (available positions at the company) matching the most the profiles of friends of employees (on social networks). Every week, registered employees receive emails containing the names of friends matching the available positions at the company. They can notify the concerned friends by sharing the links of jobs or sending an email, the employee is rewarded by a bonus in case of a successful referral. Figure 2.6 presents an example of emails sent by the ERP system to a user.

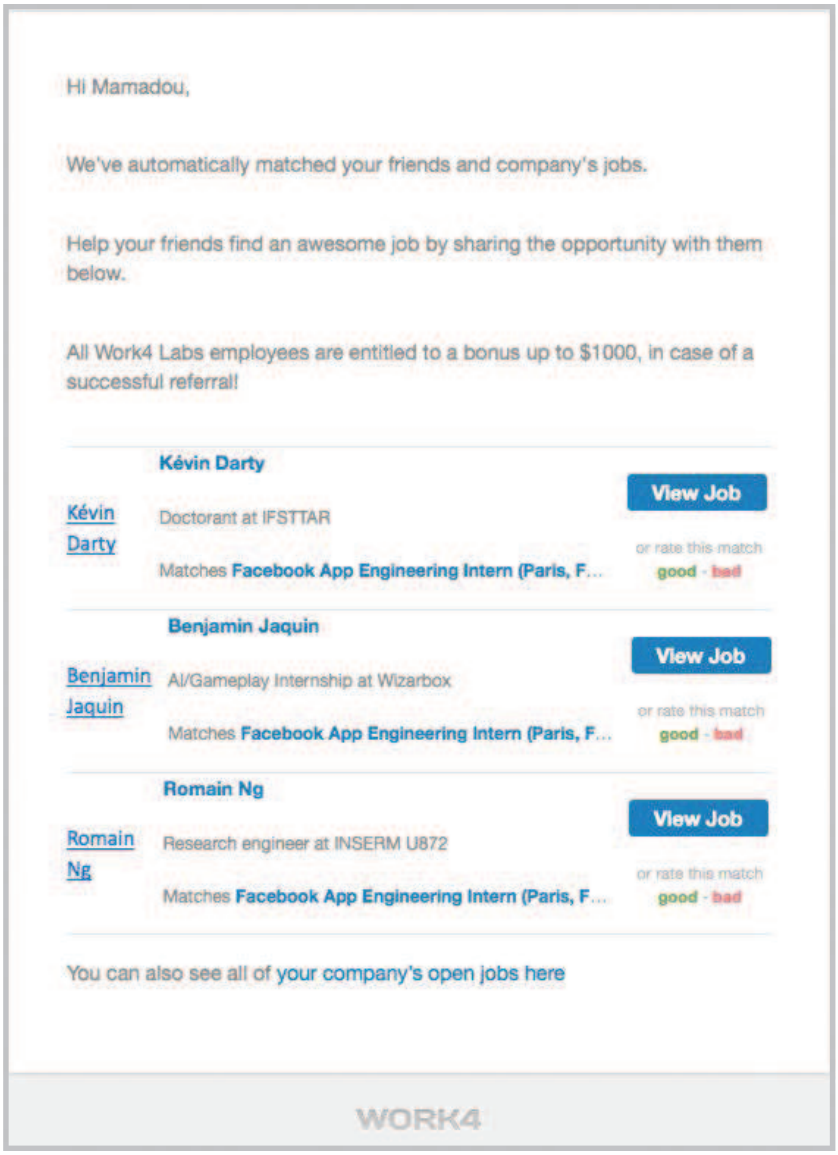



Figure 2.6 – An example of an email sent by the ERP system.

### 2.3.5 Other Jobs

This is simple application allowing users to define a list of their favorite jobs on a Facebook job page. The tool automatically notifies users when new jobs similar to their favorite ones are added to Facebook job pages.

### 2.3.6 Job Cards

This application transforms the description of a job into a card containing the map of the location and other important information (see Figure 2.7). Then, the generated cards can be posted on social networks. Transforming job descriptions into maps may make them more attractive to social network users.

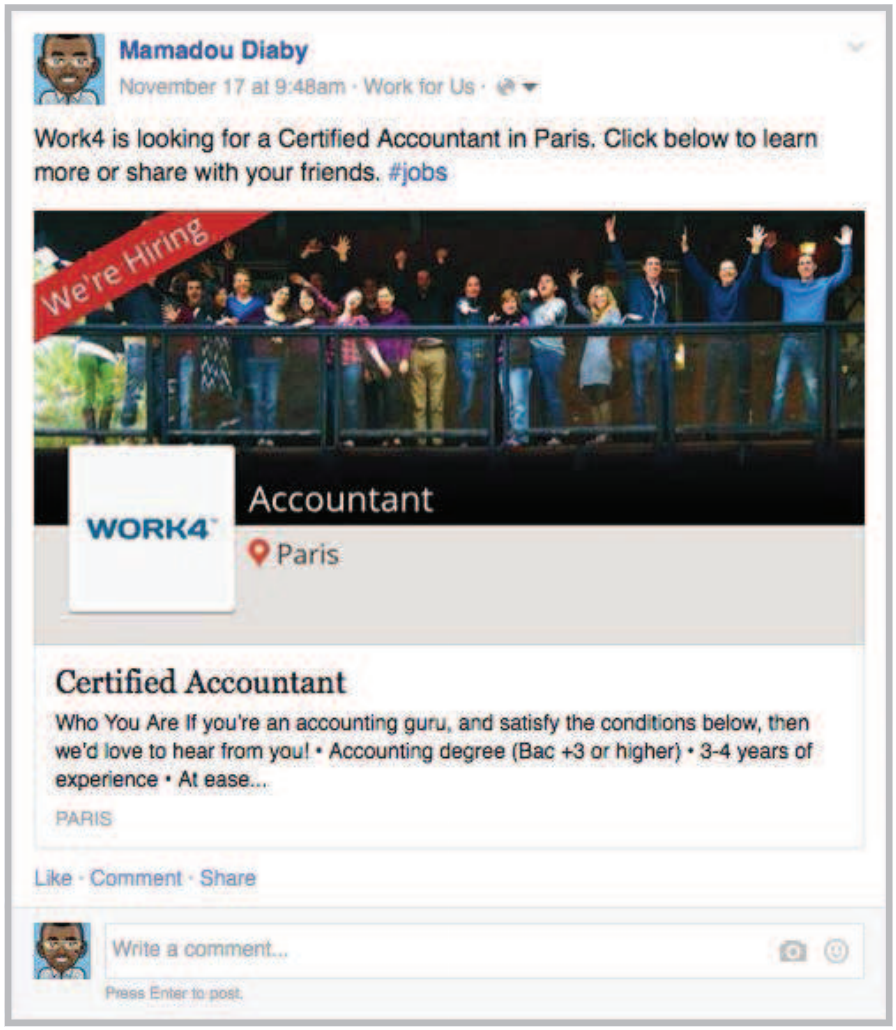



Figure 2.7 – An example of a job card generated by the Job Card tool.

### 2.3.7 GSR: Graph Search for Recruiters

The Graph Search for Recruiter (GSR) is an advanced search tool developed by Work4 to source and search specific talents on Facebook in a specific location, it is based on the Facebook Graph Search[7]. Figure 2.8 shows an example of results that can be obtained using this tool.

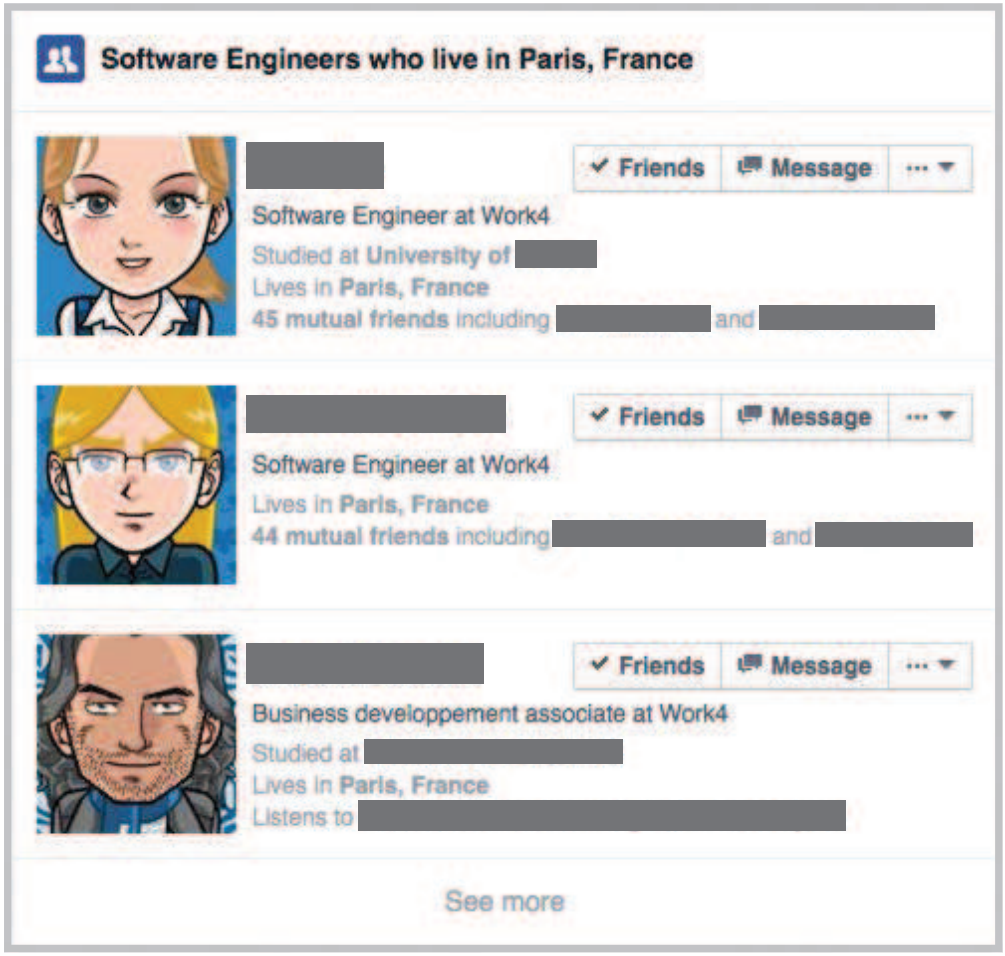



Figure 2.8 – An example of results obtained using GSR to search “software engineers in Paris”.

### 2.3.8 Social Job Sharing for Recruiters (SJS-R) and Social Referrals

Social Job Sharing for Recruiter also known as SJS-R is a tool for HRs developed by Work4. Recruiters register their Facebook profiles and select the list of jobs to share with their friends and the frequency of posts and other filters. The application automatically posts jobs in the defined job list on these social network profiles (Facebook or LinkedIn walls, tweet jobs for Twitter). The posts done by this tool are visible to all friends of the posters.

[7] https://www.facebook.com/help/558823080813217

Social Referrals is similar to SJS-R but is employee-centric. Another difference with SJS-R is that posts are only visible to lists of selected friends of the posters. The friends are selected based on one or more criteria (similarity with jobs, locations, etc.). Figure 2.9 shows the main interface of Social Referrals.

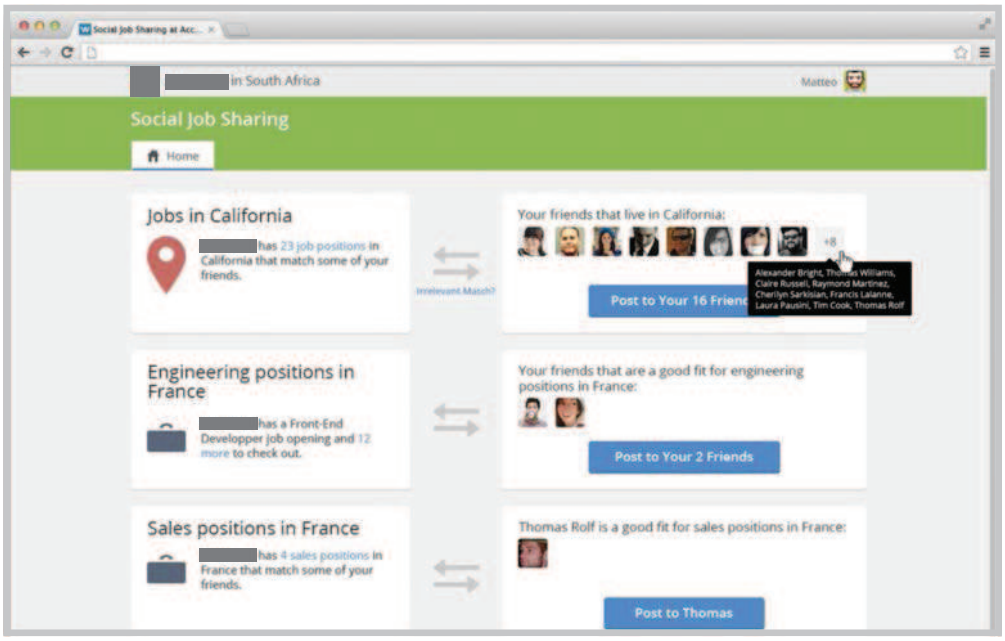



Figure 2.9 – Main interface of Social Referrals.

## 2.4 Conclusion

We presented in this chapter the company Work4, at which we spent the last 3 years doing this thesis in collaboration with the laboratory L2TI. We described how Work4's applications are managing privacy on social networks and how our different data are collected. We chose to not talk about the company's competitors since most of them are/were startups with unstable business models: some went bankrupt while other changed their business model - BranchOut[8] for instance, has been our competitor in the past but is not anymore since they have recently changed their business model.

The next chapter presents the state-of-the-art of recommender systems and tackles all aspects of recommendation engines.

[8] https://branchout.com

# 3 Recommender Systems in the Literature

*"If the Starbucks secret is a smile when you get your latte...*
*ours is that the Web site adapts to the individual's taste."*

- Reed Hastings, Co-founder and CEO of Netflix

## Contents

## 3.1 Introduction

The main objective of this thesis was the development of job recommender systems on social networks. Xiao and Benbasat define recommender systems [Jannach et al., 2011] as software that elicit the interests or preferences of individual consumers for products, either explicitly or implicitly, and make recommendations accordingly. Melville and Sindhwani give a similar definition: "the goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them".

Recommender systems [Jannach et al., 2011] emerged as an independent research area in the mid-1990s [Adomavicius and Tuzhilin, 2005; Hill et al., 1995; Resnick et al., 1994], they are mainly related to statistics [Hastie et al., 2009], information retrieval [Baeza-Yates and Berthier, 1999; Salton et al., 1975], machine learning [de Campos et al., 2010; Hastie et al., 2009; Salakhutdinov and Mnih, 2008], data mining [Han, 1996; Séguela, 2012] and other research fields. They have many industrial applications that help users to deal with information overload. Among these applications, we can cite:

- Recommendation of products on Amazon.com [Linden et al., 2003].
- Netflix' movie recommendations [Bennett et al., 2007].
- Recommendation of jobs to social network users: Work4 [Diaby et al., 2013, 2014], LinkedIn [Sumbaly et al., 2013].

Recommender systems are generally studied using a matrix of $n$ users and $m$ items [Melville and Sindhwani, 2010] containing the values of interests of users for items as shown by Figure 3.1. The task of recommendation then consists of predicting the missing values in this user-item matrix by finding a function $f$ that measures the interest or usefulness of an item for a user. Adomavicius and Tuzhilin formally state this function as follows:

$$\begin{aligned} f &: U \times I \rightarrow R \\ u, i &\mapsto f(u,i) \end{aligned} \tag{3.1}$$

where $U$ and $I$ are respectively the sets of users and items, $u$ is a user, $i$ is an item, $f(u,i)$ is the interest of $u$ for $i$ and $R$ is a totally ordered set.

One recommends to a given user $u$ the list of $TOP\ K$ items (those with the highest $f(u,j)$ for $j \in I$, $K$ (an integer) $\geq 1$. In the particular case of $K = 1$, we recommend to $u$ the item $j^\star = \arg\max_{j \in I} f(u,j)$. In this context, Wang and Blei define two types of recommendations:

1. **In-matrix prediction** which consists of recommending items that have been rated by at least one user.
2. **Out-of-matrix prediction** in which recommendation algorithms recommend all items

even those with no rating, this is a particular case of **cold start recommendation** which consists of recommending items with few ratings or few descriptions to users.

One of the main current challenges in recommender systems remains the cold start recommendation problem.

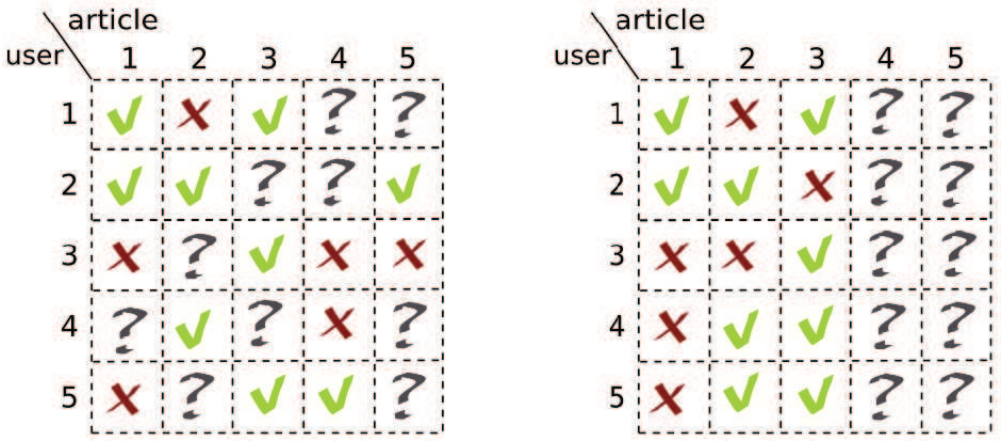



Figure 3.1 – Illustration of the task of recommending scientific articles to users (from [Wang and Blei, 2011]), where ✓ refers to "like", ✗ to "dislike"and **?** to "unknown (to be estimated by a recommender system)". Figure on the left depicts an *in-matrix prediction* while that on the right refers to an *out-of-matrix prediction*.

Adomavicius and Tuzhilin 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.

Contrasted to rating-based recommender systems, preference-based filtering techniques predict the correct relative order of items for a given user. For instance, let assume the following preferences for a given user: iPad 3 ≻ Galaxy S III ≻ 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 ≻ iPad 3 ≻ Galaxy S III ≻ Galaxy tab 2.

In the literature of recommender systems, studies generally focus on rating-based recommendations since a rating-based recommender system can output the relative order of items for a user sorting the items by estimated interests of the user. This thesis is focused on rating-based recommender systems due to the nature of our data.

Recommender systems are generally classified into four categories [Balabanovic and Shoham, 1997; Bobadilla et al., 2013; Kazienko et al., 2011]: content-based methods, collaborative filtering, demographic filtering systems and hybrid approaches. Content-based and Demographic filtering techniques are close in the way they work and are often grouped into the same cate-

gory called *content-based recommendation*. In the literature, we can also find other categories of recommender systems like knowledge-based recommender systems, social recommender systems and trust recommender systems.

Data used in recommender systems are generally collected using at least one of the two following approaches [Porcel et al., 2012]:

- **Implicit collection of data**: one can observe the behavior of users on a platform to gather data that will be used by recommender systems. Example: when a user clicks on a job description or apply to a job on a job-board, one can use this information to make additional job recommendation to him.
- **Explicit collection of data**: in this approach, collected data are based on explicit feedback of users about items. Example: a recommender system can recommend new items to a user using the items he liked in the past.

In this thesis, we use both implicit and explicit collected data to develop our job recommender systems (see section 5.2).

The next sections present in details the different categories of recommender and point out their advantages and limitations.

## 3.2 Collaborative Filtering Recommender Systems

Collaborative filtering systems use the opinion of a community of users similar to the active user (the user for whom the recommendations are made) [Adomavicius and Tuzhilin, 2005; Jannach et al., 2011; Lemire and Maclachlan, 2005; Mnih and Salakhutdinov, 2007]. Conversely, rating predictions of an item involves known ratings of similar users. Historical first recommender systems have been collaborative filtering systems: Grundy system [Rich, 1979] and GroupLens [Konstan et al., 1997]. According to [Melville and Sindhwani, 2010], the term "collaborative filtering" has been introduced by [Goldberg et al., 1992]. Two key assumptions behind collaborative methods [Jannach et al., 2011] are:

1. Preferences of users remain stable and consistent over time: this assumption is not necessarily true in many practical applications of recommender systems in which users' preferences can change, like in [Diaby et al., 2014] when recommending jobs to social network users.
2. Users who had similar preferences in the past, will have similar preferences in the future.

Collaborative filtering methods are generally classified into two main groups [Breese et al., 1998; Wang and Blei, 2011]: latent factor models and neighborhood methods. Latent factor models use dimensionality reduction techniques like SVD [Golub and Reinsch, 1970] to

make recommendations while neighborhood methods rely on similarity between users or items to recommend items. The next two subsections present in details these two groups of collaborative filtering methods.

### 3.2.1 Neighborhood Methods

Neighborhood collaborative recommendation methods are generally based on the use the similarity heuristics to extract the preferences of similar users to make recommendations [Adomavicius and Tuzhilin, 2005; Melville and Sindhwani, 2010]. They directly use the matrix of ratings given by users to items to determine the preference of users. Neighborhood collaborative filtering systems can be divided into two main families [Jannach et al., 2011]: user-based methods and item-based approaches.

The rating $r_{u,i}$ that a user $u$ would give to an item $i$ in **user-based** methods [Wang et al., 2006; Zhao and Shang, 2010] is computed following the 2 steps below:

1. Find a set $N_k$ of the $k$ most similar **users** to $u$ who have rated item $i$ (known as the neighborhood of $u$ in the literature). One can use a set all similar users.

2. Compute $r_{u,i}$ as an aggregation of the ratings that those $k$ users gave to $i$, generally following the equation:

$$r_{u,i} = \delta_u + \frac{\sum_{u' \in N_k} w_{u,u'} \times (r_{u',i} - \delta_{u'})}{\sum_{u' \in N_k} w_{u,u'}} \tag{3.2}$$

   where $w_{u,u'}$ represents a similarity between $u$ and $u'$, $\delta_u$ is an adjustment parameter that is generally set to either 0 or $\overline{r}_u$ and $\overline{r}_u$ is the mean rating given by $u$.

Alternatively, **item-based** collaborative filtering approaches [Lemire and Maclachlan, 2005; Linden et al., 2003] use the ratings given to items similar to active item to make recommendations. Similarities between items are computed using the vector of users' interactions with items, contrasted to content-based systems (see section 3.3) where similarities between items are computed using vectors from their associated information. Formally, in item-based system, the prediction of the rating $r_{u,i}$ that a user $u$ would gave to an item $i$ follows:

1. Find a set $N_k$ of the $k$ most similar **items** to $i$ that have been rated by $u$ (known as the neighborhood of $i$ in the literature). One can use a set all similar items.

2. Compute $r_{u,i}$ as an aggregation of the ratings given to the $k$ items, generally following the equation:

$$r_{u,i} = \delta_u + \frac{\sum_{i' \in N_k} w_{i,i'} \times (r_{u,i'} - \delta_u)}{\sum_{i' \in N_k} w_{i,i'}} \tag{3.3}$$

where $w_{i,i'}$ represents a similarity between $i$ and $i'$, $\delta_u$ is an adjustment parameter that is generally set to either 0 or $\overline{r}_u$ and $\overline{r}_u$ is the mean ratings given by $u$.

In case of negative $w_{u,u'}$ and $w_{i,i'}$, the equations (3.2) and (3.3) can be respectively modified as follows [Adomavicius and Tuzhilin, 2005]:

$$r_{u,i} = \delta_u + \frac{\sum_{u' \in N_k} w_{u,u'} \times (r_{u',i} - \delta_{u'})}{\sum_{u' \in N_k} |w_{u,u'}|} \tag{3.4}$$

$$r_{u,i} = \delta_u + \frac{\sum_{i' \in N_k} w_{i,i'} \times (r_{u,i'} - \delta_u)}{\sum_{i' \in N_k} |w_{i,i'}|} \tag{3.5}$$

In the step 1 of both user-based and item-based collaborative filtering, the similarity between users (using the vectors of ratings they gave to items) or items (using the vectors of ratings given to items by users) can computed using various measures. The most used measures are cosine similarity (see eq (4.15)) and Pearson correlation coefficient (PCC) (see eq (4.16)). Pearson correlation coefficient is generally more popular in collabortive filtering community [Melville and Sindhwani, 2010]: empirical studies showed that it generally performs better in this context [Breese et al., 1998].

In the step 2, aggregation of ratings can be done using either heuristic aggregation functions (memory-based collaborative filtering systems) or aggregation functions learnt from underlying data (model-based collaborative filtering systems) [Adomavicius and Tuzhilin, 2005; Breese et al., 1998]. The most often used aggregation functions in the literature are: *simple average, weighted sum* and *adjusted weighted sum*.

In the *simple average*, we set all $w_{u,v}$ to 1 and $\delta_u$ to 0 in equations (3.2) and (3.3), this is one of the simplest aggregation method in the literature. A more complex aggregation method is the *weighted sum* which uses the equations (3.2) and (3.3) by setting $w_{u,u'}$ to similarity between $u$ and $u'$ and $\delta_u$ to 0. The main problem with *simple average* and *weighted sum* aggregation methods is that they do not take into account the difference of scale of between users' ratings [Adomavicius and Tuzhilin, 2005]: some users tend to give higher ratings to items while other give lower ratings; so the ratings computed using these two aggregation methods can be affected by the difference of scale between users' ratings. To address this problem, the *adjusted weighted sum* has been used: it aggregates different ratings using the equations (3.2) and (3.3) by setting $w_{u,u'}$ to similarity between $u$ and $u'$ and $\delta_u$ to $\overline{r}_u$ where $\overline{r}_u$ is the mean ratings given by $u$.

At this point, a natural question is: *when should we use item-based methods instead of user-based recommender systems?* The answer depends on the number of users and items in the system - generally in industrial applications of collaborative filtering, the number of users is

much more larger than that of items, in that case it might be more interesting to use item-based systems since collaborative filtering methods badly scale (see section 3.2.3).

Additional to the techniques presented above, several other solutions have been proposed to improve the recommendations made by neighborhood methods [Adomavicius and Tuzhilin, 2005; Melville and Sindhwani, 2010]:

- Due to the highly sparsity of the rating matrix, similarity between users or items are computed using few amount of ratings: this could lead to unstable, inaccurate predictors. Breese et al. propose to multiply similarity weights by a signifiance weighting factor to decrease correlations based on few ratings: Melville and Sindhwani called this *signifiance weighting*.
- Breese et al. also propose to replace missing values of ratings by default values when computing similarity between items or users: this is known as *default voting*.
- To favor users with high similarity with the active user, one can use what is called *case amplification* [Breese et al., 1998; Melville and Sindhwani, 2010]: the basic idea is to transform the original weights $w_{u,v}$ to $w_{u,v} \times |w_{u,v}|^{\rho-1}$ where $\rho \geq 1$ is the amplification parameter.

### 3.2.2 Latent Factor Models

Instead of using heuristic similarity functions, collaborative filtering systems can use learnt models to predict the preference of users for items: this is known as model-based recommendation [Breese et al., 1998; de Campos et al., 2008; Kim et al., 2005; Su and Khoshgoftaar, 2009; Xia et al., 2006]. Many techniques have been used in the literature, among them we can cite:

- Xia et al. design a collaborative filtering system based on support vector machine (SVM) which is used to iteratively estimate missing ratings. This could be seen as an improvement of the heuristic consisting of replacing missing values of ratings by default values.
- Breese et al. propose a probabilistic model in which ratings are integer valued. The model is then learnt using Bayesian networks.
- Kim et al. develop neural network-based collaborative methods: an user-based and item-based methods.

Using learnt models leads to models that neatly fit data and therefore an improvement in the quality of recommendations. However, learning a model involves to gather training data which could be difficult in some applications.

Matrix factorization [Koren et al., 2009; Melville and Sindhwani, 2010; Mnih and Salakhutdinov, 2007; Wang and Blei, 2011] is one of the most popular methods of model-based recommender systems, it is part of latent factor models in which users and items are represented in a low-dimensional space. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from items' rating patterns [Koren et al., 2009], it is closely related to singular value decomposition [Golub and Reinsch, 1970; Koren et al., 2009; Melville and Sindhwani, 2010]. The latent factor vectors of users ($W$) and items ($H$) can be computed by minimizing the squared error [Wang and Blei, 2011]:

$$min_{W,H} \sum_{u,i} (r_{u,i} - w_u^T h_i)^2 \tag{3.6}$$

where $r_{u,i}$ is the rating that the user $u$ gave to item $i$, $w_u$ and $h_i$ are respectively the new representations of the user $u$ and the item $i$, $W$ and $H$ are a set of users' new representation and a set of items' new representation respectively.

To avoid overfitting during the learning stage of $W$ and $H$, we can use the regularized squared error (3.6) [Wang and Blei, 2011]:

$$min_{W,H} \sum_{u,i} (r_{u,i} - w_u^T h_i)^2 + \lambda_1 \| w_u \|^2 + \lambda_2 \| h_i \|^2 \tag{3.7}$$

where $\lambda_1$ and $\lambda_2$ are regularization parameters.

Let us consider $R$ as the user-item matrix containing the ratings that users gave to items ($r_{u,i}$), if all $r_{u,i}$ are known, the solution of the equation (3.6) is obtained using the truncated SVD [Golub and Reinsch, 1970; Melville and Sindhwani, 2010]:

$$R = UDV^T \text{(by SVD decomposition)}, \tag{3.8a}$$

$$W = U_k D_k^{\frac{1}{2}}, \tag{3.8b}$$

$$H = D_k^{\frac{1}{2}} V_k^T, \tag{3.8c}$$

where $k$ is the dimension of $W$ and $H$ (number of latent factors).

Unfortunately, in practice the vast majority of $r_{u,i}$ are unknown, in that case, we can use several optimization procedures including gradient descent methods [Luenberger, 1973; Nocedal and Wright, 2006] and alternating least squares [Koren et al., 2009; Melville and Sindhwani, 2010; Young et al., 1976].

Once the new representations of users ($W$) and items ($H$) have been computed, we can estimate missing ratings as follows:

$$\hat{r}_{u,i} = w_u^T h_i \tag{3.9}$$

where $\hat{r}_{u,i}$ is an estimation of $r_{u,i}$.

Sometimes the $w_u^T h_i$ is not sufficient to neatly estimate $r_{u,i}$, this problem is generally addressed by adding a bias $b_{u,i}$ (that contains the specificities related to $u$ and $i$ individually) to $w_u^T h_i$ [Koren et al., 2009]. $b_{u,i}$ is defined as follows:

$$b_{u,i} = \mu + b_u + b_i \tag{3.10}$$

where $b_{u,i}$ is the bias involved in $r_{u,i}$, $\mu$ is the overall average rating, $b_u$ and $b_i$ are respectively the biases for $u$ and $i$.

The prediction equation becomes:

$$\hat{r}_{u,i} = w_u^T h_i + \mu + b_u + b_i \tag{3.11}$$

And the optimization problem is stated as follows:

$$min_{W,H} \sum_{u,i} (r_{u,i} - w_u^T h_i - \mu - b_u - b_i)^2 + \lambda_1 \| w_u \|^2 + \lambda_2 \| h_i \|^2 + \lambda_3 b_u^2 + \lambda_4 b_i^2 \tag{3.12}$$

where $\lambda_3$ and $\lambda_4$ are regularization parameters.

Matrix factorization methods can be generalized in probabilistic models called Probabilistic Matrix Factorization [Mnih and Salakhutdinov, 2007; Salakhutdinov and Mnih, 2008; Wang and Blei, 2011].

Sometimes users' interests for items are constantly changing (temporal dependence), we can adapt the equation (3.11) to make dynamic predictions [Koren et al., 2009]:

$$\hat{r}_{u,i}(t) = w_u^T(t) h_i + \mu + b_u(t) + b_i(t) \tag{3.13}$$

where $w_u$, $b_u$ and $b_i$ are the parameters varying over time.

In some cases, one can trust some ratings than others (ratings from some users could be more reliable than those from other users); in those cases, the equation (3.12) can be adapted [Koren et al., 2009] as follows:

$$min_{W,H} \sum_{u,i} c_{u,i} (r_{u,i} - w_u^T h_i - \mu - b_u - b_i)^2 + \lambda_1 \| w_u \|^2 + \lambda_2 \| h_i \|^2 + \lambda_3 b_u^2 + \lambda_4 b_i^2 \tag{3.14}$$

where $c_{u,i}$ is the confidence of the rating $r_{u,i}$.

Another popular latent factor models is the Nonnegative Matrix Factorization (NMF) [Lee and Seung; Melville and Sindhwani, 2010] in which $W$ and $H$ (presented above) are constrained to be non-negative: this is essentially equivalent to PLSA (probabilistic latent semantic analysis) [Hofmann, 1999, 2004; Melville and Sindhwani, 2010]. The optimization problem in NMF can be solved using the generalized Kullback–Leibler divergence [Melville and Sindhwani,

2010] as follows:

$$min_{W,H} \sum_{u,i} r_{u,i} \log(\frac{r_{u,i}}{w_u^T h_i}) - r_{u,i} + w_u^T h_i \tag{3.15}$$

The representation of users and items in a low-dimensional space in Latent factor models mitigates the cold start recommendation problem but could raises a scalability issue. Latent factor methods are known to generally yield better results than neighborhood methods [Koren et al., 2009; Mnih and Salakhutdinov, 2007; Wang and Blei, 2011].

### 3.2.3 Advantages, challenges and limitations of collaborative filtering systems

Analyzing the way collaborative filtering systems work, one can notice that they can make various recommendations (unlike content-based methods presented in the next section) since they use opinions of community of users to predict recommendations [Adomavicius and Tuzhilin, 2005]. Collaborative filtering systems are very popular and some of the state-of-the-art methods of recommender systems use collaborative filtering techniques [Koren et al., 2009; Melville and Sindhwani, 2010]. Matrix factorization methods are generally flexible and can be adapted to solve various recommendation tasks and use various different sources of data; they can also deal with preferences varying over time [Koren et al., 2009].

We can note that collaborative filtering systems are very interesting but they presents some challenges and limitations discussed below. Among the current limitations and challenges of collaborative filtering systems [Adomavicius and Tuzhilin, 2005; Melville and Sindhwani, 2010], we can cite:

- **New user/item problem**: to make accurate recommendations to a user, collaborative systems need a sufficient amount of ratings this user gave to items, similarly, an item should be rated by a sufficient number of users to be accurately recommended to users. This is the main limitation of collaborative filtering systems and is known as **cold start recommendation problem**. The use of Matrix Factorization methods generally mitigates this problem.

- **Scalability**: memory-based systems have generally a scalability problem in the sense of the systems need to calculate the similarity between all users/items to make recommendations. Latent factor methods also can suffer from this problem.

- **Sparsity**: the number of available ratings is generally very small compared to the total number of couples user-item. As result, the computed similarities between users and items or latent vectors could not be stable (adding of some new ratings can dramatically change prediction models).

## 3.3 Content-based Recommender Systems

Content-based recommender systems [Lops et al., 2011; Pazzani and Billsus, 2007], the roots of which are in information retrieval [Adomavicius and Tuzhilin, 2005; Baeza-Yates and Berthier, 1999; Belkin and Croft, 1992], use the descriptions of items to define their profiles. They generally extract the profiles of users using the ratings that they gave to items in the past. Sometimes, users have only descriptions (instead of ratings), in this case their profiles are constructed using these descriptions [Diaby et al., 2013, 2014]. In content-based recommendation, the profiles of users and items are sets of attributes characterizing them.

Formally, the profiles of items are generally extracted from their descriptions by using bag-of-words models together with weighting functions (see section 4.2.2) and preprocessing techniques (see section 4.2.1) [Balabanovic and Shoham, 1997; Diaby et al., 2014]. They are generally represented as a set of couples (feature, importance of feature). As example, let us consider an item with the following description: "An iPhone is a very innovative smartphone with a touchscreen"; considering "an", "is", "a", "very"and "with"as stop words and using the term frequency weighting function (see section 4.2.2), we can extract the following profile: {*(iPhone, 1), (innovative, 1), (smartphone, 1), (touchscreen, 1)*}.

The profiles of users can be extracted using one of the following methods:

1. One can use the descriptions associated to users to extract their profiles (as done in [Diaby et al., 2013, 2014]): here, the profiles of users are constructed using the same technique as the extraction of the profiles of items described above.
   Example: a user with a description "I like an innovative smartphone with a touchscreen" will have {*(like, 1), (innovative, 1), (smartphone, 1), (touchscreen, 1)*} as profile.

2. An alternative way is to construct the profiles of users using the profiles of items related to them (the items they rated in the past) [Balabanovic and Shoham, 1997]. There are many ways to do that: one method is to compute the vector of users as the average vectors of items related to them: this is similar to the relevance feedback [Rocchio, 1971] in which the initial profiles of users are empty. The following example illustrates this method.
   Example: let us consider a user who rated items $i_1$ and $i_2$ with respectively 1 and -1; the profiles of $i_1$ and $i_2$ are respectively {*(iPhone, 1), (innovative, 1), (smartphone, 1), (touchscreen, 1)*} and {*(Asimo, 1), (innovative, 1), (robot, 1)*}. The profile of the user could be {*(Asimo,* $-\frac{1}{2}$*), (innovative,* 0*), (iPhone,* $\frac{1}{2}$*), (smartphone,* $\frac{1}{2}$*), (touchscreen,* $\frac{1}{2}$*), (robot,* $-\frac{1}{2}$*)*}. Note the term *innovative* with 0 as weight, might be not important for the user.

3. A third way to compute the profiles of users is to combine the profiles obtained using their associated descriptions with the profiles obtained combining the profiles of items related to them [Diaby et al., 2014]: this is known as relevance feedback [Rocchio, 1971] in the literature.

One can find concrete applications of these vector extraction techniques in chapter 5.

In 2011, Lops et al. published a very interesting paper about content-based recommender systems. Adomavicius and Tuzhilin and Melville and Sindhwani define two categories of content-based recommender systems: heuristic-based and model-based systems, presented in the next subsections.

### 3.3.1 Heuristic-based systems

Once the profiles of users and items have been computed, we need to determine the similarity between a given user's and job's profiles to be able to make recommendations. In heuristic-based recommendation, the interests of users for items are computed using similarity measures. In the literature, many similarity measures have been developed, the most used ones are cosine similarity, Pearson correlation coefficient, euclidean distance defined in the section 4.2.5. Cosine similarity is mostly used in content-based recommender systems: it yields better results in item-item filtering systems [Jannach et al., 2011]. The advantage of the heuristic-based systems is the fact that they are simple and easy to develop while their main drawback is the the fact that similarity measures do no fit the data on which recommendations are made. As a result, these systems could yield good results on some kinds of data and fail on others.

### 3.3.2 Model-based systems

Model-based recommender systems are designed to neatly fit the data on which recommendations are made, they use similarity models learnt from recommendations' data based on machine learning techniques like Artificial Neural Networks [Bengio, 2009; Werbos, 1974a], Support Vector Machines (SVMs) [Cortes and Vapnik, 1995; Vapnik, 1998], Bayesian Networks [Pazzani and Billsus, 1997; Pearl, 1988] and Winnow algorithm [Adomavicius and Tuzhilin, 2005; Littlestone, 1988; Pazzani, 1999]. Model-based recommendation techniques generally require to extract the profiles of couples (user, item) from the profiles of users and those of items. They generally yield better results than heuristic-based recommendation [Diaby et al., 2014] but learning efficient similarity models generally involves a large amount of qualitative examples of good and bad recommendations.

### 3.3.3 Advantages, challenges and limitations of content-based systems

Content-based recommender systems are not as popular as collaborative filtering techniques but present some interesting advantages [Adomavicius and Tuzhilin, 2005; Lops et al., 2011] presented below:

1. **User Independence**: in this kind of systems, users are assumed independent in other words, the interests of a given user for items do not affect the interests of other users for

these items. This could be interesting in recommending items to users with particular preferences different from those of users similar to them: recommendations made to users only depend on his own preferences.

2. **Transparency**: recommendations made to users by a content-based system can be directly explained by attributes in their profiles. This is useful when guiding users to improve their profiles to obtain better recommendations.

3. **New Item Recommendation**: contrary to collaborative filtering systems (see Section 3.2), content-based systems are capable of accurately recommending new items that haven't been yet rated by any user: recommendations are based on the descriptions of items, not on the users who liked them.

Analyzing content-based recommender systems, we can note the following drawbacks [Adomavicius and Tuzhilin, 2005; Lops et al., 2011; Melville and Sindhwani, 2010; Pazzani and Billsus, 1997]:

1. **Limited Content Analysis**: using content-based systems, it is tedious to extract interesting attributes from the descriptions of items and users in some cases:

   - It is difficult to extract features from multimedia documents.
   - Sometimes, the vocabulary used in the descriptions associated to users is different from the vocabulary of the descriptions of items; we faced this problem when recommending jobs to Facebook users (who have noisy, fake information in their profiles) [Diaby et al., 2013].

2. **Over-specialization**: content-based recommender systems can be subject to over-specialization since they only recommend to users items that are similar to their associated descriptions or to the items they liked in the past, recommendations made in this context are generally not varied.

3. **New user problem**: content-based recommender systems will fail to make accurate recommendations to users with only few ratings and too small associated descriptions: typically new users. This makes them subject to the famous cold-start recommendation problem.

## 3.4 Demographic and Knowledge-based Recommender Systems

In demographic-based recommender systems, recommendations are made using users' personal attributes (age, spoken languages, gender, income, country, survey responses, purchase history, etc.) [Bobadilla et al., 2013; Gao et al., 2007; Kazienko et al., 2011; Krulwich, 1997; Pazzani, 1999; Porcel et al., 2012]. They are based on of the principle that people with certain common personal attributes will also have common preferences according to [Bobadilla et al.,

2013]. Based on demographic attributes, users can be clustered, the obtained clusters can be used to make recommendations or to better profile users in order to improve the quality of recommendations made to them [Krulwich, 1997]. The main advantage of demographic-based recommender systems is their abilities to recommend items to users with some demographic characteristics. For instance, recommending tea to English people.

Burke defined a knowledge-based recommender system [Blanco-Fernández et al., 2011; Felfernig et al., 2006; Towle and Quinn, 2000] as a system that uses knowledge about users and items to recommend items meeting users' requirements. These systems mitigate the cold start recommendation problem since they use knowledge about users and items when making recommendations but the main difficulty in the development of knowledge-based recommender systems is the acquisition of knowledge which could be tedious and generally necessitates a manual validation. There seems to be relatively few studies about knowledge-based recommender systems compared to the other categories of recommender systems previously presented.

## 3.5 Hybrid Recommender Systems

A hybrid recommender system combines two or more types of recommender systems into a single model [Bobadilla et al., 2013; de Campos et al., 2010; Porcel et al., 2012]. For instance, Vozalis and Margaritis combine demographic data with a collaborative filtering technique to make recommendations. The basic assumption behind them is the combination can allow to benefit from the advantages of each systems while mitigating their drawbacks and limitations.

### 3.5.1 Combination of different categories of recommender systems

Research are generally focused on the combination of a content-based method and a collaborative filtering technique; in that case, [Adomavicius and Tuzhilin, 2005] defined four methods: combination of separate recommender systems, addition of content-based characteristics to collaborative models, addition of collaborative characteristics to content-based models and development of single models using both collaborative and content-based characteristics.

One of the simplest method of hybrid recommendations is to aggregate the two recommendations of a content-based and collaborative filtering systems [Adomavicius and Tuzhilin, 2005; Claypool et al., 1999]: the basic idea here is to develop a content-based system and a collaborative filtering system separately and combine their recommendations using various techniques:

- linear or non-linear combination of recommendation scores,
- default vote: choose recommendations of one of the systems given the context.

A second method of combination is to add content-based characteristics to a collaborative

filtering system: here, recommendations are made in collaborative filtering framework but similarities between users or items are computed using content-based data (instead of using items linked to users or users linked to items as done in pure collaborative filtering recommendation) [Adomavicius and Tuzhilin, 2005; Balabanovic and Shoham, 1997].

A third method of combination is to add collaborative filtering to a content-based system using dimensionality reduction techniques on users' profiles obtained by combining content and collaborative data to create a collaborative view [Adomavicius and Tuzhilin, 2005; Nicholas and Nicholas, 1999].

And finally the fourth method of combination is to develop a single unifying recommendation model that uses both content-based and collaborative data [Wang and Blei, 2011]. These methods of combination are generally based on machine learning methods.

In the rest of this section, we present some examples of hybrid systems (using the fourth method of combination presented above) and then, discuss their advantages and limitations.

Wang and Blei propose a hybrid recommender system that combines a collaborative filtering method to a topic modeling method called LDA (Latent Dirichlet Allocation) [Blei et al., 2003; Blei and Lafferty, 2009]. We compare the job recommender systems we developed in this thesis to the methods presented in [Wang and Blei, 2011]. In the literature, we meet some hybrid recommender systems based on Neural Networks [Christakou et al., 2007; Ren et al., 2008], clustering [Shinde and Kulkarni, 2012] and Bayesian Networks [de Campos et al., 2010]. Hybrid recommender systems generally yield better results than simple recommendation techniques [Adomavicius and Tuzhilin, 2005] but are much more complex to design.

### 3.5.2 Advantages, challenges and limitations of hybrid recommender systems

The sections 3.2.3 and 3.3.3 presented the advantages, limitations and drawbacks of collaborative filtering and content-based methods. Hybrid recommender systems are designed to mitigate some of the limitations of pure approaches, so they present the following advantages:

1. They generally mitigate the problem of cold-start recommendation in collaborative filtering systems by using content-based or demographic data.
2. They generally mitigate both the problems of new item recommendations and that of recommendations to new users.
3. They also mitigate the problem of over-specialization in content-based recommendation.

As we can see, hybrid recommender systems generally mitigate several limitations of pure approaches. However, they can suffer from the following limitations:

1. The Limited Content Analysis of content-based recommendation.

2. They need both user-item rating matrix and content-based or demographic data.

3. They are generally difficult to design and validate since they generally require machine learning algorithms to learn models from underlying data, learning models from data necessitates to have good datasets and carefully fit models to avoid overfitting.

## 3.6 Social and Trust Recommender Systems

*This section is mainly based on a work done in collaboration with other colleagues at L2TI (and recently accepted for publication)* [Bernardes et al., 2014].

A new emerging category of recommender systems is social recommender systems which use both active user's opinion and the opinions of his social connections (friends in social networks) to make recommendations to him. This category of recommender systems is gaining popularity with the rapid growth of social networks in recent years [Facebook, 2015; LinkedIn, 2015; Twitter, 2015].

### 3.6.1 Social Recommender Systems

According to [Tang et al., 2013], social recommendation has been studied since 1997 [Kautz et al., 1997], there are two definitions for social recommendation [Tang et al., 2013]:

1. **A narrow definition**: a social recommendation is any recommendation with online social relations as additional input in other words, augmenting an existing recommendation engine with additional social signals [King et al., 2010; Tang et al., 2013].

2. **A broad definition**: a social recommender system is any recommender system that targets social media domains [Guy and Carmel, 2011; Tang et al., 2013].

We adopt the narrow definition of social recommendation. In this setting, collaborative filtering systems (see section 3.2) could be seen as basic social recommender systems since they use opinions of similar users (implicit social relationships between users) to make recommendations.

It was shown that users generally prefer recommendations made by their friends than those provided by online recommender systems, which use anonymous people similar to them [Sinha and Swearingen, 2001]. Based on this observation, many recommendation methods have been developed to use both users' and their friends' data to make recommendations.

**Memory-based methods** in social recommendation are similar to those in collaborative filtering (see section 3.2), the only difference being the use of explicit social relationships for computing similarities.

Zheng et al. present Social Network Collaborative Filtering (SNCF), which is a modified version of the traditional user-based CF and test it on Essembly.com[1] which provides two sorts of links: friends and allies. In this paper, a user's neighborhood is considered as the set of his friends on the network (first circle). This approach provides results slightly worse than the best collaborative filtering method, but the computation load is much reduced compared to computing the similarity of all pairs of users, the scalability issue of the traditional collaborative filtering is then mitigated. They also show that if the allies are used instead of friends, then the results are as good as collaborative filtering method, but at a much reduced computation cost. Rong Zheng and Ghose use a graph theoretic approach to compute users' similarity as the minimal distance between two nodes (using Dijkstra's algorithm for instance), instead of using the ratings patterns as in traditional collaborative filtering; it is assumed that the influence will exponentially decay as distance increases.

He and Chu observe on a dataset from Yelp[2] that friends tend to give restaurant ratings (slightly) more similar than non-friends. However, immediate friends tend to differ in ratings by 0.88 (out of 5), which is rather similar to results in [Sinha and Swearingen, 2001]. Their experimental setup compares their probabilistic model to a Friends Average approach (whose recommendations are based on average of ratings of immediate friends), a Weighted Average Friends (recommendations of which are based on weighted average of ratings of immediate friends), a Naive Bayes approach and a traditional collaborative filtering method. All methods using the influence from friends achieve better results than pure collaborative filtering in terms of prediction accuracy.

Carmel et al. presents SaND (Social Network and Discovery), a social recommendation system, which is an aggregation tool for information discovery and analysis over the social data gathered from IBM Lotus Connections' applications. For a given query, the proposed system combines the active user's score, scores from his connections and scores between terms and the query. Shang et al. proposes two social recommendation models: the first one is based on social contagion while the second is based on social influence. The authors define the social contagion model as a model to simulate how an opinion on certain items spreads through the social network. Gartrell et al. proposes a group recommendation system in which recommendations are made based on the strength of the social relationships in the active group. The strength is computed using the strengths of the social relationships between pairwise social links (scaled from 1 to 5 and based on daily contact frequency).

**Model-based methods** in social recommenders represent users and items into a latent space vector making sure that users' latent vectors are close to those of their friends. Aranda et al. combined matrix factorization and friendship links to make recommendations: the recommendation score for the active user is the sum of the scores of his friends. Reference [Ma et al., 2011] proposes algorithms which yield better results than non-negative matrix factorization [Lee and Seung], probabilistic matrix factorization [Mnih and Salakhutdinov, 2007] and a

[1] http://www.essembly.com
[2] http://www.yelp.com

trust-aware recommendation method [Ma et al., 2009]. It presents two social recommender systems:

- The first model performs a matrix factorization making sure that the latent vector of a given user is close to the weighted average latent vectors of his friends.
- The second model also performs matrix factorization by minimizing the difference between a user's and his friends' latent vectors individually.

Recently, a new type of social recommender systems has been introduced which rely not upon an explicit social network but upon networks which can be derived from users' behaviors and have thus been named implicit networks. Users will be – implicitly – connected if, for example [Gupte and Eliassi-Rad, 2012], they take pictures in the same locations, they attend the same events or click on the same ads [Ngonmang et al., 2013]. In the literature, such implicit networks are usually derived from the projection of a bipartite graph: users – locations, users – events or users – ads for example. Then the bipartite graph is projected onto a users' graph (and an objects' graph) where the weight of a link can be viewed as the strength of the relationship [Gupte and Eliassi-Rad, 2012].

Finally, a few social recommender systems combine social and content-based techniques. For example, [Diaby et al., 2014] proposes two ways to aggregate users' preferences with those of their friends: enrich users' profiles with those of their friends or aggregate users' recommendation scores with those of their social relationships.

### 3.6.2 Trust Recommender Systems

As explained in [Ma et al., 2011] "trust relationships"are different from "social relationships" in many aspects. Trust-aware recommender systems are based on the assumption that users have taste similar to other users they trust, while in social recommender systems, some of the active user's friends may have totally different tastes from him [Ma et al., 2011].

This was also observed in [Zheng et al., 2008], with the differences between friends and allies, which represents a case where trust is explicitly provided by users. In everyday life, people may ask other people (friends, relatives, someone they trust) for a recommendation. If the person cannot provide sufficient information, he may indicate another person whom he knows which could, and so on. The notion of trust network arises naturally: one tends to have faith in the opinion of people trusted by the people he trusts himself, transitively. Conversely, the notion of social influence has long been used in marketing, relying on the assumption that users are likely to make choices similar to their role-models [Richardson and Domingos, 2002]. The notion of influence can be seen as close to that of trust: when providing a friend with a referral, a trusted user influences his friend. It has long been known that this "word-of-mouth effect" can be used commercially, such as for example in viral marketing.

Recently, it was attempted to incorporate trust or influence knowledge into recommender systems. Beyond the mere expected increase in efficiency, computing trust may also alleviate recurrent problems of traditional recommender systems, such as data sparsity, cold start or shilling attacks (fake profile injections) to bias recommendations.

Several fields of research in computer science deal with trust, from security and Peer-to-peer (P2P) systems [Latapy et al., 2013] to semantic web, and the definition may be more or less broad. It is evaluated between two individuals and intends to model how one individual may trust another one. The trust relationship is directional, i.e. the fact that a user $u_1$ trusts a user $u_2$ at some level $t$ does not necessarily mean that $u_2$ trusts $u_1$ at the same or another level. Trust can be represented by a binary value, 0 for "not trusted user" and 1 for "trusted user", or through more gradual scales [Golbeck, 2005; Guha et al., 2004; Massa and Avesani, 2004] or even with a probabilistic approach [Despotovic and Aberer, 2004; Richardson et al., 2003]. Some models include an explicit notion of distrust [Guha et al., 2004; Ziegler and Lausen, 2005], but most of them ignore it.

For Recommender systems, trust is computed over an explicit social network to increase the information available to generate recommendations. There exist two cases in the literature: either trust is provided explicitly in a trust network, or it has to be inferred.

In an explicit trust network, we propagate and aggregate trust to infer long chains of trust. There are two ways to compute trust [Ziegler and Lausen, 2005]: with a centralized or distributed computation, considering trust either as a global or local metric. Both require propagation and an aggregation steps.

Trust computation relies on the assumption that trust is transitive, i.e. if user $u_1$ trusts $u_2$, and $u_2$ trusts $u_3$, then it may be assumed that $u_1$ would trust $u_3$. Given the existing values of trust between $u_1$ and $u_2$ and between $u_2$ and $u_3$, an estimated value of trust between $u_1$ and $u_3$ is propagated (the most common propagation operator is multiplication of trust values). For long trust propagation chains, length can be taken into account, as a decaying factor [Ziegler and Lausen, 2005] or a threshold to consider only shortest paths [Golbeck, 2005].

Trust computation also requires an aggregation strategy, to combine estimates obtained from different paths from one user to another. Several operators may be used like minimum, maximum, (un)weighted sum and average. Different strategies may also be applied: propagate trust first, then aggregate; or aggregate first, then propagate (the latter allowing easier distributed computation).

In non-explicit trust networks, trust has to be inferred. For example, O'Donovan and Smyth define a profile and item-level trust based on correct previous recommendations.

In explicit trust networks, users provide the system with trust statements for their peers, be it on a gradual scale (Moleskiing [Avesani et al., 2005]), allies (Essembly [Sinha and Swearingen, 2001]) or lists of trusted and non-trusted people (Epinions [Massa and Avesani, 2004]). Then,

to make recommendations to a specific user $u$, trust is estimated between $u$ and the relevant users, in order to weight the recommendation computation. The two main weighting procedures are "trust-based weighted average" and "trust-based collaborative filtering". Trust-based weighted average calculates the rating of a user $u$ for item $i$ as an ordinary weighted average of the ratings of users which have rated $i$; the weights are the estimated trusts for these users. Trust-based collaborative filtering is similar to classic collaborative filtering methods: the prediction formula for an unknown rating is the same, we only replace similarity-based weights by trust-based weights (obtained via propagation and aggregation strategies as described above).

### 3.6.3 Advantages, challenges and limitations of social and trust recommender systems

The main advantage of social and trust recommender systems is they generally mitigate the cold start recommendation problem of recommender systems: using friends or trusted users, we can make accurate recommendations to users even if his profile's quality is poor. Depending on the social network and the task of recommendation, but generally not all friends' preferences are interesting to be considered when making social recommendations: one interesting challenge in memory-based social recommendations is how to compute similarity or trust between users and their friends if the quality of user's profile is poor (user does not rate enough items or his content-based profile is incomplete)? We faced this problem when recommending jobs to Facebook users (with incomplete and noisy profiles) in the paper [Diaby et al., 2014]: as a result, the social recommendation's results were disappointing.

Social recommender systems using only users (trusted) friends to make recommendations can mitigate the problem of scalability of pure collaborative filtering systems since the number friends is generally much more lower than the total number of users in the systems (used to compute neighborhood in collaborative filtering). It worth noting that social or trust recommender systems that propagate similarity or trust to all users suffer from the same scalability issue of collaborative filtering systems.

Using preferences of friends or similar/trusted users allows to make diverse and various recommendations to a user. Currently the results about social recommender systems seem mixed according to [Kantor, 2009]: some papers reported that social recommender systems are no more accurate than classic ones except in special cases [Diaby et al., 2014; Golbeck, 2006] while others argued that they yield better results than classic ones [Groh and Ehmig, 2007].

## 3.7 Performance Metrics for Recommender Systems

For a reminder, we have two main groups of recommender systems (see section 3.1): 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 3.1). 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:

$$P(c) = \frac{\text{number of items correctly affected to c}}{\text{number of items affected to c}} \tag{3.16}$$

$$R(c) = \frac{\text{number of items correctly affected to c}}{\text{number of items that belong to c}} \tag{3.17}$$

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's why $F_\beta$ [Rijsbergen, 1979] has been designed to take into account the Recall and the Precision. $F_\beta$ for a class $c$ is defined as follows:

$$F_\beta(c) = \frac{(1+\beta^2) \times P(c) \times R(c)}{\beta^2 \times P(c) + R(c)} \tag{3.18}$$

We use in this thesis $F_1$ since it is the most often mentioned in the literature; it is defined by:

$$F_1(c) = \frac{2 \times P(c) \times R(c)}{P(c) + R(c)} \tag{3.19}$$

The global Precision, Recall and $F_\beta$ 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's 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 Area Under the Curve Precision-Recall (AUC-PR) can be preferred to the AUC-ROC for unbalanced datasets, the reference [Davis and Goadrich, 2006] presented an comparative study of AUC-PR and AUC-ROC.

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

$$\text{MAE} = \frac{1}{|\Gamma|} \sum_{i,j \in \Gamma} |r_{ij} - \hat{r}_{ij}| \tag{3.20}$$

$$\text{RMSE} = \sqrt{\frac{1}{|\Gamma|} \sum_{i,j \in \Gamma} (r_{ij} - \hat{r}_{ij})^2} \tag{3.21}$$

where $r_{ij}$ and $\hat{r}_{ij}$ are the original and predicted ratings respectively and $\Gamma$ 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 [3], it is defined as follows:

$$\text{MAP@K} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{K} \sum_{k=1}^{K} \frac{C_{uk}}{k} \times 1_{uk} \tag{3.22}$$

where $U$ is the set of users and $C_{uk}$ is the number of correct recommendations in the k first recommendations 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:

$$\text{DCG (b)@K} = \frac{1}{|U|} \sum_{u \in U} \sum_{k=1}^{K} \frac{r_{uk}}{\max(1, \log_b(k))} \tag{3.23}$$

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

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:

$$\text{NDCG (b)@K} = \frac{\text{DCG (b)@K}}{\text{IdealDCG (b)@K}} \tag{3.24}$$

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).

## 3.8 Conclusion

In this chapter, we presented and described different categories of recommender systems developed by researchers in the literature and discussed the advantages and limitations of each category. This study showed that collaborative filtering is very popular and some of the state-of-the-art recommendation engines use collaborative filtering (mainly Matrix Factorization techniques). To complete our study, we also presented the different metrics used to assess the performance of recommender systems. This study of the literature of recommender systems leads us to publish several papers: [Bernardes et al., 2014; Diaby et al., 2013, 2014].

Despite the significant advances made in the development of efficient recommendation algorithms, this research area remains very active. Among the current challenges of recommender systems are facing, we can cite:

- **Scalability**: with the rise of Big Data in recent years, we need recommender systems that are scalable, capable of finding relevant items (among millions of items) for millions of users.
- **Cold start recommendation problem**: many work have been done by researchers to address the cold start recommendation problem, despite the good results obtained by the proposed algorithms, next generations of recommender systems needs to propose much more efficient mechanisms to address the cold start recommendation problem.
- **Aggregation of preference of users from various sources**: current generation of recommender systems use only few sources of data to make recommendations but with the rapid development the Web, we can have many sources of data about users' preferences. The challenge is therefore to aggregate different sources of data about users' preferences (their purchase histories, their survey responses, their data and friends on social networks, the data collected from their smart connected devices, the web sites they browsed, etc.) to make much more accurate, precise recommendations.

After studying the literature of recommender systems, we analyzed our available datasets (see section 5.2) to select the most interesting techniques of recommender systems for our

job recommendation problem, we finally decided to use content-based recommendation techniques for the following reasons:

- We have textual descriptions associated to our jobs and social network users which is very suited to the use of content-based recommendation techniques.
- Our user-job matrices (see section 1.1) are very sparse and we know that collaborative filtering systems generally badly perform in the recommendation tasks with very sparse user-item matrices.
- Using content-based recommendation techniques also allows us to benefit from their advantages discussed in the section 3.3.3.

The next chapter presents how to handle, represent and process textual documents in recommendation tasks, it also describes different machine learning techniques used to develop our recommendation engines.

# 4 Knowledge Discovery in Databases and Data Mining in the Literature

*"Machines will be capable, within twenty years, of doing any work that a man can do."*

- Herbert Simon, 1965
A pioneer in the field of Artificial Intelligence

**Contents**



## 4.1 Introduction

Work done in this thesis are related to data mining [Hall et al., 2009; Kantardzic, 2011], knowledge discovery in databases [Hamel, 2011], information retrieval [Baeza-Yates and Berthier, 1999; Manning et al., 2008] and social network analysis [Carrington et al., 2005; De Nooy et al., 2011; Wasserman, 1994], so we briefly study the state-of-the-art of these different research areas in this chapter.

According to [Fayyad et al., 1996], Knowledge Discovery in Databases also known as KDD refers to the overall process of discovering useful information from databases while data mining refers to a particular step of this process. In the literature, this distinction is generally not clearly made. Many work have been done in data mining and knowledge discovery in databases (which are closely related to machine learning), for instance, Wu et al. present several popular data mining algorithms. As for Information retrieval, Manning et al. define it as follows: "Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)".

## 4.2 Representing and Mining Text

During this thesis, all of our data have been managed using MongoDB [Chodorow, 2013] and MySQL [Schwartz et al., 2012] databases. We used several data mining and information retrieval techniques to develop our recommendation and audience prediction systems. Feldman and Sanger broadly defines text mining as "a knowledge-intensive process in which a user interacts with a document collection over time by using a suite of analysis tools". Text mining systems extract useful information from data sources identifying and exploring interesting patterns [Feldman and Sanger, 2007]. One can notice that text mining and data mining are very close, the main difference is that in *text mining*, interesting patterns are extracted from unstructured textual data in document collections instead of using formalized database records as done in *data mining* according to [Feldman and Sanger, 2007]. In the literature, one can find very interesting papers and books about text mining like [Berry and Castellanos, 2004] and [Hotho et al., 2005].

To develop efficient text mining systems, one has to deal with several difficulties (mainly related to Natural Language Processing):

- How to find the most suited representations for textual resources?
- How to deal with the ambiguity of natural language?
- How to handle polysemy (a same word can have different meanings, depending on the context) and synonymy (different words can have the same meaning)?
- How to deal with missing and noisy data?
- Which data mining and machine learning techniques to use to better find interesting patterns?
- Which performance metrics to use to assess the performance of developed systems?

We start this chapter by studying different techniques to better represent documents (social network users and jobs in our case). Representation techniques generally require preprocessing techniques, we then study these latter techniques. After finding the appropriate

representation method, one generally needs to compute similarity between documents, we end this section by studying similarity measures.

### 4.2.1 Preprocessing techniques

When dealing with textual documents (description of products, profile of a social network users, etc.) in data mining or text mining, one needs sometimes to remove some terms that are not useful for the task he is performing (document categorization, classification, ...). These terms are considered as stop words. Many techniques have been developed to do that [Séguela, 2012; Srividhya and Anitha, 2011; Zaman et al., 2011], Séguela classifies them into 3 categories:

1. 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.
2. Filter out the very high and very low frequency terms, which requires to define two thresholds: one for high frequency terms and one for low frequency ones. To set these two thresholds, one needs to conduct experiments on his datasets.
3. 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.

After removing stop words, it is sometimes interesting to group the different variants of terms together in order to reduce the dimensionality of the problem and then obtain robust systems. There are at least two ways to do that: stemming and lemmatization.

Stemming [Jivani et al., 2011; Lovins, 1968; Paternostre et al., 2002; Porter, 1997, 2001, 1980] is one of the simplest method to group the variants of a term. Lovins defines a stemming algorithm as a computational procedure which reduces all words with the same root (or, if prefixes are left untouched, the same stem) to a common form, usually by stripping each word of its derivational and inflectional suffixes. Stemming algorithms are generally based on a set of reduction rules. Example: fished, fishing → fish (stem) and better → better (stem) (using porter algorithms [Porter, 1997, 2001, 1980]). It is important to note that the stem of a valid word/term is not necessarily a valid word/term. Example: temptation → temptat (using porter algorithms).

A most sophisticated method to group different variants of terms together is lemmatization [Habash et al., 2009; Liu et al., 2012; Schmid, 1994]. Liu et al. define Lemmatization as a morphological transformation that changes a word as it appears in running text into the base or dictionary form of the word, which is known as a lemma, by removing the inflectional ending of the word. Lemma corresponds to the singular form in the case of a noun, the infinitive form in the case of a verb, and the positive form in the case of an adjective or adverb [Liu

et al., 2012]. Example: run, runs, ran, running → run (lemma) and better → good (lemma). Lemmatization algorithms generally use at least 2 steps:

1. Determine the category of the word (masculine form, adj. etc.).
2. Find the lemma (generally using a dictionary).

Using both stemming and lemmatization leads to a lost of information: this problem is much more severe using stemming than lemmatization since stems of valid words are not necessarily valid words. One needs to make a trade-off between the lost of information and robustness of the developed systems. Stemming algorithms [Jivani et al., 2011] are generally faster than lemmatization ones. In the literature, some studies reported that the comparison between lemmatization and stemming depends on the language: lemmatization algorithms generally yield slightly better results than stemming ones for inflected languages like French, Hebrew and Dutch [Kettunen et al., 2005] but for English language, the two techniques seem comparable. Lemaire reported that using lemmatization can decrease the performance of a system in some special contexts. As a conclusion, the choice of using stemming algorithms or lemmatization algorithms or nothing depends on the language and the task one is performing.

### 4.2.2 Vector Space Model

In information retrieval, text mining and data mining 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 and grouping different variants of terms using lemmatization or stemming (see section 4.2.1). 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.

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:

$$\mathrm{TF}_{t,d} = \frac{f_{t,d}}{\max_k f_{k,d}} \tag{4.1}$$

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:

$$\text{Bool}_{t,d} = \begin{cases} 1 & \text{if } t \in d \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{LTF}_{t,d} = \log(1 + f_{t,d}) \tag{4.3}$$

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):

$$\text{IDF}_t = 1 + \log\left(\frac{N}{n_t}\right) \tag{4.4}$$

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:

$$\text{Entropy}_t = 1 + \sum_d \frac{p_d^t \log(p_d^t)}{\log(N)} \tag{4.5}$$

where $p_d^t = \dfrac{f_{t,d}}{\sum_k f_{t,k}}$ is the probability that the term $t$ belongs to the document $d$ and $f_{t,d}$ is defined in eq. (4.1).

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:

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \text{IDF}_t \tag{4.6}$$

where $t$ is a term and $d$ is a document.

Another combination is Log-Entropy defined as follows:

$$\text{Log-Entropy}_{t,d} = \text{LTF}_{t,d} \times \text{Entropy}_t \tag{4.7}$$

**Example**: a document containing the sentence “software engineer” can have *[(“software”, 5.8), (“engineer”, 3.2)]* as TF-IDF vector.

Claveau presents an interesting modified version of TF-IDF called Okapi also known as BM-25,

it better takes into account the lengths of documents and is defined as follows:

$$\text{Okapi}_{t,d} = \text{TF}^{\text{BM-25}}_{t,d} \times \text{IDF}^{\text{BM-25}}_{t} \tag{4.8}$$

where $\text{TF}^{\text{BM-25}}_{t,d} = \frac{f_{t,d} \times (k_1+1)}{f_{t,d} + k_1\left(1 - b + b \times \frac{dl(d)}{dl_{\text{avg}}}\right)}$, $\text{IDF}^{\text{BM-25}}_{t} = \log\left(\frac{N - n_t + 0.5}{n_t + 0.5}\right)$, $k_1 = 2$ is a constant, $b = 0.75$ is a constant, $f_{t,d}$ is the frequency of the term $t$ in the document $d$, $n_t$ is the number of documents that contain the term $t$, $dl(d)$ is the number of distinct terms in $d$ and $dl_{\text{avg}}$ is the average lengths of documents. According to [Claveau, 2012], $\text{TF}^{\text{BM-25}}$ is based on probabilistic model developed in [Jones, 1972] and $\text{IDF}^{\text{BM-25}}$ is a simplification of PRP (Probability Ranking Principle) [Sparck Jones et al., 2000].

### 4.2.3 Ontology-based Representation

Some of the vector models we propose in this thesis are related to ontologies and taxonomies [Diaby and Viennet, 2014b, 2015c]. Ontologies have several definitions depending on the context: the term refers to the study of existence in philosophy while in computer science, it refers to representations useful to explain the world(s) as perceived by a given application [Biemann, 2005]. They are generally classified into three main categories [Biemann, 2005; Sowa, 2010]: formal, terminological and prototype-based ontologies. The different concepts are based on definitions and axioms in formal ontologies while they are defined by typical instances also called prototypes in prototype-based ontologies. In terminological ontologies, concepts are distinguished by using subtype-supertype relations and describing concepts by labels or synonyms [Sowa, 2010].

Taxonomies are collections of entities ordered by a classification scheme and usually arranged hierarchically, this corresponds to the notion of terminological ontologies [Biemann, 2005]. There is only one type of relation in taxonomies: "IS-A"or "PART-OF" according to [Biemann, 2005]. Several studies reported that the use the ontologies could improve the results of an information retrieval system [Aseervatham, 2007].

We use O*NET-SOC taxonomy [National Center for O*NET Development, 2013; Peterson et al., 2001] to develop our different systems, this taxonomy contains several models, tables and collections about worker requirements, experience and occupational requirements, worker characteristics, occupation-specific requirements and occupation characteristics. The appendix B presents a description of the version of O*NET-SOC taxonomy used in this thesis, further explanations and details about different models in O*NET-SOC Taxonomy are available in [National Center for O*NET Development, 2010, 2013; O*NET, 2015; Peterson et al., 2001].

**Example**: the O*NET taxonomy-based vector of a document containing the sentence "software engineer" will look like *[("Software Developers", 1), ("Aerospace Engineers", 0.97), ("Electrical Engineers", 0.97), ..., ("Software Quality Assurance Engineers and Testers", 0.85), ..., ("Web Developers", 0.52), ..., ("Database Architects", 0.24), ..., ("Avionics Technicians", 0.01)]*.

### 4.2.4 Latent Feature Representation

Representing documents into a term space (matrix document-terms) as presented in the section 4.2.2, can lead to very high dimensionality problems since the total number of terms can be very high (sometimes greater than 100,000). It is known that dealing with high dimensional representations generally leads to higher computation costs, that why it is sometimes interesting to find a new representation of documents into a low dimensional latent space (each latent variable is a combination of several terms). This naturally allows a dimensionality reduction since the total number of latent features is much more lower than that of terms. Dimensionality reduction presents some advantages:

- It generally removes from data, noisy components or components that are useless.
- It generally mitigates the sparsity issue.
- It generally speed up different computations (similarity between documents, ...).

However, the Cover's theorem [Cover, 1965] states that data are more likely to be linearly separable in high dimension: in that case Dimensionality reduction could be not interesting.

Latent Semantic Analysis (also known as LSA or LSI) [Deerwester et al., 1990] is one of the most popular methods of latent feature representation, it follows the steps:

1. Construct the matrix document-term using a weighting function (see section 4.2.2) and preprocessing techniques (see section 4.2.1), let $X$ be this matrix.
2. Decompose $X$ as $UDV^T$ ($X \approx UDV^T$) using Singular Value Decomposition (also known as SVD) [Golub and Reinsch, 1970; Koren et al., 2009].
3. Compute the new representation of documents in latent space as $\tilde{X}_k = U_k D_k V_k^T$ where $k$ is the desired number of latent dimensions, $D_k$ is a matrix containing the k first lines and columns of $D$ (with the $k$ highest eigenvalues), $U_k$ and $V_k$ are the matrices containing the $k$ first columns of $U$ and $V$ respectively. Eckart–Young theorem [Eckart and Young, 1936] states that $\tilde{X}_k$ is the best approximation of rank k of $X$.

Latent Semantic Analysis has the ability to somehow address the problem of polysemy and synonymy (presented in the section 4.2) since it groups terms appearing in similar contexts. It also allows to drastically reduce the computation costs by representing documents into a very low dimensional space. This can explain the popularity of this dimensionality reduction method. Despite its popularity, LSA presents some limitations like the problem of scalability (it requires higher computation resources), the problem of setting the number of latent features and the difficulty to interpret latent features. The application of Latent Semantic Analysis to improve the performance our job categorization system (see appendix C.1) gave disappointing results but Séguela reported an improvement of the quality of her proposed systems using

this technique. In the literature, we can find a probabilistic version of LSA called Probabilistic latent semantic analysis (PLSA) [Hofmann, 1999].

Instead of using LSA or PLSA, one can use Principal Component Analysis also known as PCA [Jolliffe, 2005; Schölkopf et al., 1997; Smith, 2002] to project a matrix $X$ (a document-term matrix for instance) into a low-dimensional space. The basic idea of PCA is to find new features (which are the linear combination of the original ones) called principal components that capture the maximum variance of data. It generally necessitates the following steps:

- Step 1 (normalization step): transform $X$ into $Z$ as using the following transformation:

$$Z = X - \overline{X} \tag{4.9}$$

  where $\overline{X}$ contains the means of the columns of $X$.

- Step 2: compute the covariance matrix $C$ of $Z$ as:

$$C = \frac{1}{n-1} Z^T Z \tag{4.10}$$

  where $n$ is the total number of points in $X$.

- Step 3: compute the eigenvectors $V$ of $C$ using SVD [Golub and Reinsch, 1970; Koren et al., 2009] for instance $C \approx UDV^T$

- Step 4: select the $k$ eigenvectors (defined by the matrix $V_k$, each column of which represents an eigenvector) with the highest eigenvalues (defined by the diagonal values of $D$)

- Step 5: finally compute the projection (new representation) of $Z$ into $k$ dimensions by:

$$ZV_k \tag{4.11}$$

PCA has been extended to use kernels (see eq. (4.24)), this extension is known as Kernel Principal component Analysis (KPCA) [Schölkopf et al., 1997, 1998], the basic idea of which consists of mapping original points (with a function $\phi$) into a higher dimension using a kernel function $K(x, x') = \phi(x)^T \phi(x')$ and then follow all the steps of PCA described above except the step 1 and adapting the step 2. In KPCA, to compute the covariance matrix $C$ (see eq. (4.10)) in the step 2, we have 2 cases:

1. One assumes that the projected data into a higher dimension is centered (mean = 0) to simplify the computations, in that case, we adapt the calculation of $C$ as follows:

$$C_{i,j} = \frac{1}{n-1} K(X_i, X_j) \; \forall 1 \leq i, j \leq m \tag{4.12}$$

   where $m$ is total number of dimensions (columns of $X$), $X_i$ and $X_j$ are respectively the $i^{th}$ and $j^{th}$ columns of $X$.

2. In general, data projected into a higher dimension may not be centered, in that case, we only need to replace $K(X_i, X_j)$ in Eq. (4.12) by $\bar{K}(X_i, X_j)$ defined by:

$$\bar{K}(X_i, X_j) = K(X_i, X_j) - \frac{2}{n}\sum_{k=1}^{n} K(X_i, X_k) + \frac{1}{n^2}\sum_{l,k=1}^{n} K(X_l, X_k) \tag{4.13}$$

The matrix form of the equation (4.13) is

$$\bar{K} = K - 2M_{\frac{1}{n}}K + M_{\frac{1}{n}}KM_{\frac{1}{n}} \tag{4.14}$$

where $M_{\frac{1}{n}}$ is a matrix with all elements set to $\frac{1}{n}$.

**Note**: the combination of $\bar{K}(X_i, X_j) = \bar{\phi}(X_i)^T\bar{\phi}(X_j)$ with $\bar{\phi}(X_i) = \phi(X_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(X_k)$ leads to the eq. (4.13).

Another latent representation algorithm is the Latent Dirichlet Allocation (a.k.a LDA) [Blei et al., 2003], a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. We briefly applied LDA to categorize our job descriptions (see appendix C.1) but the results were disappointing. Wang and Blei successfully applied LDA to the task of recommending scientific articles to users.

### 4.2.5 Similarity Measures

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). These similarity measures generally perform better than Euclidean distance (dissimilarity measure) which does not perform very well for high dimensional problems [Ertoz et al., 2002].

For recommender systems, cosine similarity is mostly used in content-based recommendations: 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:

$$\cos(u, v) = \frac{u^T v}{\|u\|\|v\|} = \frac{\sum_{k=1}^{K} u_k v_k}{\sqrt{\sum_{k=1}^{K} u_k^2}\sqrt{\sum_{k=1}^{K} v_k^2}} \tag{4.15}$$

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

In the context of recommender systems, PCC is mainly used in collaborative filtering tech-

niques and is defined as follows:

$$\text{PCC}(u,v) = \frac{\sum_{k=1}^{K}(u_k - \overline{u})(v_k - \overline{v})}{\sqrt{\sum_{k=1}^{K}(u_k - \overline{u})^2}\sqrt{\sum_{k=1}^{K}(v_k - \overline{v})^2}} \tag{4.16}$$

where $u$ and $v$ are the vectors of users or items, $\overline{u}$ and $\overline{v}$ 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:

$$\text{Gaussian}(u,v) = \exp(-\frac{\sum_{k=1}^{K}(u_k - v_k)^2}{2\sigma^2}) \tag{4.17}$$

$$\text{Exponential}(u,v) = \exp(-\frac{\sqrt{\sum_{k=1}^{K}(u_k - v_k)^2}}{\sigma}) \tag{4.18}$$

where $\sigma$ is the parameter of the standard deviation to be set.

Other similarity measures have been developed in the literature, among them [Herlocker et al., 2004; Melville and Sindhwani, 2010; Su and Khoshgoftaar, 2009], we can cite: Spearman rank correlation [Zar, 1998], Kendall's $\tau$ correlation entropy [Kendall, 1938] and adjusted cosine similarity [Sarwar et al., 2001].

An item-based collaborative filtering system using cosine similarity to measure the similarity between users can face the problem of the difference in rating scale between different users. To deal with this problem, the *adjusted cosine similarity* has been designed. The formulations of adjusted cosine similarity and Pearson Correlation Coefficient are close, to better understand the difference, let us consider the rating matrix $R$ containing the rating users gave to items, $U$ the set of users and two items $i$ and $j$. The similarities between $i$ and $j$ using *adjusted cosine similarity* and *Pearson Correlation Coefficient* are computed using the equations (4.19) and (4.20) respectively, one can note the difference between the 2 similarity measures.

$$\text{sim}(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \overline{R_{u,\bullet}})(R_{u,j} - \overline{R_{u,\bullet}})}{\sqrt{\sum_{u\in U}(R_{u,i} - \overline{R_{u,\bullet}})^2}\sqrt{\sum_{u\in U}(R_{u,j} - \overline{R_{u,\bullet}})^2}} \tag{4.19}$$

where $\overline{R_{u,\bullet}}$ is the mean of rating given by the user $u$.

$$\text{sim}(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \overline{R_{\bullet,i}})(R_{u,j} - \overline{R_{\bullet,j}})}{\sqrt{\sum_{u\in U}(R_{u,i} - \overline{R_{\bullet,i}})^2}\sqrt{\sum_{u\in U}(R_{u,j} - \overline{R_{\bullet,j}})^2}} \tag{4.20}$$

where $\overline{R_{\bullet,i}}$ is the mean of rating given to the item $i$.

Recently [Aiolli, 2013] introduced an interesting idea of using asymmetric cosine similarity to recommend items to users. The asymmetric cosine similarity is defined as follows:

$$\text{asym-cos}(u,v) = \frac{u^T v}{\|u\|^{2\alpha}\|v\|^{2(1-\alpha)}} \tag{4.21}$$

where $0 \leq \alpha \leq 1$.

What is interesting in using asymmetric cosine similarity is the fact that one can learn the optimal value of $\alpha$ for a specific problem in order to optimize the performance of developed systems. We used this similarity measure in the paper [Bernardes et al., 2014].

Classic similarity measures (Cosine similarity, PCC, ...) can work on some specific recommendation 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 section 3.3), 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 [Launay, 2012; Pazzani and Billsus, 1997], SVMs [Diaby et al., 2013; Joachims, 1998] are two examples of methods used by researchers.

## 4.3 Statistics and Machine Learning

In this thesis, the recommender systems we design are mainly based on machine learning algorithms. In machine learning literature we have three families of algorithms: supervised learning methods [Breiman, 2001; Chang and Lin, 2011; Cortes and Vapnik, 1995; Friedman, 2001; Friedman et al., 1997; Vapnik, 1998], semi-supervised learning algorithms [Chapelle et al., 2006; Zhu, 2005] and unsupervised learning techniques [Barlow, 1989; Hofmann, 2001].

Supervised learning methods use the couples of (data, label) to fit their models while unsupervised learning techniques only use data with no label to learn their models. In semi-supervised learning, we have a few amount of couples (data, label) and the rest of training data is composed with data whose labels are unknown: semi-supervised learning algorithms use both data with labels and those without labels to fit their models.

In this study, we focus on supervised learning algorithms, as a result, we will not explain in detail the two other families of machine learning algorithms. In supervised learning, we have two types of tasks: classification and regression. All labels associated to training points take discrete values in a classification tasks while they take continuous values in regression tasks.

The next sections present a set of machine learning techniques we used when developing our

job recommendation and ads' performance prediction systems.

### 4.3.1 Support Vector Machines

We use machine learning algorithms to learn models from our collected data to make job recommendation or to predict the audience of job advertisements (see chapters 5 and 6). After studying the literature, we finally choose Support Vector Machines (SVMs) [Cortes and Vapnik, 1995; Vapnik, 1998] since they are known to yield good results on text categorization problems [Joachims, 1998].

For our classification problems, we use the C-SVM form of SVM [Chang and Lin, 2011; Cortes and Vapnik, 1995; Vapnik, 1998] suited to classification tasks. It is formally stated as follows:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & z_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \cdots, l. \end{aligned} \tag{4.22}$$

where $\{(x_1, z_1), \cdots, (x_l, z_l)\}$ is a set of training points, $x_i \in R^n$ is a feature vector, $z_i \in R^1$ is the target output, $n$ is the number of dimensions, $l$ is the size of the training set, $C > 0$ is the regularization parameter, $\phi(\bullet)$ maps $\bullet$ into a higher dimensional space.

**Note**: the equation (4.22) can be adapted to have different regularization parameters $C$ for different classes [Chang and Lin, 2011].

To efficiently handle our different regression problems, we used the adaption of Support Vector Machines [Chang and Lin, 2011; Cortes and Vapnik, 1995] for regression problems called $\epsilon$-SVR (Support Vector Regression). We use the standard form of $\epsilon$-SVR [Chang and Lin, 2011; Vapnik, 1998], formally stated as follows:

$$\begin{aligned} \min_{w,b,\xi,\xi^\star} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i + C \sum_{i=1}^{l} \xi_i^\star \\ \text{subject to} \quad & w^T \phi(x_i) + b - z_i \leq \epsilon + \xi_i, \\ & z_i - w^T \phi(x_i) - b \leq \epsilon + \xi_i^\star, \\ & \xi_i, \xi_i^\star \geq 0, i = 1, \cdots, l. \end{aligned} \tag{4.23}$$

where $x_i$, $z_i$, $l$, $C$, $\phi$ have the same definitions as in the equation (4.22), $\epsilon > 0$ is the epsilon-tube[1] within which no penalty is associated with points predicted within a distance epsilon from the actual value.

The main advantage of SVMs is their ability to build a robust and flexible non-linear model by using parametrized kernels. Vapnik-Chervonenkis theory [Vapnik, 2000] tells us that mapping

[1] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR

inputs into a higher dimensional space (than the original dimension of the input space) often provides a greater classification power. We thus use Linear, Polynomial (Poly) and Radial Basis Function (RBF) kernels in our experiments, they are defined as follows:

$$\text{Linear-kernel}(x, x') = \phi(x)^T \phi(x') = \langle x, x' \rangle \tag{4.24a}$$

$$\text{Poly-kernel}(x, x') = \phi(x)^T \phi(x') = (\gamma \langle x, x' \rangle + r)^d \tag{4.24b}$$

$$\text{RBF-kernel}(x, x') = \phi(x)^T \phi(x') = \exp(-\gamma \| x - x' \|_2^2) \tag{4.24c}$$

where $\phi$ has the same definition as in the equation (4.22), $r$ is a constant, $d$ is the degree of the kernel function and $\gamma > 0$ is the coefficient for RBF and Poly kernels. It is worth noting that it is not necessary to have an explicit definition of the function $\phi$, the only important thing is to have the mathematical definition of the dot product $\phi(x)^T \phi(x')$.

It is usually more interesting to solve the dual forms of both equations (4.22) and (4.23) since $w$ are generally high dimensional vectors. The dual forms of the problems (4.22) and (4.23) are respectively defined [Chang and Lin, 2011] by the equations (4.25) and (4.26).

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to} \quad & z^T \alpha = 0, \\ & 0 \le \alpha_i \le C, \quad i = 1, \cdots, l. \end{aligned} \tag{4.25}$$

where $e = [1, \cdots, 1]^T$ is a vector of all ones, $Q$ is an $l$ by $l$ positive semidefinite matrix defined by $Q_{ij} = z_i z_j K(x_i, x_j)$, $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is a kernel function and $\phi$ has the same definition as in the equation (4.22).

$$\begin{aligned} \min_{\alpha, \alpha^\star} \quad & \frac{1}{2} (\alpha - \alpha^\star)^T Q (\alpha - \alpha^\star) + \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^\star) + \sum_{i=1}^{l} z_i (\alpha_i - \alpha_i^\star) \\ \text{subject to} \quad & e^T (\alpha - \alpha^\star) = 0, \\ & 0 \le \alpha_i, \alpha_i^\star \le C, \quad i = 1, \cdots, l. \end{aligned} \tag{4.26}$$

where $K$ and $e$ have the same definitions as in the equation (4.25) and $Q_{ij} = K(x_i, x_j)$.

After reformulating the equation (4.26), we obtain [Chang and Lin, 2011]:

$$\begin{aligned} \min_{\alpha,\alpha^\star} \ & \frac{1}{2}\left[(\alpha^\star)^T,\ \alpha^T\right]\begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix}\begin{bmatrix} \alpha^\star \\ \alpha \end{bmatrix} + \left[\epsilon e^T - z^T,\ \epsilon e^T + z^T\right]\begin{bmatrix} \alpha^\star \\ \alpha \end{bmatrix} \\ \text{subject to } & y^T\begin{bmatrix} \alpha^\star \\ \alpha \end{bmatrix} = 0, \\ & 0 \le \alpha_i, \alpha_i^\star \le C,\ \ i = 1, \cdots, l. \end{aligned} \tag{4.27}$$

where $y = [\underbrace{1, \cdots, 1}_{l \text{ times}}, \underbrace{-1, \cdots, -1}_{l \text{ times}}]^T$.

**Note**: after solving the equation (4.25), $w^\star$ satisfies $w^\star = \sum_{i=1}^{l} z_i \alpha_i \phi(x_i)$ and therefore the decision function is $sign\left(\sum_{i=1}^{l} z_i \alpha_i K(x_i, x) + b\right)$ (if all $z_i \in \{-1, 1\}$). Solving the equation (4.27) leads to the approximate function $\sum_{i=1}^{l}(-\alpha_i + \alpha_i^\star)K(x_i, x) + b$.

Observing the equations (4.25) and (4.27), we can notice that they are in the form of a *Quadratic Problem with one Linear Constraint*, the general form [Chang and Lin, 2011] of which is:

$$\begin{aligned} \min_{\alpha} \ & f(\alpha) \\ \text{subject to } & y^T\alpha = \Delta, \\ & 0 \le \alpha_i \le C,\ \ i = 1, \cdots, l. \end{aligned} \tag{4.28}$$

where $f(\alpha) = \frac{1}{2}\alpha^T Q\alpha + p^T\alpha$ and $y_i = \pm 1,\ \ i = 1, \cdots, l$.

One can note that directly solving the equation (4.28) maybe difficult since it requires to store all elements of $Q$ which is a fully dense matrix. In this thesis, we used implementations of SVMs based on LIBSVM [Chang and Lin, 2011] which uses a decomposition method proposed by [Fan et al., 2005] and inspired from the Sequential Minimal Optimization (SMO) [Platt et al., 1999]. A decomposition method only modifies a subset of $\alpha$ (denoted as the working set $B$) at each iteration, leading to a smaller optimization subproblem; SMO restricts B to have only two elements [Fan et al., 2005]. The SMO-type algorithm used by LIBSVM to solve the equation (4.28) is described by the algorithm 1, a suitable stopping condition for this algorithm used in LIBSVM [Chang and Lin, 2011] is defined by the equation (4.29).

$$\mathbf{m}(\alpha^k) - \mathbf{M}(\alpha^k) \le \epsilon \tag{4.29}$$

where $\epsilon$ is the tolerance, $\mathbf{m}(\alpha) = \max_{i \in \mathrm{I_{up}}(\alpha)} -y_i \nabla_i f(\alpha)$, $\mathbf{M}(\alpha) = \min_{i \in \mathrm{I_{low}}(\alpha)} -y_i \nabla_i f(\alpha)$,
$\mathbf{I_{up}}(\alpha) = \{t |\ \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}$ and $\mathbf{I_{low}}(\alpha) = \{t |\ \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}$.

The two-variable subproblems (4.30) and (4.31) are solved using a more general problem detailed in the section 6 of the reference [Chang and Lin, 2011]. This reference also describes various techniques to deal with unbalanced classes, probability estimates and multi-class

**Algorithm 1:** SMO-type decomposition method in [Chang and Lin, 2011; Fan et al., 2005]

Find $\alpha^1$ as the initial feasible solution. Set k = 1.

**if** $\alpha^k$ *is a stationary point of the problem* (4.25) **then**

  Stop.

**else**

  Find a two-element working set $B = \{i, j\}$ by **WSS** (see algorithm 2). Define $N = \{1, \cdots, l\} \setminus B$. Let $\alpha_B^k$ and $\alpha_N^k$ be sub-vectors of $\alpha^k$ corresponding to $B$ and $N$, respectively.

  **if** $a_{ij} = K_{ii} + K_{jj} - 2K_{ij} > 0$ **then**

    Solve the following subproblem with the variable $\alpha_B = [\alpha_i \ \alpha_j]^T$

$$\begin{aligned} \min_{\alpha,\alpha^\star} \ & \frac{1}{2}[\alpha_i \ \alpha_j]\begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{bmatrix}\begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \left(p_B + Q_{BN}\alpha_N^k\right)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ \text{subject to } \ & 0 \le \alpha_i, \alpha_j \le C, \\ & y_i\alpha_i + y_j\alpha_j = \Delta - y_N^k\alpha_N^k \end{aligned} \tag{4.30}$$

  **else**

    Let $\tau$ be a small positive constant and solve

$$\begin{aligned} \min_{\alpha,\alpha^\star} \ & \frac{1}{2}[\alpha_i \ \alpha_j]\begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{bmatrix}\begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \left(p_B + Q_{BN}\alpha_N^k\right)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ & + \frac{\tau - a_{ij}}{4}((\alpha_i - \alpha_i^k)^2 + (\alpha_j - \alpha_j^k)^2) \\ \text{subject to } \ & 0 \le \alpha_i, \alpha_j \le C, \\ & y_i\alpha_i + y_j\alpha_j = \Delta - y_N^k\alpha_N^k \end{aligned} \tag{4.31}$$

  **end**

  Set $\alpha_B^{k+1}$ to be the optimal solution of subproblem (4.30) or (4.31), and $\alpha_N^{k+1} = \alpha_N^k$.

  Set $k = k + 1$.

  Go to **step 2**.

**end**

**Algorithm 2:** WSS (Working Set Selection) in [Chang and Lin, 2011; Fan et al., 2005]

**forall the** *t, s* **do**

  Define $a_{ts} = K_{tt} + K_{ss} - 2K_{ts}$, $b_{ts} = -y_t\nabla_t f(\alpha^k) + y_s\nabla_s f(\alpha^k) > 0$,

$$\overline{a}_{ts} = \begin{cases} a_{ts} & \text{if } a_{ts} > 0 \\ \tau & \text{otherwise} \end{cases}$$

**end**

Select

$$i \in \arg\max_t \left\{-y_t\nabla_t f(\alpha^k) \mid t \in \mathbf{I_{up}}(\alpha^k)\right\},$$

$$j \in \arg\min_t \left\{-\frac{b_{it}^2}{\overline{a}_{it}} \mid t \in \mathbf{I_{low}}(\alpha^k), \ -y_t\nabla_t f(\alpha^k) < -y_i\nabla_i f(\alpha^k)\right\}.$$

Return $B = \{i, j\}$

classification in SVMs.

### 4.3.2 OLS, Ridge, Lasso and Elatsic-Net regression methods

Our proposed methods to estimate the audience of jobs posted on social networks (see chapter 6) are mainly related to machine learning algorithms. We propose to model the audience of posts on social networks using a multivariate regression. A simple regression method is the Ordinary Least Squares (OLS) [Craven and Islam, 2011] in which one learns a model $w$ to minimize the sum of squares between the observed values of target variables and predicted ones, formally the problem is stated as follows:

$$w^\star = \arg\min_w \|Xw - z\|_2^2 \tag{4.32}$$

where $w^\star$ is the optimal learnt parameters and $X$ and $z$ are respectively the problem data and the target variable.

Regression models (as described by eq. (4.32)) in high dimensional spaces are subject to overfitting. In order to get a robust solution, with good generalization properties, one has to control the complexity of learnt models, which can be done by introducing a penalty term in the objective function [Cortes and Vapnik, 1995]. Adding $\ell_2$ penalty term to OLS objective function leads to the Ridge Regression [Hoerl and Kennard, 1970], formally defined by:

$$w^\star = \arg\min_w \|Xw - z\|_2^2 + \alpha\|w\|_2^2 \tag{4.33}$$

where $X$ and $z$ have the same definition as in the equation (4.32) and $\alpha \geq 0$ is a parameter controlling the complexity of learnt models.

Another popular regression technique using a penalty term is the Lasso regression method [Haury, 2012; Tibshirani, 1994] which is especially interesting because it uses $\ell_1$ prior as regularizer to reinforce the optimization toward sparse solutions (fewer non zeros parameters), which are both robust and computationally efficient. Mathematically, the problem is stated as follows:

$$w^\star = \arg\min_w \frac{1}{2n}\|Xw - z\|_2^2 + \alpha\|w\|_1 \tag{4.34}$$

where $X$ and $z$ have the same definition as in the equation (4.32), $n$ is the number of instances in $X$ and $\alpha$ has the same definition as in the equation (4.33).

It worth noting that Lasso regression method allows to select relevant features (the weights of which are non zeros in learnt models). Selecting relevant features for the prediction of the audience advertisements allows us to find out which attributes are important and to quantify their importance and then, to be able to explain to our customers why some of their ads perform better than others. Many optimization algorithms to solve the Lasso regression problem exist in literature [Haury, 2012; Yang et al., 2010] but the implementation we used in this thesis is based on a coordinate descent algorithm [Wu and Lange, 2008].

We also use a combination of Ridge and Lasso Regression called Elastic Net[2] [Zou and Hastie, 2005], which is mathematically stated as follows:

$$w^\star = \arg\min_w \frac{1}{2n}\|Xw - z\|_2^2 + \alpha\rho\|w\|_1 + \frac{\alpha(1-\rho)}{2}\|w\|_2^2 \tag{4.35}$$

where $X$ and $z$ have the same definition as in the equation (4.32), $n$ and $\alpha$ have the same definition as in the equation (4.34) and $\rho \in [0,1]$ is the $\ell_1$-ratio allowing to balance $\ell_1$ and $\ell_2$ penalty terms.

The implementation of Elastic Net used in this thesis is based on the algorithm called LARS-EN [Zou and Hastie, 2005] which is inspired from the algorithm LARS (Least Angle Regression) [Efron et al., 2004].

### 4.3.3 Artificial Neural Networks and Deep Learning

An interesting alternative of using SVMs is Deep learning [Bengio, 2013; Erhan et al., 2010; Martens, 2010; Ngiam et al., 2011; Weston et al., 2012]. Deep learning algorithms are mainly based Artificial Neural Networks (ANNs) [Hinton and Salakhutdinov, 2006; Yegnanarayana, 2009] which are inspired from biological neural networks. One of the most popular techniques to train ANNs is backpropagation [Werbos, 1974b]. Training ANNs is often difficult since one has to deal with many hyperparameters tricky to be set like the number of hidden layers/nodes to use and the type of architecture of neural networks, that's why their popularity decreased in 1990s. Since the mid-2000s ANNs are regaining in popularity with the recent development of deep learning algorithms which allows to increase their performance. It is reported that learning with two-layer network (one hidden layer) generally yields good results [Bengio, 2009] but learning with more than two layers (deep architectures) is much more challenging: the references [Bengio et al., 2007; Larochelle et al., 2009] suggest that poor tuning of lower layers might be responsible for the worse results of deep neural networks. Experiments in [Erhan et al., 2009] reported that unsupervised pre-training generally improves the prediction of deep neural networks, this may be explained by the fact that unsupervised pre-training can be seen as a form of regularizer and prior: unsupervised pre-training of deep neural networks seems to be the key to improve the performance of ANNs using deep architectures. There are several architectures for deep learning [Bengio, 2009] in the literature, among them we can cite:

- (Deep) Convolutional Neural Networks [Collobert and Weston, 2008; Fukushima, 1980] are one of the best performing systems according to [Bengio, 2009].
- Restricted Boltzmann Machines [Bengio, 2009] are particular energy-based models which associate a scalar energy to each configuration of variables of interest [Ranzato et al., 2007]. The learning process of energy-based models corresponds to the modification of that energy function so that its shape has desirable properties. One of the most

[2] http://scikit-learn.org/stable/modules/linear_model.html#elastic-net

successful techniques to train RBF is the Contrastive Divergence (an approximation of the log-likelihood gradient) [Bengio, 2009; Carreira-Perpinan and Hinton, 2005].

- Deep Belief Networks [Hamel and Eck, 2010; Hinton et al., 2006] are based on Restricted Boltzmann Machines [Bengio, 2009].
- Stacked auto-encoders [Ranzato et al., 2007] are neural networks consisting of multiple layers of sparse autoencoders [Bengio et al., 2007] in which the outputs of a layer is wired to the inputs of the successive layer.

One can find the recent advances in learning deep architectures in [Bengio, 2009; Bengio et al., 2007; Ciresan et al., 2012; Collobert and Weston, 2008; Deng et al., 2013; Erhan et al., 2010; Martens, 2010; Ngiam et al., 2011; Weston et al., 2012].

Deep learning algorithms are applied to solve a broad range of problems[3] using available datasets like:

- Symbolic Music Datasets[4].
- MNIST (handwritten digits[5]).
- TIMIT Speech Corpus (phoneme classification[6]).
- Recommendation Systems (MovieLens[7]).

We did not use deep learning algorithms in this thesis but they are being more and more popular and can be used to improve the results we obtained with our predictive systems, that's why we described these techniques in this manuscript.

### 4.3.4 Bootstrapping, Cross-validation and Grid search

The machine learning algorithms we presented in the sections 4.3.1, 4.3.2, and 4.3.3 have some parameters called hyper-parameters. To efficiently learn models using a machine learning algorithm, one needs to set the optimal values of hyper-parameters (which generally depends on the dataset he is using). Searching the optimal values for hyper-parameters is called *hyper-parameter optimization*, there are several methods of hyper-parameter optimization in the literature, among them we can cite:

1. Manually assigning some values to hyper-parameters: there is no guarantee that the assigned values are the most suited.

[3] http://deeplearning.net/datasets
[4] http://musedata.stanford.edu
[5] http://yann.lecun.com/exdb/mnist
[6] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1
[7] http://www.grouplens.org

2. Random hyper-parameter optimization:
    - For $k$ times:
        - Fit a model by randomly assigning values to hyper-parameters and assess its performance.
    - Choose the hyper-parameter values leading to the highest performance.
    - Once again, one can notice that there is no guarantee the assigned values are the most suited but this method can be faster than Grid search presented below.
3. A very popular hyper-parameter optimization method is **Grid search** [Bergstra and Bengio, 2012; Szepannek et al., 2010]. Let $n$ be the number of hyper-parameters, the Grid search methodology follows:
    - For each hyper-parameter $p_i$ ($i \in \{1,2,\ldots,n\}$), manually define a list of (all) values that can be assigned to $p_i$, let $l_i$ be this list.
    - For each $n$-tuple $(v_1,\ldots,v_n)$ where $\forall i \in \{1,2,\ldots,n\}, v_i \in l_i$:
        - Fit a model by assigning $v_i$ to $p_i$ for each $i \in \{1,2,\ldots,n\}$ and assess its performance.
    - Choose the hyper-parameter values that yield the highest performance.

    One can notice that the number of times a model is fit and tested is $\prod_{i=1}^{n} |l_i|$, as a result, Grid search suffers from the *curse of dimensionality* because the number of joint values grows exponentially with the number of hyper-parameters [Bergstra and Bengio, 2012].

For our experiments, we used Grid search to optimize the hyper-parameters of our proposed systems in order to make sure to use the values of hyper-parameters suiting the most to our problems.

When using machine learning algorithms, one of the most important steps is to make sure to correctly assess the performance of learnt models. One cannot use the training set to assess the performance of the learnt model since the performance of a model on the training set is not a good indicator of its generalization capacity, that's why it is common to split a dataset into 2 mutually exclusive subsets called training and test sets. In the literature, there are several methods to split datasets into training and test sets, we describe 2 of them below.

One of the most popular techniques to split datasets into training and test sets is the **Cross-validation**. Let $k$ be the number of desired folds, k-fold Cross-validation methods [Kohavi et al., 1995] follow the steps below:

1. Randomly split the active dataset ($D$) into k mutually exclusive subsets $(D_1, D_2, \ldots, D_k)$ called folds of approximately equal size.
2. For each $D_i$ ($i \in \{1,2,\ldots,k\}$):

- Fit a model $M$ using $D \setminus D_i$ (training set).
- Test the model $M$ on $D_i$ (test set) by assessing its performance.

3. The overall performance is computed as the average performance for each fold.

Note that $k$ can take any value ($\geq 2$) but the common values are 2, 3, 5 and 10.

Another interesting technique is the **Bootstrapping** which was introduced by [Efron and Tibshirani, 1994]. Let $D$ be the original dataset containing $n$ instances and $k$ the desired number of bootstraps, Bootstrapping [Kohavi et al., 1995] follows:

1. For each $i \in \{1, 2, \ldots, k\}$:
   - Construct a bootstrap sample $D_i$ by uniformly drawing (with replacement) $n$ instances from $D$.
   - Fit a model $M$ using the bootstrap sample $D_i$ (training set).
   - Test the model $M$ on $D \setminus D_i$ (test set) by assessing its performance.
2. The overall performance is computed as the average performance for each bootstrap.

The number of bootstraps $k$ is generally set to 10, $10^2$, $10^3$ or $10^4$. Note that the probability that a given instance of $D$ is not chosen, after randomly drawing (with replacement) $n$ instances is $(1 - \frac{1}{n})^n$. If $n$ is sufficiently large, $(1 - \frac{1}{n})^n \approx e^{-1} \approx 0.368$ [Kohavi et al., 1995], as a result, the size of test sets using Bootstrapping is closer to $0.368 \times n$.

The reference [Kohavi et al., 1995] reported the 10-fold Cross-validation generally yields good results; we used both Bootstrapping [Diaby et al., 2013] and Cross-validation in this thesis.

## 4.4 Conclusion

This chapter presented different methods from data mining, knowledge discovery in databases, information retrieval we used to develop our recommendation and audience prediction systems. Our developed models are based on Support Vector Machines (see section 4.3.1), Ridge, Lasso and Elastic-Net regressions (see section 4.3.2) but nowadays, Deep learning (see introduction in section 4.3.3), Random Forest [Breiman, 2001] and Gradient Boosting Machine [Friedman, 2001] are gaining popularity and can be used to challenge our different proposed models. One can find many tools for data mining and social network analysis in the literature. Among them, we can cite *Weka* [Hall et al., 2009], a popular data mining tool and *Pajek* [De Nooy et al., 2011], a tool for social network analysis. For our experiments, we used *Scikit-learn* [Pedregosa et al., 2011], a Python module for machine learning.

# 5 Job recommendation to social network users

*"Tell me who your friends are, and I'll tell you who you are."*

- Anonymous

## Contents

## 5.1 Introduction

The study of the literature of recommendation engines (see section 3) has revealed that the efficiency of a recommender system depends on how users and items are represented, that's why we propose, study and evaluate several models for users and jobs to make job recommendations to social network users. Our social network users are defined by two types of data: users' own data and their social connections data (data of their friends).

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

- Social connections data are those from users social connections. List of friends can be cited as an example.

Some our the proposed job recommender systems only use users' own data and the descriptions of jobs to predict users interests for jobs while the others use both users' and their social connections' data.

In this chapter, we tackle the problem of job recommendation to Facebook and LinkedIn users. We start this study by analyzing the data about job recommendation provided by the company Work4 to better understand the profiles of our social network users and job descriptions. As presented in the section 2.2, our Facebook users have authorized Work4 applications to access data in 5 fields: Work, Education, Quote, Bio, and Interests. LinkedIn users only have authorized 3 fields: Headline, Educations, Positions. LinkedIn Educations and Positions fields are almost equivalent to Facebook Education and Work fields respectively. The description of Work4 jobs has 3 fields: Title, Description, Responsibilities.

The comparison of different weighting functions, preprocessing techniques and similarity functions using bag-of-words models allows us to obtain the best combination for our recommendation task. We estimate the importance of user/job fields in the task of recommendation and show how to use machine learning techniques to improve the quality of recommendations. We show that the use of knowledge databases (taxonomies for instance), relevance feedback and social recommendation techniques can mitigate the problem of missing data in the context of job recommendation and therefore improve the job recommendation.

Since matching a user with a job is a multidimensional problem (as presented in the section 1.1), we show how to use knowledge databases to tackle this problem in the context of job recommendation to social network users.

The analysis of the results we obtained allows us to draw key conclusions about job recommendation to social network users.

## 5.2 Job recommendation datasets

Data are the most important thing we needed to validate hypotheses we made. At the beginning of this thesis, we had no dataset we could use to assess the performance of our proposed methods, so we dedicated the first months of the thesis to collect the first datasets. We defined several datasets (based on the data collected by the company Work4) that we studied, analyzed and cleaned up to finally choose 4 of them. 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 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. This dataset cannot be directly used to compute the AUCs of our systems since it only contains label 1.
2. **Review**: it contains recommendations made by Work4's systems that have been manually validated by two different teams of the company.
3. **Validation**: it contains recommendations made by Work4's systems that have been manually validated by one team of the company.
4. **ALL**: this dataset is the union of Candidate, Review and Validation datasets.

One interesting thing to note is the fact that each job is associated to a job page (as shown by Figure 5.1) which generally represents a page of a company (in this context, 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. This impacts our experimental protocols when splitting datasets into training and test sets (see section 5.3). Table 5.1 depicts the summary statistics from our datasets while Table 5.2 shows the percentage of empty fields in each dataset. We can notice that most social network users do not completely fill the fields of their profiles that are interesting to extract their preferences for jobs, this problem is more severe for Facebook than LinkedIn profiles. Our recommender systems must thus deal with incomplete (and very noisy) data and those using machine learning based models also need to deal with unbalanced datasets since the proportion of label 0 is much more higher than that of label 1 in ALL, Validation and Review datasets. Recall that labels 1 et 0 respectively denote a matching and mismatching between a user and job offer.

After filtering out stopwords using lists of stopwords defined by Work4, we obtained a dictionary with 26,995 terms (stemming allows us to reduce to 17,954 terms and using lemmatisation leads to 11,759 terms). One can use the TF-IDF scores to select the most interesting terms in users' and jobs' profiles as done [Blei and Lafferty, 2009; Wang and Blei, 2011]. We are focused on English and French: if the language of a user or a job is different from French

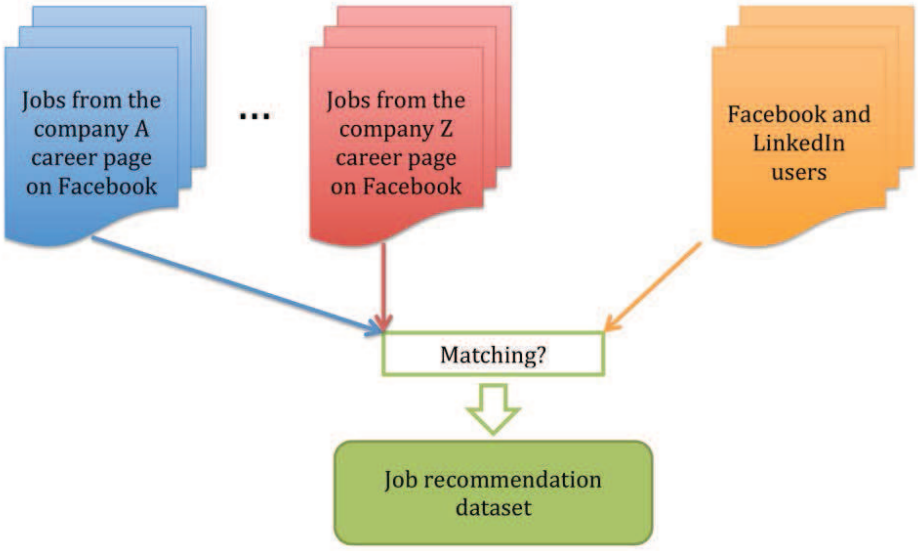



Figure 5.1 – Generation process of our job recommendation datasets. Each job is linked to a job page corresponding to the career page of a company on Facebook. Jobs from the same page are likely similar and probably belong to the same job categories.

| | | Datasets | | | |
|---|---|---|---|---|---|
| | | ALL | Validation | Review | Candidate |
| | Total number of instances | 86,524 | 54,247 | 14,414 | 17,863 |
| | Distinct users | 41,303 | 27,408 | 7,572 | 9,232 |
| | Distinct jobs | 10,527 | 1,326 | 2,171 | 7,699 |
| Proportion | label 0 | 0.70 | 0.91 | 0.75 | 0.00 |
| | label 1 | 0.30 | 0.09 | 0.25 | 1.00 |
| | Facebook users | 0.44 | 0.27 | 0.30 | 0.97 |
| | LinkedIn users | 0.56 | 0.73 | 0.70 | 0.03 |

Table 5.1 – Summary statistics of our datasets: number of instances, proportion of label 0/1 and instances linked to Facebook/LinkedIn users. ALL dataset is the union of the 3 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.

| | | Datasets | | | |
|---|---|---|---|---|---|
| | Fields | ALL | Validation | Review | Candidate |
| Facebook | Bio | **61.1** | 46.0 | **69.6** | **72.2** |
| | Education | **68.6** | **64.0** | **64.0** | **73.7** |
| | Interests | **73.0** | **74.4** | **79.3** | **70.9** |
| | Quotes | **90.0** | **98.9** | **97.8** | **80.9** |
| | Work | 21.0 | 12.4 | 7.0 | 30.6 |
| LinkedIn | Educations | 16.1 | 15.8 | 16.1 | 14.8 |
| | Headline | 0.8 | 0.7 | 1.1 | 2.6 |
| | Positions | 0.2 | 0.1 | 0.1 | 5.6 |
| Jobs | Description | 0.5 | 0.1 | 0.0 | 0.7 |
| | Responsibilities | **55.5** | **82.1** | **66.9** | 49.4 |
| | Title | 2.3 | 1.2 | 0.6 | 2.8 |

Table 5.2 – Percentage of empty fields in our datasets; in bold, fields empty at more than 50%. We note a very high percentage of empty fields for Responsibilities compared to Description and Title, this is due to fact that sometimes the field Responsibilities is not clearly indicated since it is merged with the Description field.

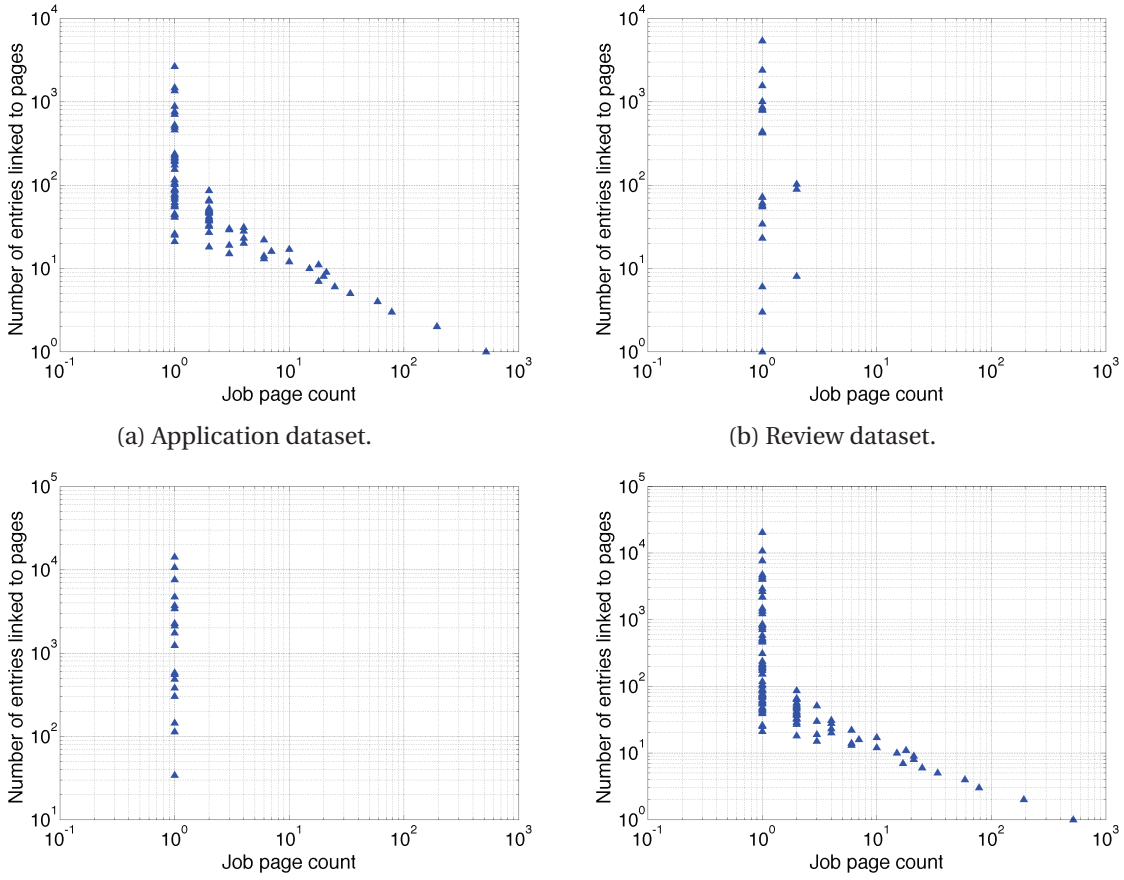
and English, its language is set to "other". As explained above, the entries in Review and Validation datasets are obtained by manually annotating some recommendations made by our systems. To annotate a recommendation of a job to a user, the annotators have all the available information about the job (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 annotators as an experience mismatch.
- -1: cannot decide if the user matches the job or not; these entries are not used in this study.

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.




(a) Application dataset.

(b) Review dataset.

(c) Validation dataset.

(d) ALL dataset.

Figure 5.2 – Distribution of job pages in our job recommendation datasets.

Figure 5.2 shows the number of entries in a given dataset linked to a given job page. Since a job page is generally linked to an organization which is generally linked to an industry/job category, this Figure allows us to know how varied (in terms of job industries/categories) are our datasets. We note that some datasets contain huge job pages, for instance, we can see that ALL dataset contains a job page linked to more 20,000 entries: some datasets are not varied enough in terms of job industries/categories.

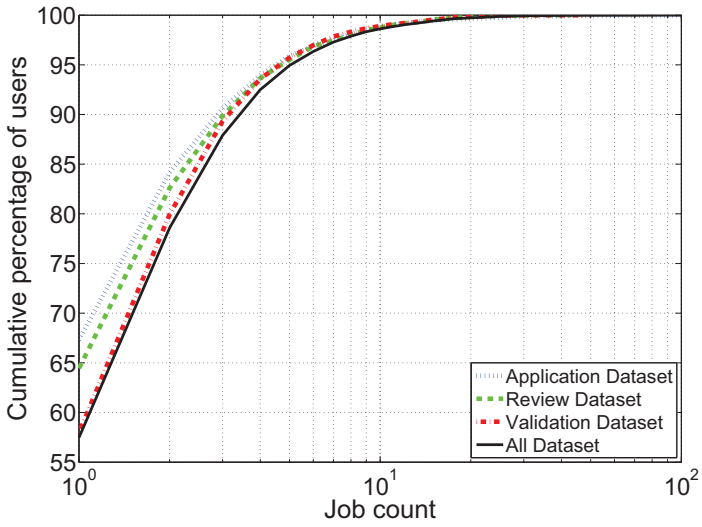
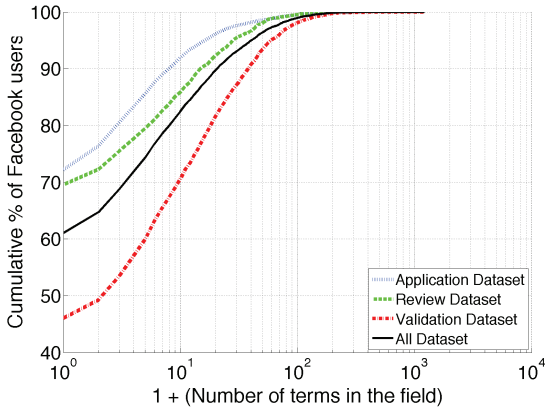



Figure 5.3 – Distribution of jobs per user in our job recommendation datasets.

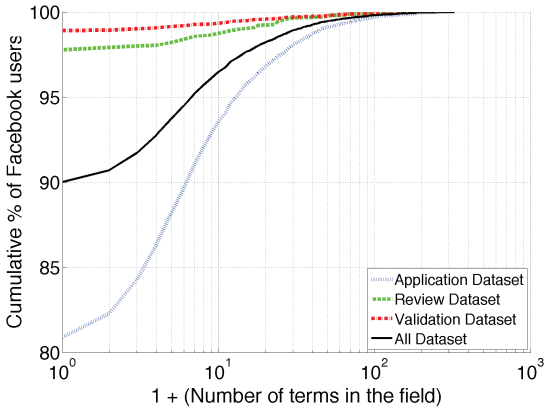
On Figure 5.3, one can note that a very majority of users (more than 90%) are linked to at most 5 jobs, as a result, the user-job matrices for different datasets are very sparse. Using Table 5.1, we obtain the following sparsity rates: 99.98% for ALL dataset, 99.85% for Validation dataset and 99.91% for Review dataset.

Figures 5.4, 5.5, 5.6 respectively show the distribution of the distinct number of terms in the profiles of Facebook users, LinkedIn users and jobs after vectorizing and removing stop words, they allow to make the following key observations:

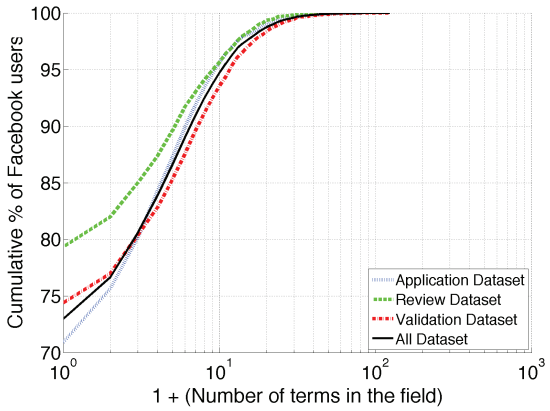
- Our Facebook users do not completely fill the fields that are interesting for job recommendation. For instance, more than 65% of our Facebook users have no information in the Education field. More than 80% of our Facebook users have no entry in the Quotes field.
- Our LinkedIn users provide more information about their educations and positions (than Facebook users). These data are useful when recommending jobs to them.
- The title and description fields of jobs are well filled but responsibilities field is empty for many jobs, this is due to the fact that responsibilities field data are sometimes merged with description field data.



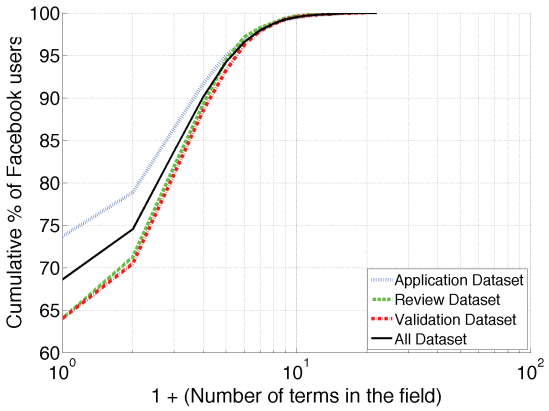

(a) Facebook Bio field.



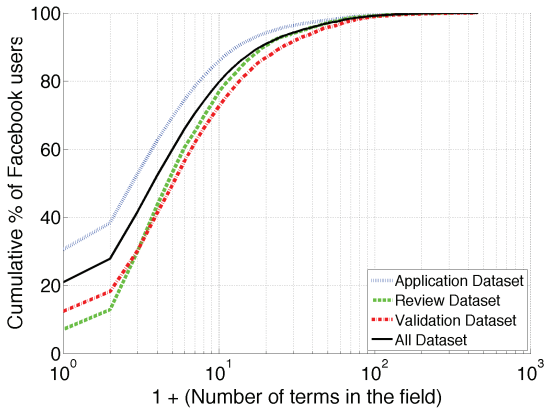

(b) Facebook Quotes field.




(c) Facebook Interests field.

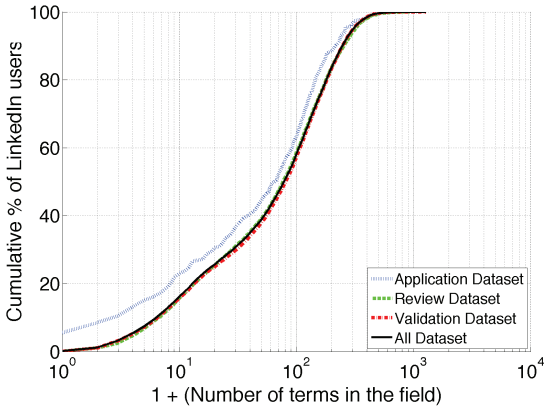



(d) Facebook Education field.

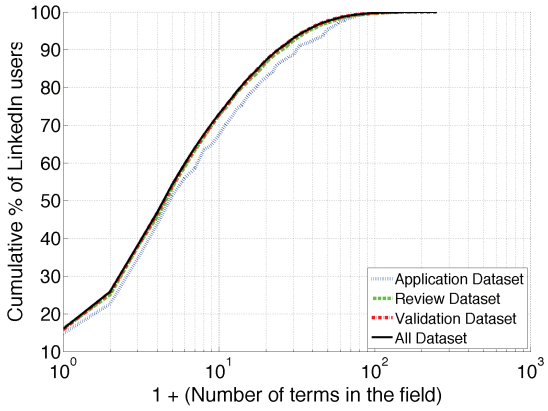



(e) Facebook Work field.

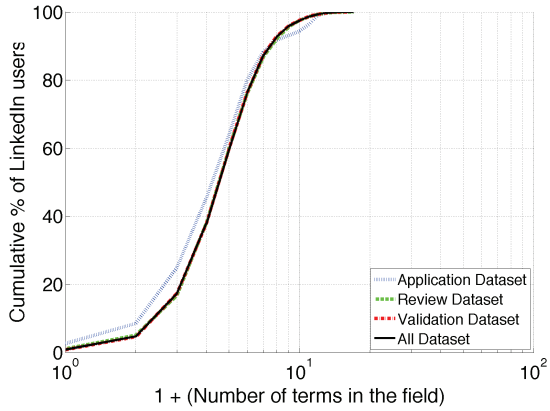
Figure 5.4 – Distribution of terms in Facebook fields in our job recommendation datasets.




(a) LinkedIn Positions field.

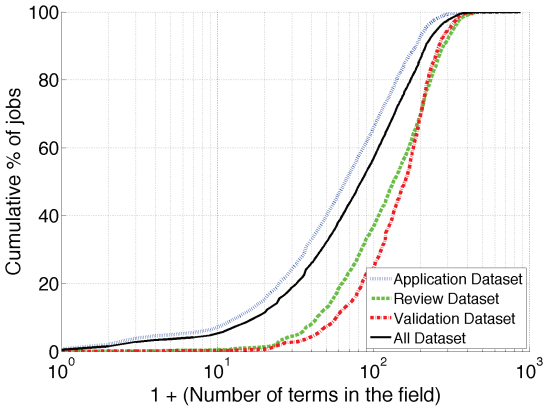



(b) LinkedIn Educations field.

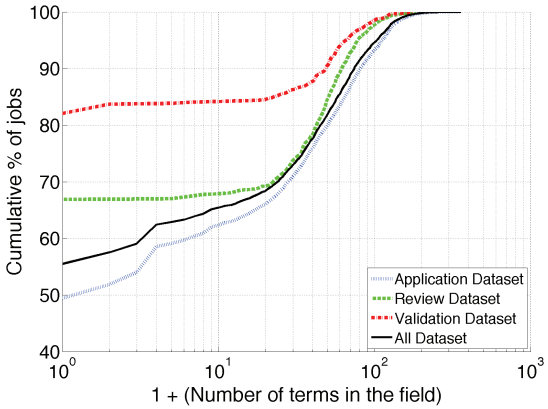



(c) LinkedIn Headline field.

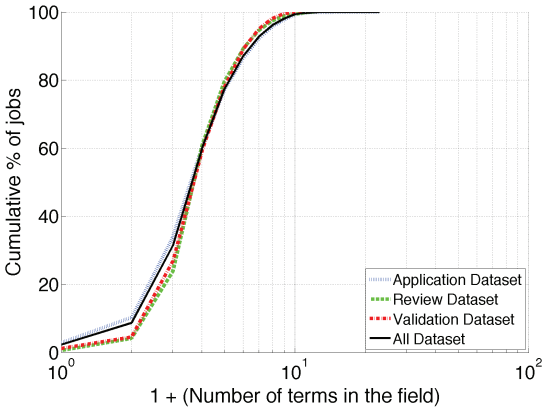
Figure 5.5 – Distribution of terms in LinkedIn fields in our job recommendation datasets.




(a) Job Description field.




(b) Job Responsibilities field.




(c) Job Title field.

Figure 5.6 – Distribution of terms in Job fields in our job recommendation datasets.

## 5.3 Experimental protocols

All our scripts are written in Python and are mainly based on scikit-learn[1] [Pedregosa et al., 2011] implementation of different machine learning algorithms and performance metrics, the implementation of SVM is based on LIBSVM [Chang and Lin, 2011]. The different experiments have been run on Intel Xeon 2.00GHz (with 12 cores). For all proposed methods, we learn a model (if needed) on a training set and tests are done on the corresponding test set for different datasets using 10-fold cross-validation (see section 4.3.4). The hyper-parameters of different algorithms (we use) have been optimized using Grid search (see section 4.3.4).

We use AUC-ROC (see section 3.7) as performance metric for all our job recommender systems. Using cross-validation allows us to compute the confidence intervals for our proposed job recommender systems.

In our industrial context, the ideal procedure of splitting our datasets into training sets and test sets could be: split a dataset into two subsets mutually exclusive in terms of job pages as shown by Figure 5.7a. This procedure of splitting leads to learn a model from labeled data linked to a set job pages and to apply the learnt model to make recommendations on new job pages (new clients/customers for instance); 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 (as presented in the section 5.2) that makes difficult the use of this procedure. Since we have several huge job pages in our datasets, splitting a dataset into a training and test sets using the above procedure leads to 2 scenarios:

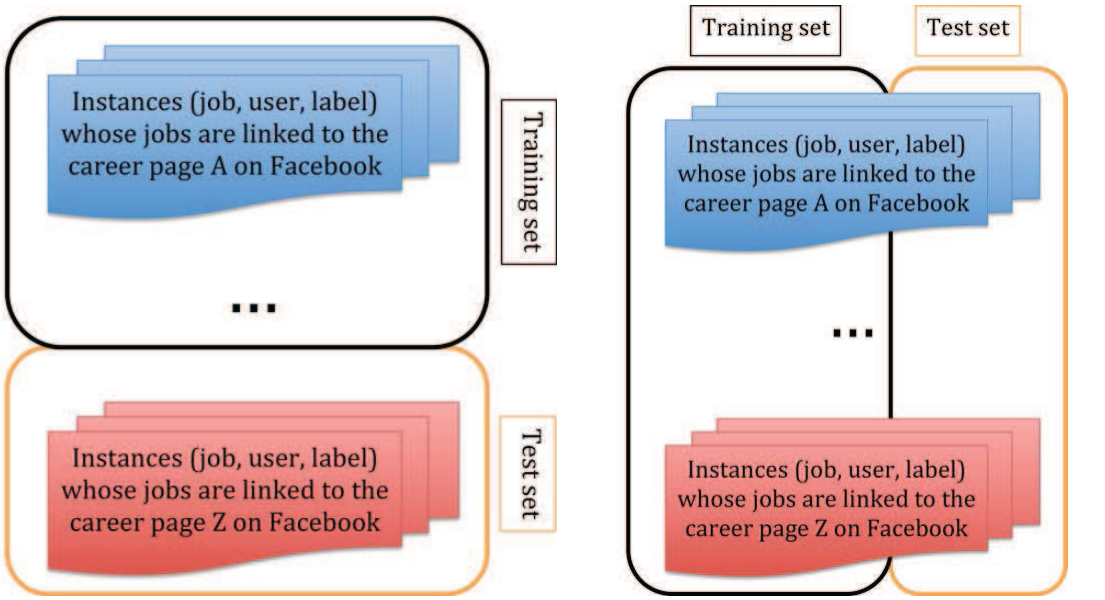
1. The training set contains exclusively huge job pages: we learn a model on few pages that cannot yield good results on new pages.
2. The training set contains exclusively pages with few linked examples: the learnt model cannot also yield good on specific huge pages.

Finally, we use an alternative method (see Figure 5.7b) based on the 10-fold cross-validation: we randomly split the active dataset into 10 subsets mutually exclusive making sure that each subset contains approximately 10% of instances linked to each job page. For each fold, we use 1 subset as test set and the 9 other subsets 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 thesis) 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 (job categories), for instance it will probably make bad recommendations for clients whose categories of jobs were not in the datasets used to learn the model.

[1] http://scikit-learn.org

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 the client.




(a) Ideal procedure (for Work4) of splitting datasets into training and test sets.

(b) Procedure of splitting datasets into training and test sets we used in this thesis.

Figure 5.7 – Procedures of splitting datasets into training and test sets.

## 5.4 Bag-of-words model-based job recommendation

This section presents the first job recommender systems we develop which are based on bag-of-word models (see section 4.2.2) to recommend jobs to social network users. First of all, we consider each social user as a document and his fields as sub-documents, we do the same for jobs. It is worth noting that fields of our documents only contain textual information and we do not directly consider the temporal aspect of job recommendation (we assume that users' preferences for jobs remain stable over time).

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 different weighting functions (see section 4.2.2). We also filter out stop words using lists of stop words defined by Work4. The vector of a document is computed as a weighted sum of the vectors of its different fields where each weight is the importance of the associated field. Figure 5.8 shows how the aggregation is done for Facebook users, LinkedIn users and Jobs.

The rest of this section presents and analyzes the different job recommender systems we proposed.

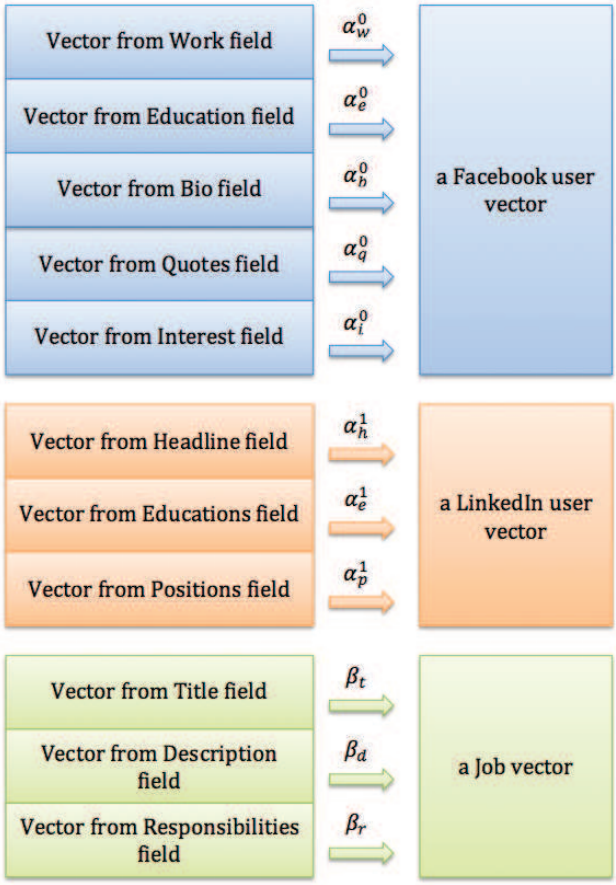



Figure 5.8 – Aggregation of vectors from different fields for Facebook users, LinkedIn users and Jobs.

### 5.4.1 Toward Engine-1: study of weighing functions, similarity heuristics and preprocessing techniques

In order to develop our first job recommender called Engine-1, we need to study and compare different preprocessing techniques (see section 4.2.1), weighting functions (see section 4.2.2) and similarity heuristics (see section 4.2.5) to find the right combination (which yields the highest AUC).

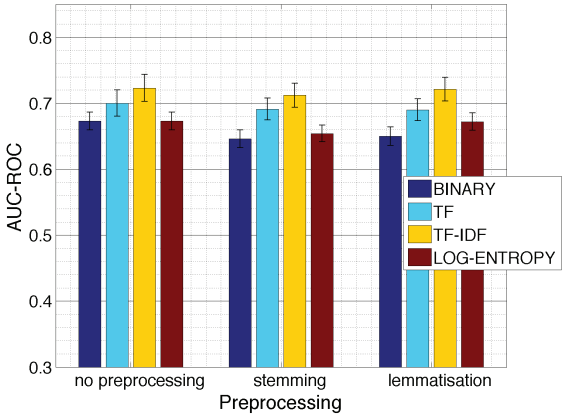
First of all, we do not know the importance of different fields of users and jobs in the task of job recommendation, so we set all the importance to 1 ($\alpha_w^0 = \alpha_e^0 = \alpha_b^0 = \alpha_q^0 = \alpha_i^0 = 1$ and $\alpha_h^1 = \alpha_e^1 = \alpha_p^1 = 1$ and $\beta_t = \beta_d = \beta_r = 1$, see Figure 5.8). Using theses settings, Figure 5.9 compares different weighting functions using various similarity heuristics and preprocessing techniques on our datasets. We note that the TF-IDF weighting function outperforms the other weighting functions. Using TF-IDF, we obtained similar performances for different preprocessing techniques, as a result, we choose to use lemmatization as preprocessing technique for all our job recommender systems based on bag-of-words model since it allows

a dimensionality reduction (see section 5.2). Figure 5.9 also reveals that cosine similarity generally yields better results than Pearson correlation coefficient.

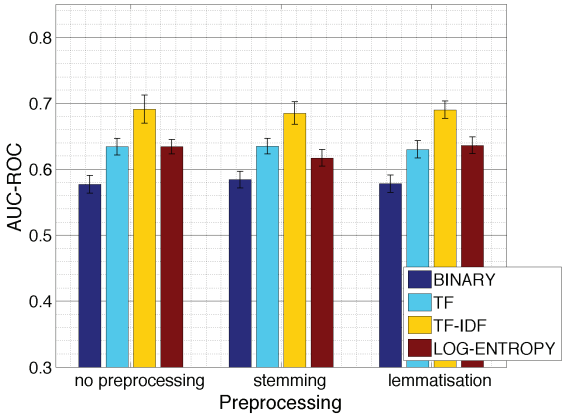
After studying different weighting functions, similarity measures and preprocessing techniques, let us define our first job recommender system also known as *Engine-1*. In this system, vectors of users and jobs are computed (following the method described above and in Section 5.4) by using TF-IDF as weighting function with lemmatization as preprocessing technique and 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 (4.15) of the user's vector and the vector of the job.

Figure 5.10 shows the results of Engine-1 on different datasets for all users, Facebook users and for LinkedIn users. We also notice that results are bad for all users in ALL dataset, these bad results are possibly due to the fact that user term space is different from job term space. 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 Figure 5.10): LinkedIn user term space seems closer to job term space than Facebook one.

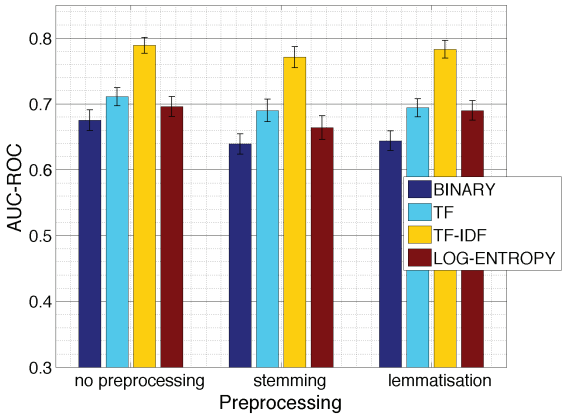
One can note that if the user term space is quite different from the job term space, Engine-1 will fail to make proper recommendations, this is the first weakness of this system. Another weakness of Engine-1 is that the assumption that *all the fields have the same importance on recommendation scores* is probably false. To address this weakness, we propose in the next section a method to estimate the importance of fields of users and jobs in the task of job recommendation.



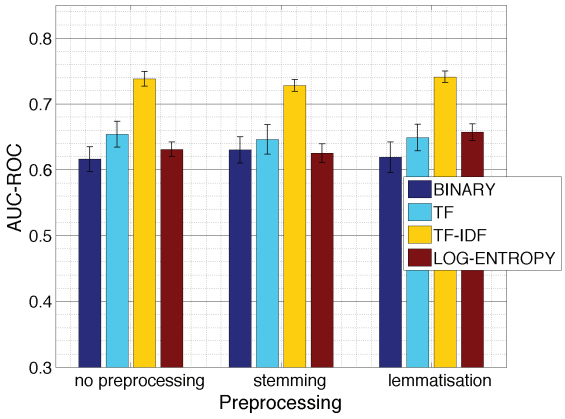

(a) Review dataset: Cosine similarity.



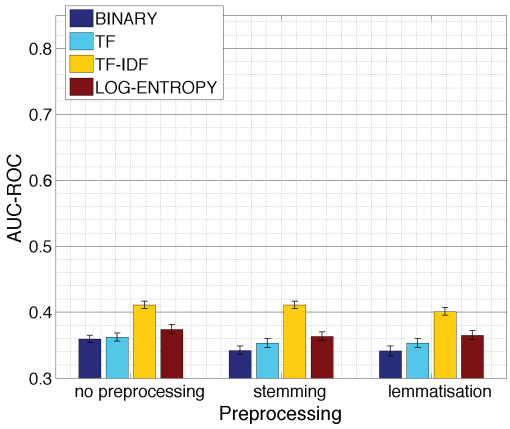

(b) Review dataset: Pearson correlation coefficient.



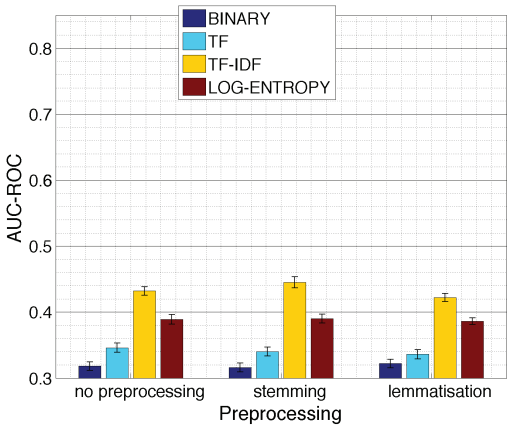

(c) Validation dataset: Cosine similarity.




(d) Validation dataset: Pearson correlation coefficient.




(e) All dataset: Cosine similarity.




(f) All dataset: Pearson correlation coefficient.

Figure 5.9 – Comparison between weighting functions, similarity functions and preprocessing techniques on our job recommendation datasets.

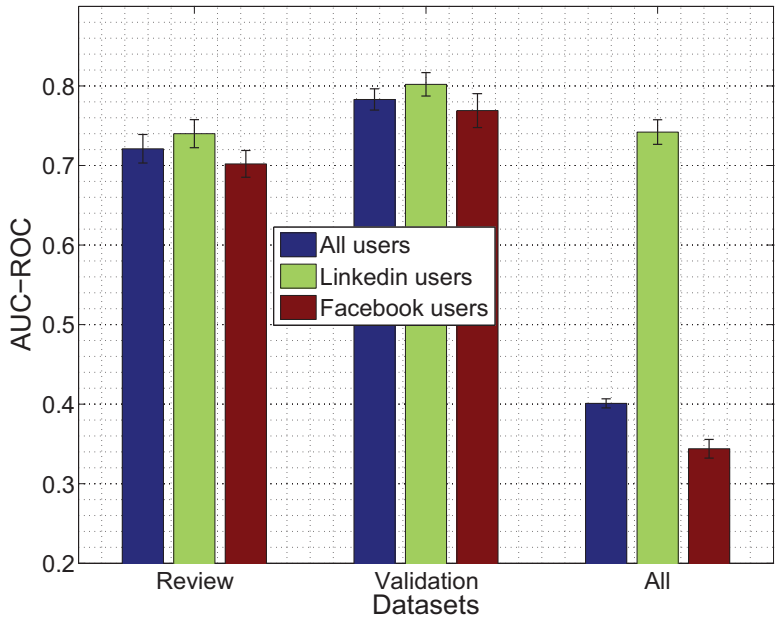



Figure 5.10 – Performance (AUC-ROC) of Engine-1 for Facebook users, LinkedIn users and all the users.

### 5.4.2 Engine-2: incorporating the importance of different fields of users and jobs into Engine-1

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

$$u(\alpha) = \sum_{f=1}^{f_u^0} \alpha_f^0 u_f^0 + \sum_{f=1}^{f_u^1} \alpha_f^1 u_f^1 \tag{5.1}$$

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

Similarly, we define the vector $v(\beta)$ of a job $v$ using the importance vector $\beta$ of job fields as the weighted sum of its fields' vectors, formally it is defined as follows:

$$v(\beta) = \sum_{f=1}^{f_j} \beta_f v_f \tag{5.2}$$

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

The dot product of $u_i$ and $v_j$ is defined by:

$$u(\alpha) \cdot v(\beta) = \sum_{k=1}^{n_k} u_k(\alpha) \cdot v_k(\beta) \tag{5.3}$$

where $n_k$ is the number of distinct terms/features.

The squared norm of $u(\alpha)$ and $v(\beta)$ are respectively defined by:

$$\|u(\alpha)\|^2 = \sum_{k=1}^{n_k} u_k(\alpha)^2 \tag{5.4}$$

$$\|v(\beta)\|^2 = \sum_{k=1}^{n_k} v_k(\beta)^2 \tag{5.5}$$

Using equations (5.3), (5.4), (5.5), cosine similarity between user $u$ and job $v$ is defined by:

$$\hat{y}_{uv}(\alpha, \beta) = \begin{cases} 0 & \text{if } \|u(\alpha)\| \|v(\beta)\| = 0 \\ \dfrac{u(\alpha) \cdot v(\beta)}{\|u(\alpha)\| \|v(\beta)\|} & \text{otherwise} \end{cases} \tag{5.6}$$

The predicted similarity $\hat{y}_{uv}(\alpha, \beta)$ between a user $u$ and a job $v$ using the importance of fields $\alpha$ and $\beta$ is computed as the $\cos(u(\alpha), v(\beta))$ (see eq. (4.15)).

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

$$E_\Gamma(\alpha, \beta, c_0, c_1) = \frac{1}{|\Gamma|} \sum_{(u,v,y) \in \Gamma} c_y \cdot (y - \hat{y}_{uv}(\alpha, \beta))^2 \tag{5.7}$$

Each entry of $\Gamma$ 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 $\hat{y}_{uv}(\lambda_1 \alpha, \lambda_2 \beta) = \hat{y}_{uv}(\alpha, \beta)$, which means that if the optimization problem has a solution, it is not unique. Therefore we solve a constrained optimization problem to reduce the number of solutions: the constraints are: $\|\alpha^0\| = 1$,

$\|\alpha^1\| = 1$ and $\|\beta\| = 1$. Formally, the optimization problem is defined as follows:

$$\begin{aligned} \alpha^\star, \beta^\star = \underset{\alpha \in \mathbb{R}^{f_u},\ \beta \in \mathbb{R}^{f_j}}{\arg\min} \quad & E_\Gamma(\alpha, \beta, c_0, c_1) \\ \text{subject to} \quad & \|\alpha^0\| - 1 = 0, \\ & \|\alpha^1\| - 1 = 0, \\ & \|\beta\| \ - 1 = 0. \end{aligned} \tag{5.8}$$

where $E_\Gamma$ is defined by the equation (5.7), $\alpha = (\alpha^0, \alpha^1)$, $f_u = f_u^0 + f_u^1$, $f_u^0$ and $f_u^1$ are respectively the numbers of Facebook and LinkedIn users fields, $f_j$ is the number of jobs fields, $c_0$ and $c_1$ are the costs of the class 0 and the class 1 respectively to be set.

To compute the importance of users and jobs fields on the recommendation scores, we solve the constrained optimization problem previously stated using the function *minimize* (from Scipy.optimize module) with the method SLSQP (Sequential Least Squares Programming) [Boggs and Tolle, 1995; Kraft, 1994]. 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.

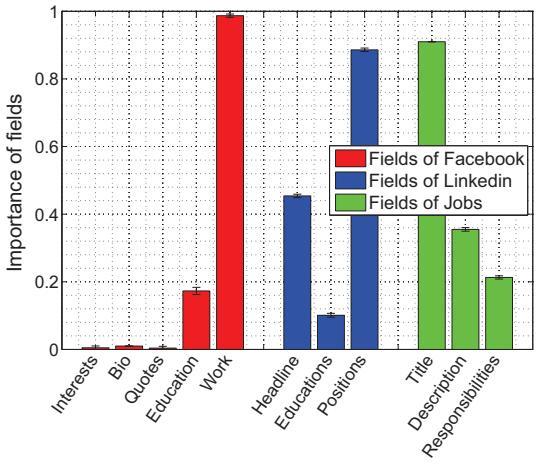
Figure 5.11 shows the importance of Facebook users', LinkedIn users' and jobs' fields with their confidence intervals respectively. It suggests that the important fields in the task of job recommendation are:

- *Work* field for Facebook users.
- *Headline/positions* fields for LinkedIn users.
- *Title* field for jobs.

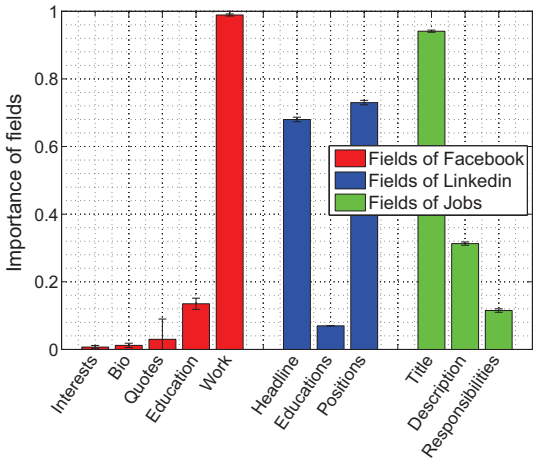
These results seems to make sense since the field work contains useful information to determine the interests of Facebook users for jobs. The field Headline sums up LinkedIn users' careers while Positions field contains their work history. The field title contains needed information about a given job to globally determine if it is relevant or not for a user.

Now let us address the weakness of Engine-1 (see section 5.4.1) related to the assumption that all the fields have the same importance on recommendation scores by using the optimal importance ($\alpha^{0*}$, $\alpha^{1*}$, $\beta^*$) of users' and jobs' fields (see Figure 5.11) when computing the vectors of documents as weighted sum of vectors of their fields (following the method described in Section 5.4): *this leads to the Engine-2*. The interest of a user for a given job is then measured by computing the cosine similarity (4.15) of the user's vector and the vector the job.

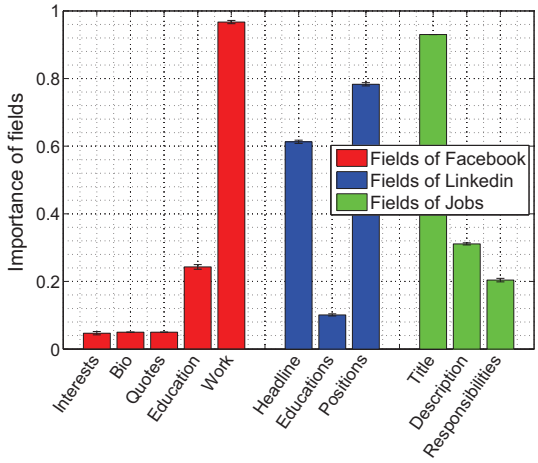
Figures 5.23a, 5.23b, 5.23c show that the application of the optimal weights of fields of users and jobs improve the quality of job recommendation. However, since Engine-1 and Engine-2 are using the same similarity function, Engine-2 suffers from the first mentioned weakness of




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.11 – 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.

Engine-1, namely the problem of mismatching user term space and job term space. To mitigate this problem, we investigate in the next section the use of machine learning techniques to learn models from our data that neatly fit our job recommendation tasks.

### 5.4.3 Engine-3: SVM-based recommender systems using TF-IDF vectors

In this section, we explore the use of trained statistical models to improve the quality of our job recommendation to social network users: there are many algorithms to learn statistical models from data but we choose SVMs (Support Vector Machines) [Cortes and Vapnik, 1995] since they are very popular and 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 $I_{\text{SVM}}$ of the SVM are stated as follows:

$$I_{\text{SVM}}(u,v) = (w(u_1,v_1),\ldots,w(u_T,v_T)) \tag{5.9}$$

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 $(\alpha^{0*}, \alpha^{1*}, \beta^{*})$ of different fields computed in the section 5.11 and $w(\bullet,\bullet)$ is a monotonic function.

We can use several $w$ functions like the product $(w(x,y) = x * y)$, the sum $(w(x,y) = x + y)$ but we want a bijective function (to be able to associate 2 unique points to each value of $w$), this naturally leads us to use the *Cantor pairing function* defined by:

$$cantor(x,y) = \frac{1}{2}(x+y)(x+y+1) + y \tag{5.10}$$

where $x$ and $y$ are integers.

One can note that the values returned by this function can be very high, to mitigate this problem, we use the logarithm of the values returned by Cantor pairing function. Finally, we define our $w$ as follows:

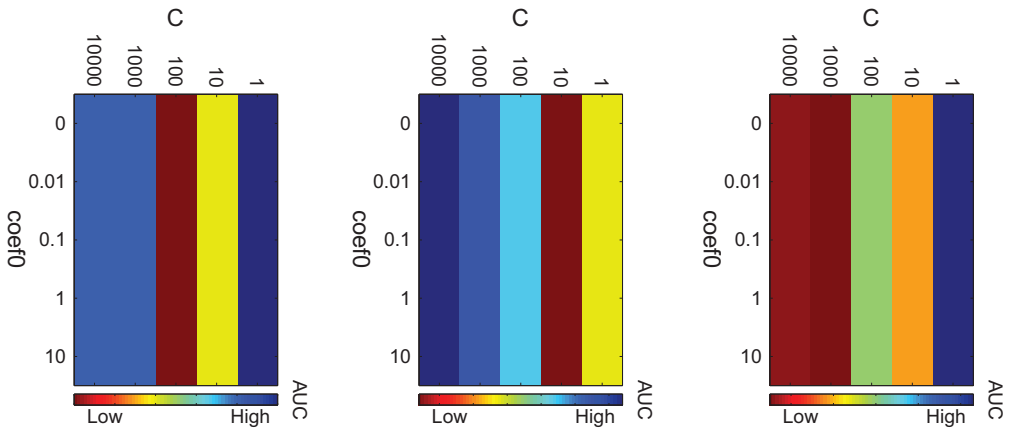
$$w(x,y) = \log(1 + cantor(\lfloor x \rceil, \lfloor y \rceil)) \tag{5.11}$$

where $\lfloor \bullet \rceil$ is the nearest integer to $\bullet$ and $cantor$ is the Cantor pairing function defined by the equation (5.10).

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 section 5.2), so we start our experiments by using a linear kernel (eq. (4.24a)): it is generally simpler, quicker and can mitigate the over-fitting problem in general. We then investigate the use of RBF (eq. (4.24c))

and Polynomial (eq. (4.24b)) kernels to improve the results obtained with the linear SVM.

In order to efficiently handle unbalanced datasets, we use different costs (as done in the section 5.4.2) for the two classes: $c_0$ and $c_1$ for the class 0 (label = 0) and the class 1 (label = 1) respectively. We use Grid search (see section 4.3.4) to find the optimal parameters for different kernels of SVM. Figures 5.12, 5.13, 5.14 depict the importance of parameters of kernels for Engine-3 we obtained. One can note that for our job recommendation tasks, the optimal values of hyper-parameters of SVM depend on the used dataset.

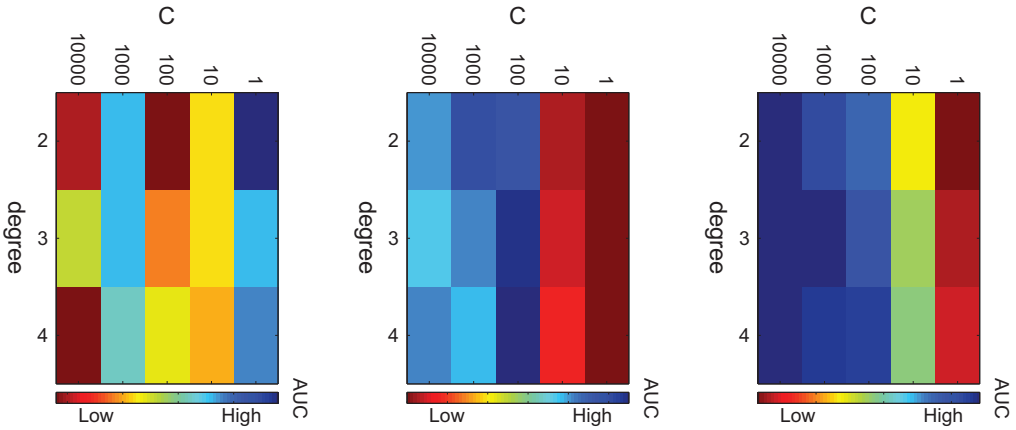



(a) ALL dataset (b) Validation dataset (c) Review dataset

Figure 5.12 – Optimization of hyper-parameters of Linear SVMs using TF-IDF vectors. Note Coef0 which corresponds to $r$ in eq (4.24a) and $C$ to the regularization parameter in eq (4.22). One can note the impact of Coef0 on the performance of our recommender systems is really very small compared to that of the regularization parameter $C$.

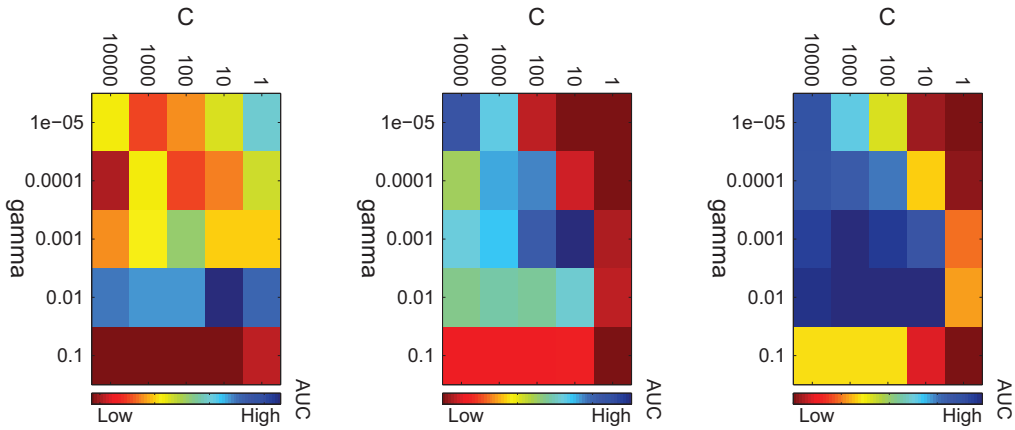
Comparing the results of Engine-3 for different kernels (see Figure 5.15), we note a clear outperformance of RBF kernel for the task of job recommendation to social network users.

Figure 5.23 reveals that the best SVM-based job recommender system (using RBF kernel) outperforms both Engine-1 and Engine-2: it mitigates the problem of difference between user term space and job term space. However Engine-3 suffers from the problem of missing data: it cannot make accurate job recommendation to users whose profiles are not well filled. To mitigate this problem, we investigate in the next section how to use social recommendation techniques (see section 3.6) to recommend jobs to Facebook and LinkedIn users.



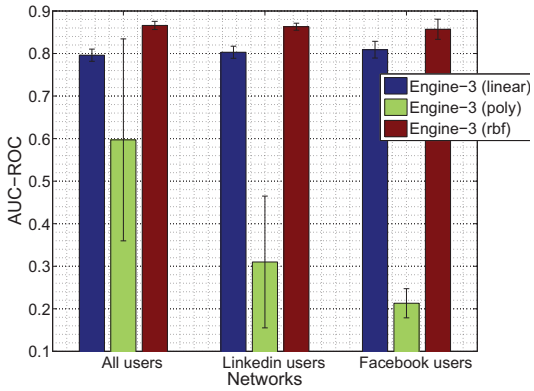

(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.13 – Optimization of hyper-parameters of Poly SVMs using TF-IDF vectors. Note Coef0 which corresponds to $r$ in eq (4.24b) is set to 1, degree corresponds to $d$ in eq (4.24b) and $C$ to the regularization parameter in eq (4.22).

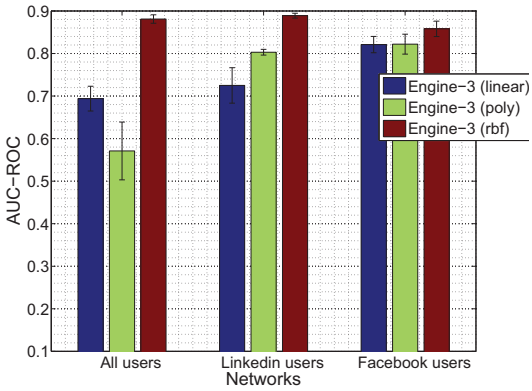



(a) Review dataset (b) Validation dataset (c) ALL dataset

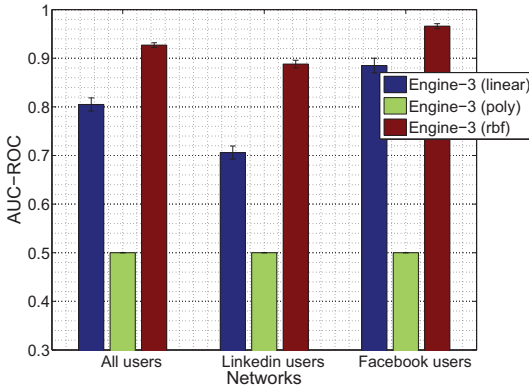
Figure 5.14 – Optimization of hyper-parameters of RBF SVMs using TF-IDF vectors. Note gamma corresponds to $\gamma$ in eq (4.24c) and $C$ to the regularization parameter in eq (4.22).




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.15 – Engine-3: comparison between different kernels of SVMs: one can note a clear out-performance of the RBF kernel.

### 5.4.4 Engine-4: social recommender systems

Since our Facebook users do not completely fill the fields that are interesting in job recommendation tasks (see Table 5.2), we experiment the use of data of 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 [Aranda et al., 2007; Kantor, 2009; Ma et al., 2011].

Firstly, we propose a generic pseudo algorithm for social recommendation (see the pseudo algorithm 3).

**Pseudo algorithm 3:** 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

```
foreach User ∈ uData do
    Find the clusters of his friends densely connected using community detection and/or clustering
    methods on fData;
    foreach Cluster of friends do
        Extract statistics about colleges/universities and companies using the sub-fields of Facebook
        and LinkedIn users' profiles;
    end
    Using these statistics, select the relevant clusters (communities of (ex-)colleagues and
    (ex-)classmates) using the thresholds θ_ex-colleagues and θ_classmates;
    foreach Job ∈ jData do
        Compute the interest of User for Job using both his data (with the weight α) and the data from
        relevant communities of friends (with the weight (1 − α));
    end
end
```

Unfortunately, due to privacy concerns, the data about users' social relationships are often not accessible to third parties, making difficult the application of the pseudo algorithm 3: 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 3 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 $v$. This leads to 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 equation (5.12), in Engine-4b following the equation (5.13) and in Engine-4c following the

equation (5.14).

$$\text{social score}(u, v) = \cos(\alpha\, u + (1-\alpha)\overline{u_F}, v) \tag{5.12}$$

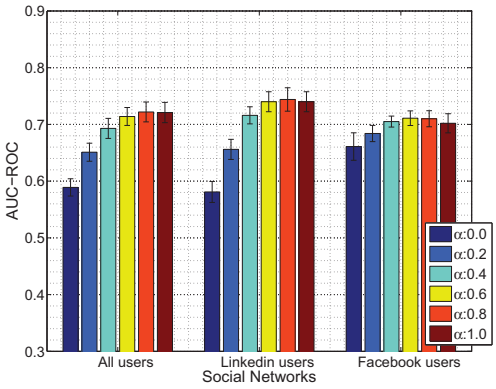
$$\text{social score}(u, v) = \alpha\cos(u, v) + (1-\alpha)\cos(\overline{u_F}, v) \tag{5.13}$$

$$\text{social score}(u, v) = \alpha\cos(u, v) + \frac{(1-\alpha)}{|F|}\sum_{f\in F}\cos(u^f, v) \tag{5.14}$$

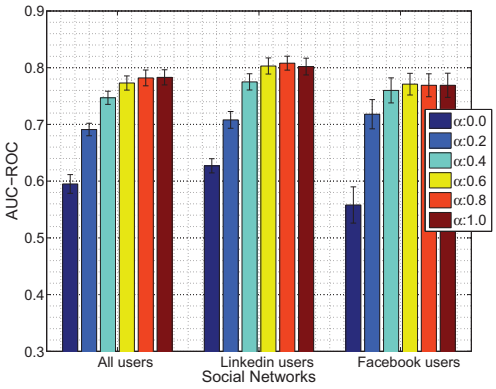
where $u$ is the original vector of the user, $\overline{u_F}$ is the average vector of the user's friends, $v$ is a vector of a job, $F$ is the set of friends of the active user, $u^f$ is the vector of the friend $f$ and $\alpha \in [0,1]$ is the importance of user's data.

In Engine-4a, Engine-4b and Engine-4c, the TF-IDF vectors of users and jobs are obtained by assuming that all the fields have same importance (as in Engine-1, see section 5.4.1).

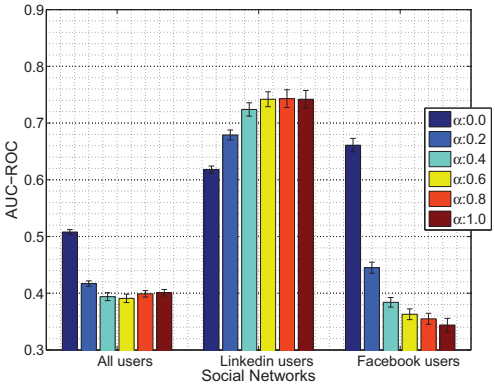
We compare the 3 basic social recommender systems to Engine-1, Figures 5.16, 5.17, 5.18 show that all the proposed social recommender systems fail to give better results than Engine-1 for any value of $\alpha$ for LinkedIn users. For Facebook users we do not find any improvement for Review and Validation datasets but for ALL dataset, we find some values of $\alpha$ that improve the quality of job recommendation: on Facebook, the use of the data of users' friends can improve the quality of job recommendation in some special cases.




(a) Review dataset.

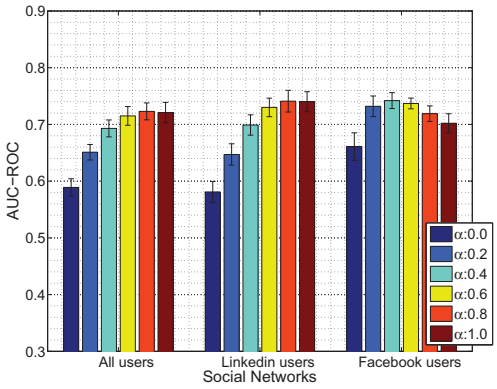



(b) Validation dataset.

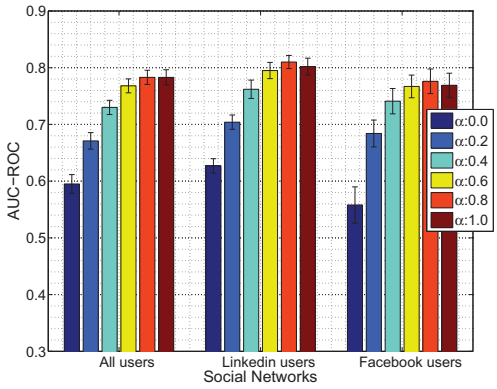



(c) ALL dataset.

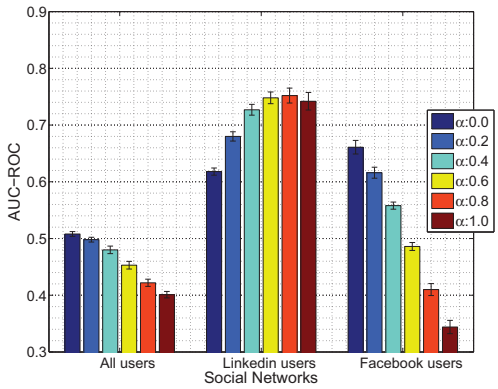
Figure 5.16 – Performance of Engine-4a on our different datasets. For a reminder, $\alpha$ represents the importance of the active user's data (and $1-\alpha$ that of his friends' data); note that $\alpha = 1$ corresponds to Engine-1.




(a) Review dataset.

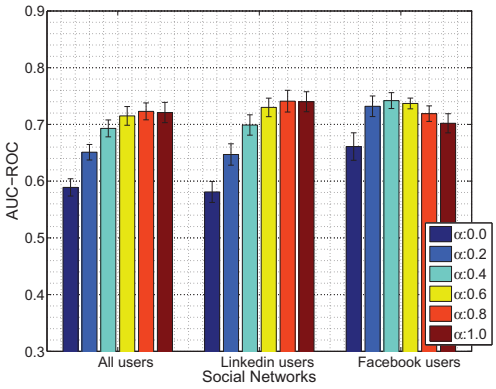



(b) Validation dataset.

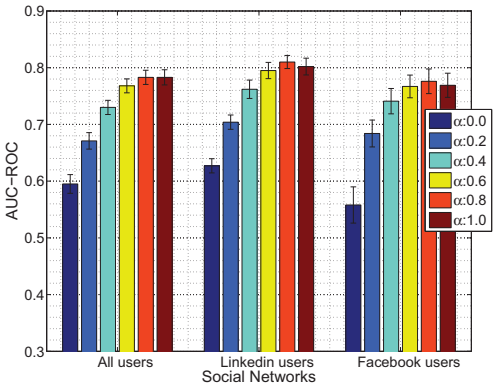



(c) ALL dataset.

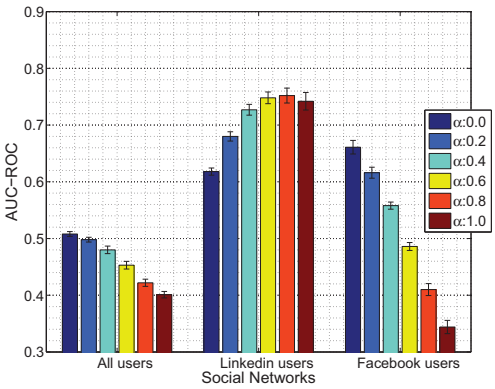
Figure 5.17 – Performance of Engine-4b on our different datasets. For a reminder, $\alpha$ represents the importance of the active user's data (and $1-\alpha$ that of his friends' data); note that $\alpha = 1$ corresponds to Engine-1.




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.18 – Performance of Engine-4c on our different datasets. For a reminder, $\alpha$ represents the importance of the active user's data (and $1-\alpha$ that of his friends' data); note that $\alpha = 1$ corresponds to Engine-1.

### 5.4.5 Engine-5: relevance feedback applied to Engine-1

In the previous section, our attempt to use social recommendation techniques to mitigate the problem of missing data when recommending jobs to social network users has led to disappointing results. Based on this observation, we explore another technique from the literature of information retrieval to enrich the profiles of users with those of jobs they liked or disliked in the past. This technique is known as **relevance feedback** [Rocchio, 1971] in the literature.

Since the profiles of our users and jobs only contain terms with positive weights, we use a slightly modified version of the famous Rocchio's formula [Rocchio, 1971] to make sure to obtain only positive weights associated to terms in the profiles of users enriched with those of the jobs linked to them.

Mathematically, the modified version of Rocchio's formula used in this study is defined as follows:

$$u^{'} = \max(0, (a.u + b\overline{u_J^+} - c\overline{u_J^-})) \tag{5.15}$$

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.

Note that in the equation (5.15), the parameters $a$, $b$, $c$ respectively represent the importance of users' original profiles, that of jobs matching users' profiles and that of jobs mismatching users' profiles.

Enriching users' profiles with those of jobs related to them leads to our *Engine-5* in which the original TF-IDF vectors of users and jobs are obtained by assuming that all of fields have the same importance (as in Engine-1, see section 5.4.1).

To enrich the profiles of users, we split our datasets into training and test sets using 10-fold cross-validation as described in the section 5.3. Each training set is used as a feedback set: the profiles of users in the feedback set are enriched with those of jobs linked to them by setting $a = b = c = 1$ (assuming that original profiles of users are as important as those of the profiles of jobs that users liked or disliked in the past) in the equation (5.15).

Our experiments reveals that the use relevance feedback drastically improves the quality of our job recommendation (see Figure 5.19). Here the confidence intervals are larger than those of Engine-3 with RBF kernel (see Figure 5.15): this is probably due to the fact that some users are only linked to few number of jobs (not enough to sufficiently improve their profiles for all folds of cross-validation).

The results obtained with Engine-5 show that enriching the profiles of users with those of jobs they liked or disliked in the past can allows to mitigate the problem of missing data in

users' profiles and then leading to an improvement in the quality of job recommendation. However, we noticed that social network users did not generally give feedback about job recommendation that our recommender systems made to them; this makes the application of Engine-5 a bit difficult (in our job recommendation tasks) and leads us to investigate additional techniques to deal with missing information.

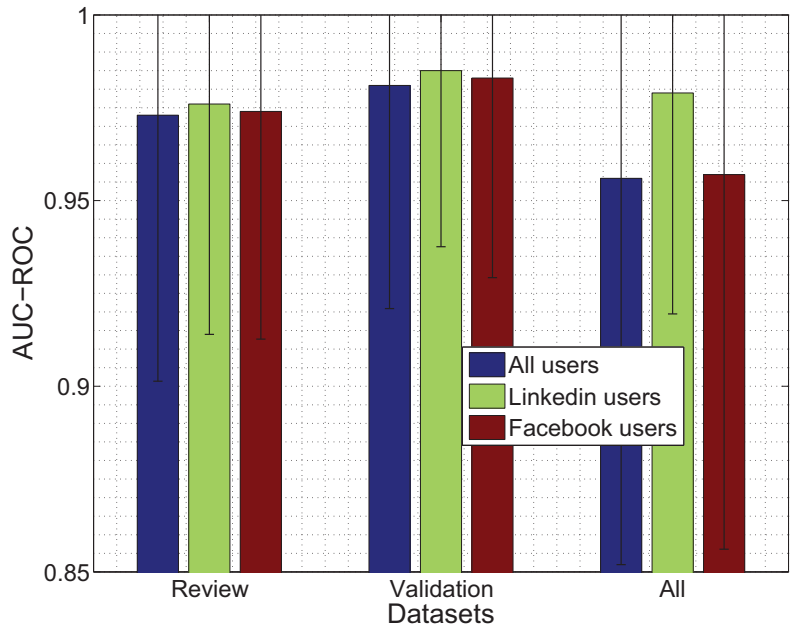


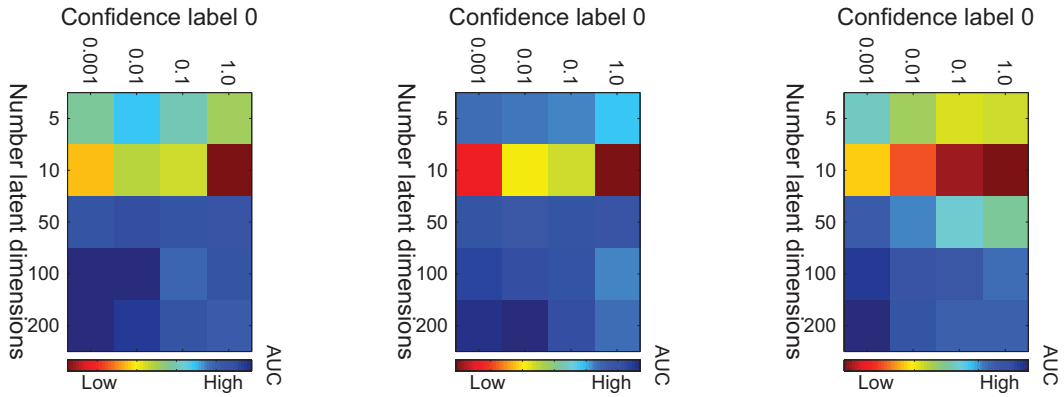

Figure 5.19 – Performance (AUC-ROC) of Engine-5 for Facebook users, LinkedIn users and all the users.

### 5.4.6 Comparison between the proposed recommender systems (based on bag-of-words models) and two state-of-the-art systems

In this section, we compare Engine-1, Engine-2, Engine-3 and Engine-5 to two methods of the literature, the first method is a simple Collaborative Filtering based on matrix factorization (MF, see section 3.2.2) and the second method is the Collaborative Topic Regression (CTR) proposed by [Wang and Blei, 2011] which is a hybrid recommender system combining content-based (see section 3.2) and collaborative recommendations (see section 3.2). We use the code provided by [Wang and Blei, 2011] to compare our methods to MF and CTR.

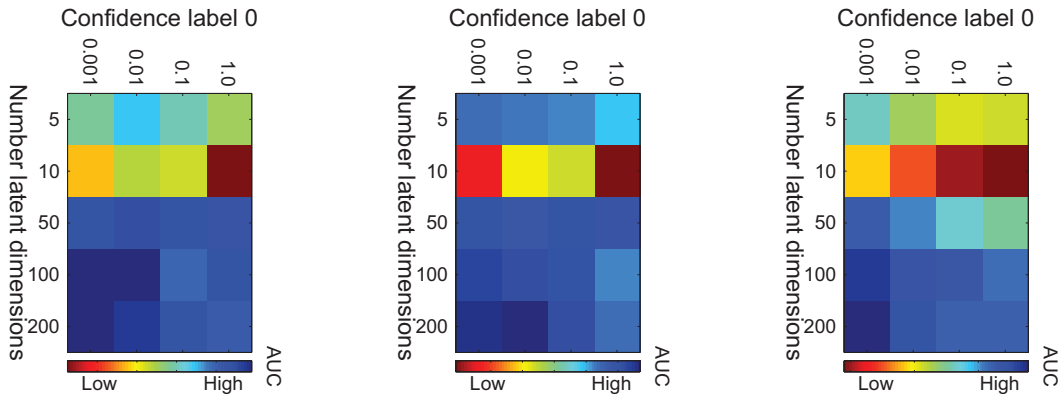
Figures 5.20 and 5.21 show the results of the optimization of MF and CTR hyper-parameters. A comparison between MF and CTR reveals that CTR outperforms MF (see Figure 5.22), which was expected and confirms the results obtained in [Wang and Blei, 2011].

The comparison between Engine-1 to Engine-2 shows that the application of the importance of fields (see Figure 5.11) significantly improves the quality of job recommendation on all



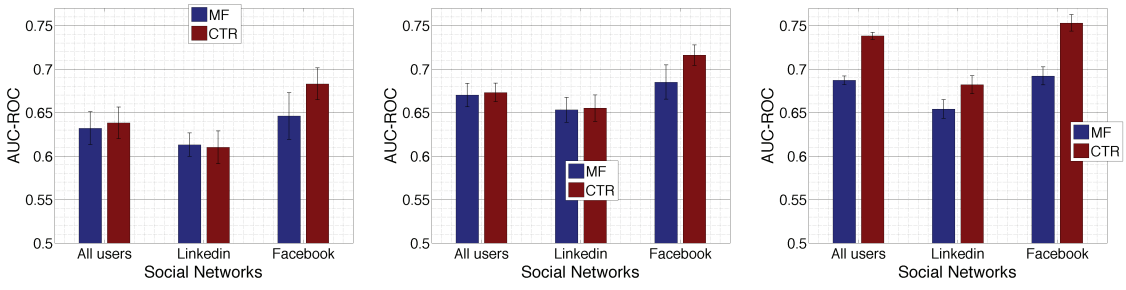

(a) Review dataset. (b) Validation dataset. (c) ALL dataset

Figure 5.20 – Optimization of Matrix Factorization (MF) hyper-parameters.




(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.21 – Optimization of Collaborative Topic Regression (CTR) hyper-parameters.




(a) Review dataset. (b) Validation dataset. (c) ALL dataset.

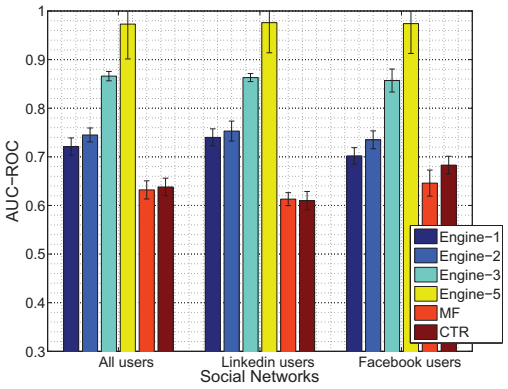
Figure 5.22 – Comparison between a Simple Matrix Factorization (MF) and the Collaborative Topic Regression (CTR).

our datasets (see Figure 5.23): we can conclude that it is important to take into account the importance of fields of users and jobs when recommending jobs to social network users. Figure 5.23 reveals that Engine-3 and Engine-5 outperform CTR which yields better results than MF. However one can note that MF outperforms Engine-2 on ALL dataset. MF and CTR suffer from cold start recommendation problem since in our datasets users have not enough related jobs. The comparison between Engine-3 and Engine-5 (based on relevance feedback) reveals that Engine-5 slightly outperforms Engine-3.

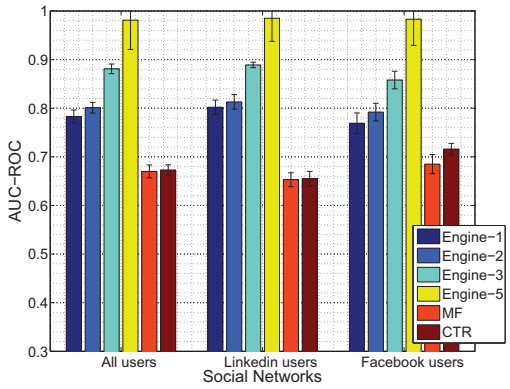
Globally, our experiments about using the bag-of-words model to recommend jobs to social network users reveal that:

- Using heuristic similarity functions leads to systems whose performance can be poor since the vocabulary used by social network users can be different from that of job descriptions and the profiles of users on some social networks like Facebook are incomplete (missing data) most of the time.
- Using machine learning or relevance feedback allows to mitigate the problems related to the mismatching of vocabulary between users and jobs and that of incomplete profiles.

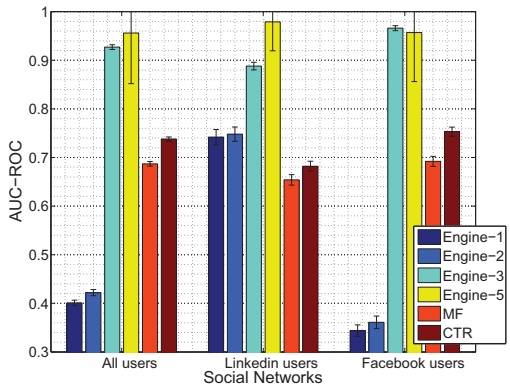
Using machine learning or relevance feedback is interesting but requires to collect feedback data about jobs matching/mismatching social network users' profiles, these data are difficult to collect sometimes since social network users did not generally give feedback about jobs matching/mismatching their profiles. To address this problem, we investigate the use of knowledge databases to deal with missing data in the next section.




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.23 – Comparison between Engine-1, Engine-2, Engine-3, Engine-5, MF and CTR. Note that Engine-3 is using the RBF kernel since this kernel outperforms the others (see Figure 5.15).

## 5.5 Taxonomy-based job recommendation

In this section, we explore how to use knowledge databases (taxonomies) to deal with missing data and to mitigate the limitations of the bag-of-words models when recommending jobs to social network users.

Recall that our Facebook users have authorized the Work4 applications to access data in 5 fields: Work, Education, Quote, Bio, and Interests. LinkedIn users only have authorized 3 fields: Headline, Educations, Positions. LinkedIn Educations and Positions fields are almost equivalent to Facebook Education and Work fields respectively. The description of Work4 jobs has 3 fields: Title, Description, Responsibilities.

In our previous developed recommender systems (see section 5.4), the vectors of users and jobs are extracted using the bag-of-words models, the main limitation of which is the fact that they assume that the relative order of terms is not important. The order of terms could be important in a task of job recommendation: as example, let us consider 2 sentences "a senior nurse" and "a nurse for senior", in bag-of-word model, the two sentences have the same vector but they represent two different types of jobs. Based on this observation, we decide to investigate the use of a taxonomy-based vector model. Our previous studies showed that the most important fields in the task of job recommendation (see Figure 5.11) are: *Work* for Facebook users, *Headline and positions* for LinkedIn users and *Title* for jobs, as a result, we use the information contained in *Work*, *Headline/positions* and *Title* fields to extract a new type of vector for social network users and jobs that we called O*NET vector using the O*NET-SOC taxonomy[2] [National Center for O*NET Development, 2013; Peterson et al., 2001] (a taxonomy that defines the set of occupations across the world of work) as described in the pseudo algorithm 4. O*NET database (see the description in the appendix B) only supports English, reason why we test our proposed methods on only users and jobs whose language is English.

We indexed O*NET databases using Elasticsearch[3] and we thus use this library to query the different O*NET occupations (with their relative relevance scores) related to a document (Work field data for Facebook users, Headline/Positions' data for LinkedIn users and Title data for jobs). Elasticsearch uses a kind of TF-IDF matching between O*NET occupations' data (in O*NET-SOC taxonomy databases) and documents (users or jobs) to compute the relevance scores but we did not try to optimize the used formula. Each document is then represented by its distribution scores over all the occupations in O*NET databases. O*NET taxonomy currently contains 1,040 distinct occupations, as a result, our O*NET vectors are encoded by *1,040 dimensions*. Figure 5.24 shows the scheme of our taxonomy-based recommender systems.

The following example in which we consider a user who only filled his last position sub-field with "software engineer", shows the difference between our proposed O*NET vector

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

**Pseudo algorithm 4:** Extraction of a document O*NET vector

**Input**:
doc: a document (a Facebook user or LinkedIn user a job)
**Output**:
d_vector: the document O*NET vector (a list of couples (job family, relevance score))

```
if doc.type = 'Job' then
    Extracting text from the Title field;
    text ← extract_from_field(doc, sub-field='Title');
else if doc.type = 'LinkedIn' then
    Extracting text from the last position and Headline fields;
    h_text ← extract_from_field(doc, sub-field='Headline');
    lp_text ← extract_from_field(doc, sub-field='Last position');
    text ← concatenate (h_text, lp_text)
else if doc.type = 'Facebook' then
    Extracting text from the Last position field;
    text ← extract_from_field(doc, sub-field='Last position');
else
    Unknown type of document;
    text ← ""
end
Cleaning the text;
text ← remove_stopwords(text);
if (doc.main_language ≠ 'English') then
    Translating the text into english;
    text ← translate(text, to='English');
end
Quering O*NET database:  the result (onet_vector) is a list of couples (job
family, relevance score);
onet_vector ← query_O*NET_dbs(text);
m_relevance ← max_relevance(onet_vector);
d_vector = [];
foreach (family, relevance) in onet_vector do
    d_vector ← append(d_vector, (family, Relevance / m_relevance));
end
Return d_vector
```

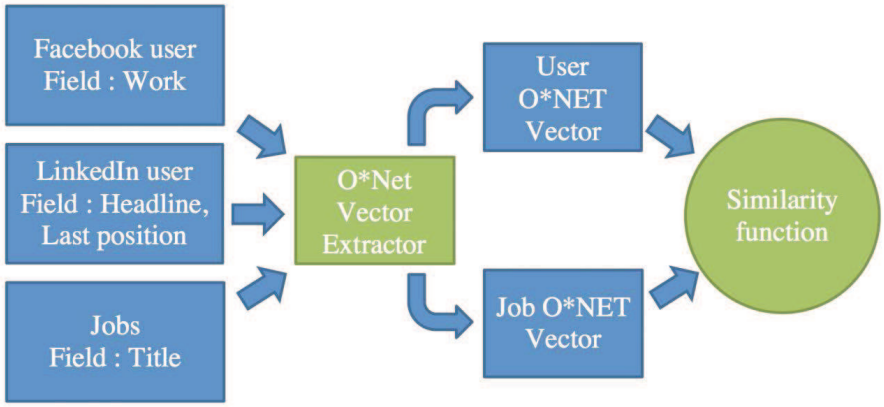



Figure 5.24 – Scheme of our taxonomy-based job recommender systems.

model and the traditional TF-IDF vector model [Salton et al., 1975]: the user's TF-IDF vector could be *[("software", 5.8), ("engineer", 3.2)]* while his O*NET vector will look like *[("Software Developers", 1), ("Aerospace Engineers", 0.97), ("Electrical Engineers", 0.97), ..., ("Software Quality Assurance Engineers and Testers", 0.85), ..., ("Web Developers", 0.52), ..., ("Database Architects", 0.24), ..., ("Avionics Technicians", 0.01)]*.

### 5.5.1 Engine-6: cosine-based recommender systems using O*NET vectors

After extracting the O*NET vectors of our social network users and jobs, we define a set of heuristic-based job recommender systems (Engine-6) that use the proposed taxonomy-based vector model.

*Engine-6a* is the first proposed job recommender system that uses our O*NET vector model (see section 5.4) together with cosine similarity (see equation (4.15)). The second proposed job recommender system called *Engine-6b* uses the proposed O*NET vector model together with Pearson Correlation Coefficient (PCC) (see equation (4.16)).

The third and fourth job recommender systems are based on fuzzy logic which has been introduced by [Zadeh, 1965] in 1965. We defined two similarity functions adapted to our vector models for job recommendation to social network users, these similarity functions are based on some AND and OR fuzzy logic's operators:

$$\text{fuzzy-sim}(u,v) = \text{fuzzy-OR}_{k=1}^{K}(\text{Fuzzy-AND}(u_k, v_k)) \tag{5.16}$$

where $K$ is the total number of O*NET occupations, $u$ and $v$ are the vectors of a user and a job respectively. There are several AND and OR fuzzy logic's operators [Castro, 1995], we use: *max* as OR operator, *min* and *product* as AND operators [Castro, 1995], this leads to two similarity measures:

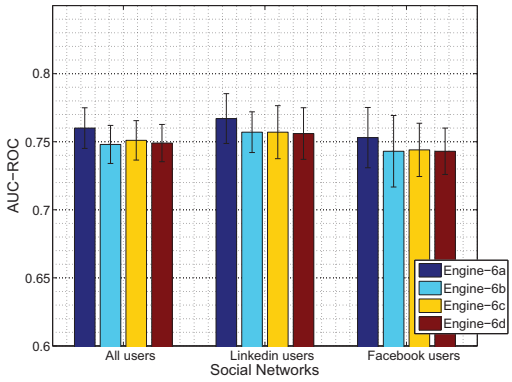
$$\text{fuzzy-sim-1}(u,v) = \max_{k=1}^{K}(u_k.v_k) \tag{5.17}$$

$$\text{fuzzy-sim-2}(u,v) = \max_{k=1}^{K}\min(u_k, v_k) \tag{5.18}$$

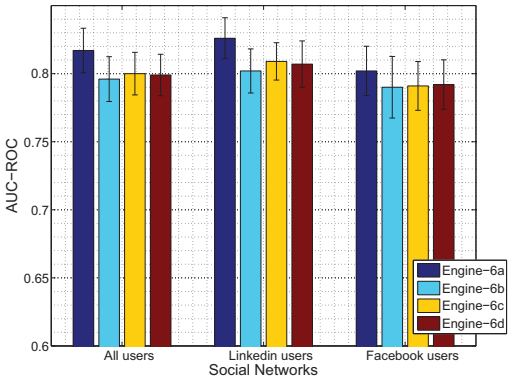
*Engine-6c* and *Engine-6d* respectively use fuzzy-sim-1 and fuzzy-sim-2 as similarity functions together with the proposed O*NET vector model.

In the first series of experiments, we compare Engine-6a, Engine-6b, Engine-6c and Engine-6d, the results (see Figure 5.25) show that Engine-6a (which is based on cosine similarity) outperforms the others on our 3 datasets. The comparison between Engine-6a and Engine-

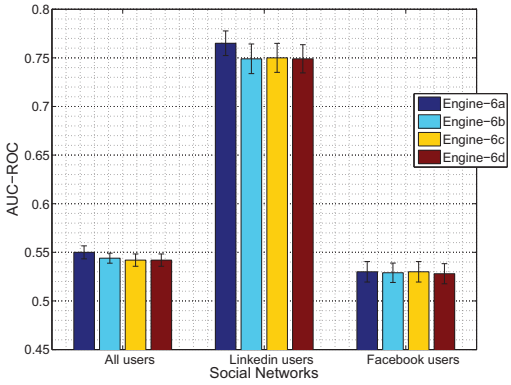
2 reveals that using O*NET vectors allows to drastically decrease the difference of quality between job recommendation made to Facabook users and those made to LinkedIn users (see Figure 5.35c). However, the results of Engine-6a on ALL dataset is low and can be probably improved using models learnt from our data, we therefore investigate how the use of SVM models can improve the quality of recommendations in the next section.




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.25 – Comparison between Engine-6a, Engine-6b, Engine-6c and Engine-6d. For a reminder, Engine-6a, 6b, 6c and 6d are respectively using cosine similarity, Pearson correaltion coefficient, fuzzy-sim-1 (see eq. (5.17)) and fuzzy-sim-2 (see eq. (5.18)).

### 5.5.2 Engine-7: SVM-based recommender systems using O*NET vectors

After showing in the previous section that the use of our proposed O*NET vector together with similarity functions can improve the quality of job recommendation made to social network users, we investigate in this section the use of trained statistical models based on SVMs (see section 4.3.1). Using SVM models together with the proposed O*NET vector model leads to our seventh job recommender system called *Engine-7*.

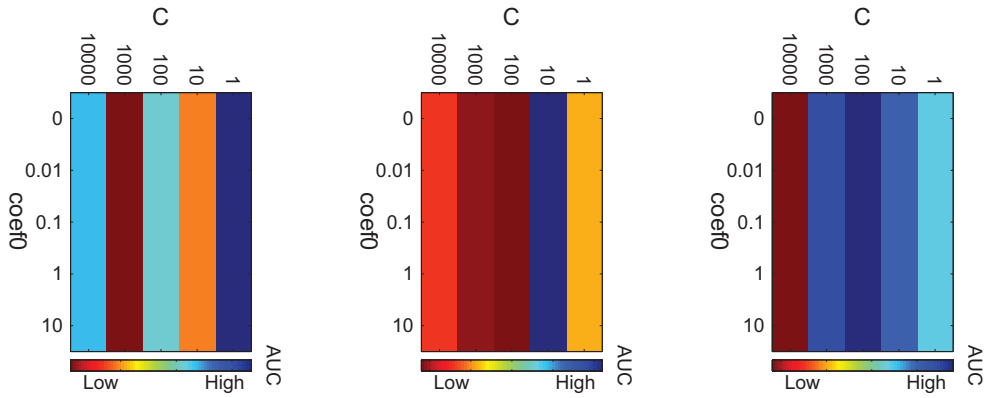
The input vectors $I_{SVM}$ of the SVM are stated (similarly to eq. (5.9)) as follows:

$$I_{SVM}(u,v) = (w(u_1,v_1),\ldots,w(u_K,v_K)) \tag{5.19}$$

where $K$ is the total number of O*NET occupations, $u$ and $v$ are respectively the O*NET vectors of a user and job and $w(\bullet,\bullet)$ is a monotonic function defined by eq. (5.11)).

We test three kernels for our SVM models: Linear (eq. (4.24a)), Polynomial (eq. (4.24b)) and RBF (eq. (4.24c)). In order to efficiently handle unbalanced datasets, we use different costs (as done in the section 5.4.2) for the two classes: $c_0$ and $c_1$ for the class 0 (label = 0) and the class 1 (label = 1) respectively.

We optimize the hyper-parameters of different kernels of SVMs as shown by Figures 5.26, 5.27 and 5.28 using Grid search technique (see section 4.3.4).

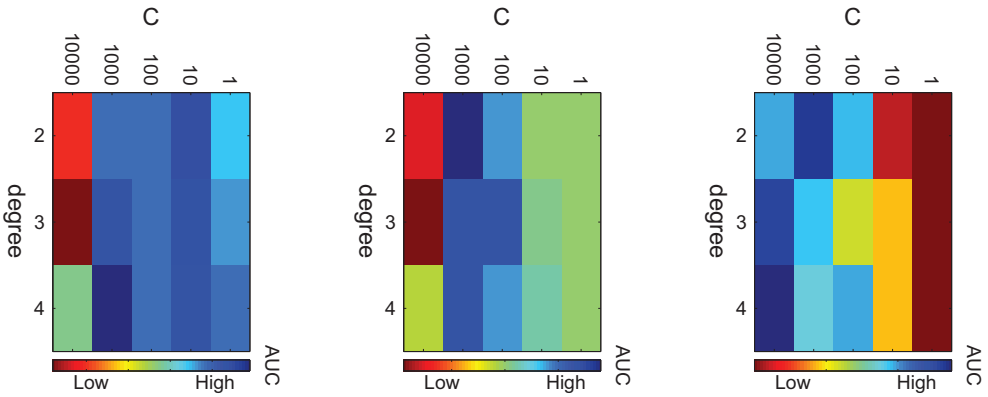



(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.26 – Optimization of hyper-parameters of Linear SVMs using O*NET vectors. Note Coef0 which corresponds to $r$ in eq (4.24a) and $C$ to the regularization parameter in eq (4.22). One can note the impact of Coef0 on the performance of our recommender systems is really very small compared to that of the regularization parameter $C$.

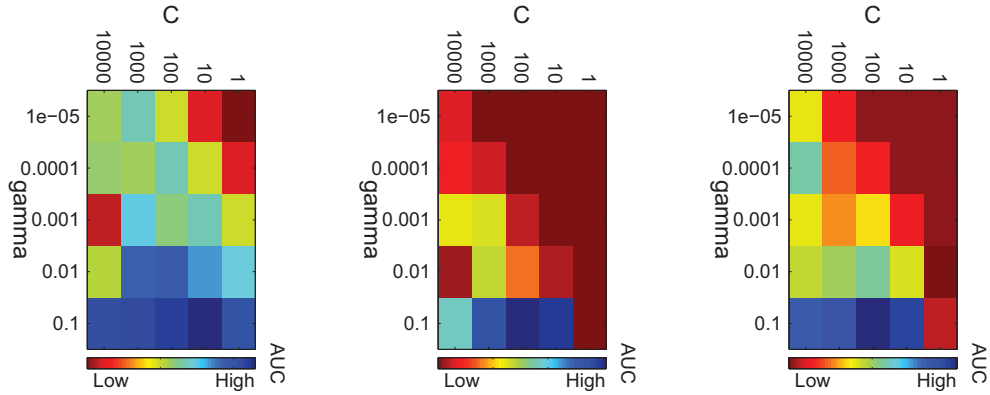
The comparison of the results of Engine-7 for different kernels (see Figure 5.29) reveals a clear out-performance of RBF kernel for the task of job recommendation to social network users, this result is similar to the results of Engine-3 (see section 5.4.3). The results of Polynomial kernel are disappointing compared to those of the linear kernel.

The comparison between Engine-6 and Engine-7 (see Figure 5.35) shows that using SVMs increases the quality of job recommendation compared to the use of similarity functions. However, Figure 5.36 reveals that Engine-3 yields better results than Engine-7: this is probably due to the fact that reducing the dimensionality of our problem using O*NET leads to a loss of information for SVM, and therefore decreases its performance.



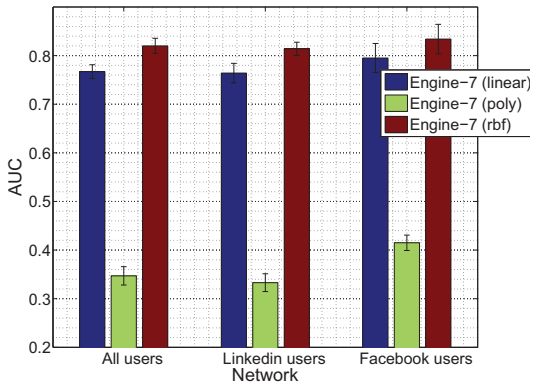

(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.27 – Optimization of hyper-parameters of Poly SVMs using O*NET vectors. Note Coef0 which corresponds to $r$ in eq (4.24b) is set to 1, degree corresponds to $d$ in eq (4.24b) and $C$ to the regularization parameter in eq (4.22).

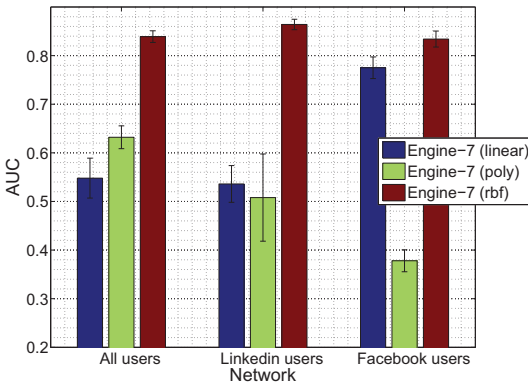



(a) Review dataset (b) Validation dataset (c) ALL dataset

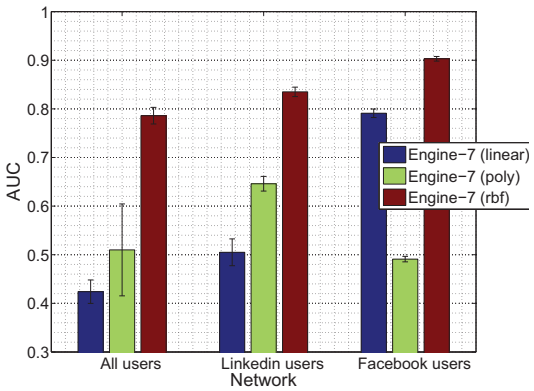
Figure 5.28 – Optimization of hyper-parameters of RBF SVMs using O*NET vectors. Note gamma corresponds to $\gamma$ in eq (4.24c) and $C$ to the regularization parameter in eq (4.22).




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.29 – Engine-7: comparison between different kernels of SVMs: one can note a clear out-performance of the RBF kernel.

As stated in the section 1.1, recommending relevant jobs to users is a multidimensional problem (matching skills, educations, abilities, countries, languages, etc. of users with those required by jobs): we study in the next section this problem to complete our study about job recommendation to social network users.

### 5.5.3 Engine-8: SVM-based recommender systems using multilayer vectors

Matching a user to a job is a multidimensional problem by nature (as presented in the section 1.1): users have to have for instance, the skills, experiences and abilities required by the corresponding positions; the language(s) and countries of the users should also match those of jobs. Based on this observation, we develop in this section a multilayer vector model for social network users and job descriptions suited for the presented multidimensional user-job matching problem. We call *multilayer vector* in our job recommendation context, a vector containing several layers where each layer contains information about a specific aspect of matching jobs

to users. Example: using O*NET databases, we can extract the following multilayer vector for a user:

- Layer 1 (occupations): { R&D Engineer, Data scientist, Research scientist}.
- Layer 2 (educations): { PhD in statistics, Master of science}.
- Layer 3 (abilities): { Great oral expression }.

The description of jobs generally contains information about the required abilities, skills, level of education and experience *but these information are difficult to automatically extract*, as a result, we decide to use O*NET-SOC taxonomy [National Center for O*NET Development, 2013; Peterson et al., 2001] (see section 4.2.3 and appendix B) to extract information like abilities, skills, etc. about users and jobs instead of a direct extraction from the description of jobs/profiles of users. Our multilayer vector model therefore combines the classic TF-IDF vector model with a vector model based on the O*NET-SOC taxonomy.

As presented previously, O*NET-SOC taxonomy (see the description in the appendix B) contains several models about worker requirements, experience and occupational requirements, worker characteristics, occupation-specific requirements and occupation characteristics. After analyzing the O*NET taxonomy models, we decide to use the following databases/knowledge bases: *Skills, Abilities, Interests, Job-zones, Knowledge, Work values, Work activities, Work context, Work styles, Task categories and Education-training-and-experience*.

The language and country vectors of a document are respectively defined as the list of the languages and countries related to this document (a social network user or a job description). After defining the language and country vectors of a document, we extract its TF-IDF vector (as for Engine-1, see section 5.4.1) by concatenating the data in its fields, removing stop words, using lemmatisation to reduce each term into its lemma (see section 4.2.1) and using the TF-IDF weighting function defined by eq. (4.6). The O*NET vectors of documents are extracted following the pseudo algorithm described in the section 5.5.

After extracting the TF-IDF vector and O*NET vector of a document, we can extract its other O*NET related vectors (Skills, Abilities, Interests, Job-zones, Knowledge, Work values, Work activities, Work context, Work styles, Task categories and Education-training-experience) using the pseudo algorithm 5. Figure 5.30 depicts the hierarchical model used to extract our proposed multilayer vector for users and jobs.

To compute the similarity between the language vectors or country vectors, we use an intersection-based heuristic function defined as follows:

$$\text{intersection\_sim}(u,v) = \begin{cases} 0.5 & \text{if u or v is empty} \\ 0 & \text{if } u \cap v \text{ is empty} \\ 1 & \text{otherwise} \end{cases} \tag{5.20}$$

**Pseudo algorithm 5:** Generic pseudo algorithm to extract the vectors related to an O*NET vector like abilities and skills.

**Data**: *oVector*: O*NET vector;
*tOutput*: type of the output vector;
`/* oVector is a list of couples (O*NET occupation, relevance score) */`
`/* tOutput ∈ {skills, abilities, interests, job zones, Knowledge, Work values, Work activities, Work context, Work styles, Task categories, Education-Training-Experience} */`
**Result**: *tOutput vector* related to the oVector

*list_features* ← all features (in the O*NET tOutput database) related to O*NET occupations *in* oVector;
`/* output_vector is a dictionary */`
output_vector ← {} ;
`/* initialization */`
**foreach** *f* ∈ *list_features* **do**
    output_vector[*f*] ← 0;
**end**
`/* vectorization */`
**foreach** *(occ, oScore)* ∈ *oVector* **do**
    *occ_features* ← all features (in the O*NET tOutput database) with their scores, related to *occupation* ;
    **foreach** *(f, fScore)* ∈ *occ_features* **do**
        output_vector[*f*] ← max (output_vector[f], oScore×fScore);
    **end**
**end**
**return** *output_vector*

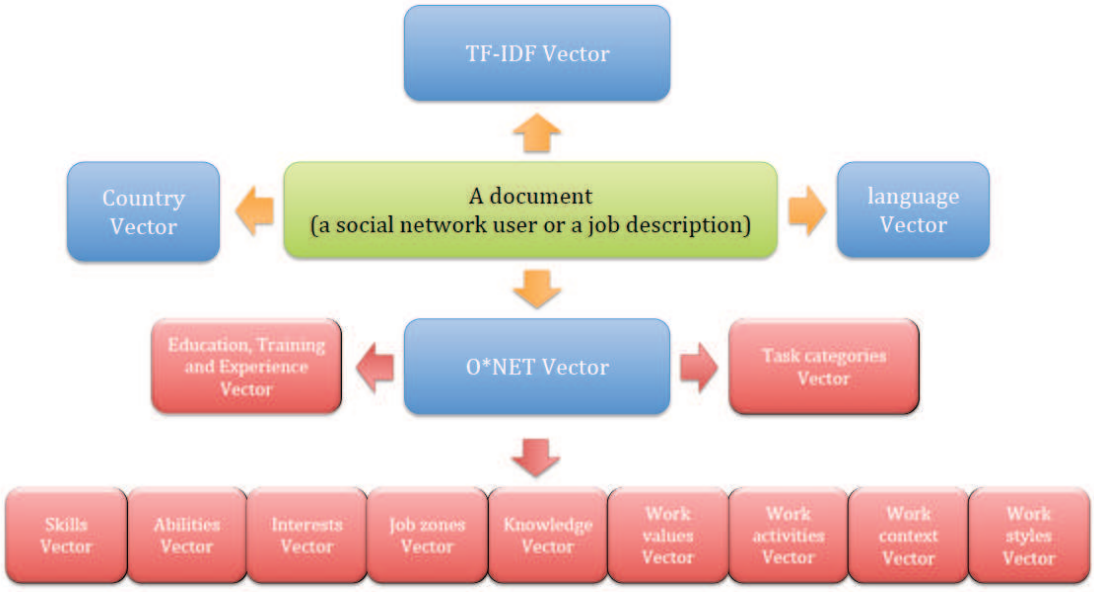



Figure 5.30 – Multilayer vector model for social network users and job descriptions: the information in the different fields of the users and jobs are used to extract the TF-IDF, O*NET, language and country vectors. We then use the O*NET vector to construct the other O*NET related vectors like abilities, skills and interests vectors.

where the 0.5 means that we do not know if $u$ and $v$ are matching since at least one of them is empty.

The similarity between the other vectors is computed using the cosine similarity (that measures the cosine of the angle between two vectors) defined by eq. (4.15).

After extracting the multilayer vectors for users and jobs, we need to encode each couple (user, job) in order to be able to use SVMs algorithms, we finally decided to represent the profile of a couple (user, job) using the following 15 attributes:

1. cos(user_tfidf_vector, job_tfidf_vector): TF-IDF Matching score (also the similarity obtained by Engine-1).
2. cos(user_onet_vector, job_onet_vector): O*NET Matching score (also the similarity obtained by Engine-6a).
3. intersection_sim(user_lang_vector, job_lang_vector): Language Matching score.
4. intersection_sim(user_country_vector, job_country_vector): Country Matching score.
5. cos(user_skills_vector, job_skills_vector): Skills Matching score.
6. cos(user_abilities_vector, job_abilities_vector): Abilities Matching score.
7. cos(user_interests_vector, job_interests_vector): Interests Matching score.
8. cos(user_job_zone_vector, job_job_zone_vector): Job-Zone Matching score.
9. cos(user_knowledge_vector, job_knowledge_vector): Knowledge Matching score.
10. cos(user_work_values_vector, job_work_values_vector): Work-values Matching score.
11. cos(user_work_activities_vector, job_work_activities_vector): Work activities Matching score.
12. cos(user_work_context_vector, job_work_context_vector): Work context Matching score.
13. cos(user_work_styles_vector, job_work_styles_vector): Work style Matching score.
14. cos(user_task_categories_vector, job_task_categories_vector): Task categories Matching score.
15. cos(user_education_training_experience_vector, job_education_training_experience_vector): Education and experience Matching score.

One can easily add additional attributes to this proposed vector model. We note that one can use each of the attribute scores in the profile a couple (user, job) as the similarity score between the user and the job, in other word, each attribute can be used to make job recommendation as shown by the Table 5.3: it reveals that matching the O*NET vectors (of users and jobs) generally yields better results than matching TF-IDF vectors and outperforms matching the other O*NET related vectors like abilities, skills and interests vectors. Note that methods based on matching user-job languages, countries, job-zones are not enough accurate in job recommendation to

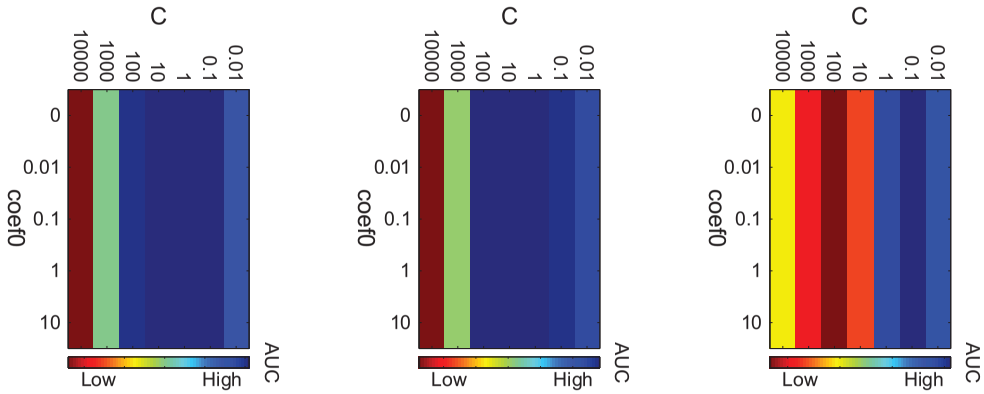
| | | Heuristic Job Recommendation Methods | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Social Networks | User-Job TF-IDF Matching (Engine-1) | User-Job O*NET Occupations Matching (Engine-6a) | User-Job Skills Matching | User-Job Abilities Matching | User-Job Interests Matching | User-Job Job zones Matching | User-Job Knowledge Matching | User-Job Work Activities Matching | User-Job Work Context Matching | User-Job Work Styles Matching | User-Job Education Experience Matching |
| Review | Any | 0.72±0.02 | **0.76±0.01** | 0.66±0.02 | 0.66±0.02 | 0.67±0.02 | 0.64±0.01 | 0.68±0.01 | 0.66±0.02 | 0.67±0.02 | 0.64±0.02 | 0.62±0.02 |
| | Facebook | 0.70±0.02 | **0.75±0.02** | 0.66±0.02 | 0.67±0.02 | 0.66±0.04 | 0.65±0.03 | 0.68±0.03 | 0.66±0.04 | 0.67±0.03 | 0.64±0.03 | 0.62±0.03 |
| | LinkedIn | 0.74±0.03 | **0.77±0.02** | 0.66±0.02 | 0.65±0.01 | 0.67±0.02 | 0.64±0.01 | 0.69±0.02 | 0.67±0.02 | 0.66±0.02 | 0.65±0.02 | 0.62±0.02 |
| Validation | Any | 0.78±0.01 | **0.82±0.01** | 0.71±0.01 | 0.68±0.02 | 0.69±0.02 | 0.65±0.01 | 0.73±0.01 | 0.69±0.02 | 0.70±0.02 | 0.67±0.01 | 0.63±0.01 |
| | Facebook | 0.77±0.02 | **0.80±0.02** | 0.70±0.02 | 0.70±0.04 | 0.69±0.03 | 0.67±0.02 | 0.72±0.03 | 0.70±0.02 | 0.71±0.03 | 0.68±0.03 | 0.63±0.04 |
| | LinkedIn | 0.80±0.02 | **0.83±0.02** | 0.71±0.01 | 0.68±0.02 | 0.69±0.02 | 0.65±0.01 | 0.74±0.02 | 0.69±0.01 | 0.70±0.02 | 0.67±0.02 | 0.64±0.01 |
| ALL | Any | 0.40±0.01 | **0.55±0.01** | 0.45±0.01 | 0.44±0.01 | 0.44±0.01 | 0.42±0.01 | 0.46±0.01 | 0.45±0.01 | 0.44±0.01 | 0.44±0.01 | 0.43±0.01 |
| | Facebook | 0.34±0.01 | **0.53±0.01** | 0.44±0.01 | 0.44±0.01 | 0.44±0.01 | 0.43±0.01 | 0.45±0.01 | 0.44±0.01 | 0.44±0.01 | 0.44±0.01 | 0.42±0.01 |
| | LinkedIn | 0.74±0.01 | **0.76±0.01** | 0.66±0.01 | 0.64±0.01 | 0.65±0.01 | 0.62±0.01 | 0.69±0.01 | 0.66±0.01 | 0.65±0.02 | 0.64±0.01 | 0.61±0.01 |

Table 5.3 – Comparison (on Review, Validation and ALL datasets) between heuristic-based methods using the different proposed vector models. Results for Work values, Language and Country matching are lower than those of the other vector models.

social network users, as a result, Table 5.3 only contains the most interesting results (to make it more readable).

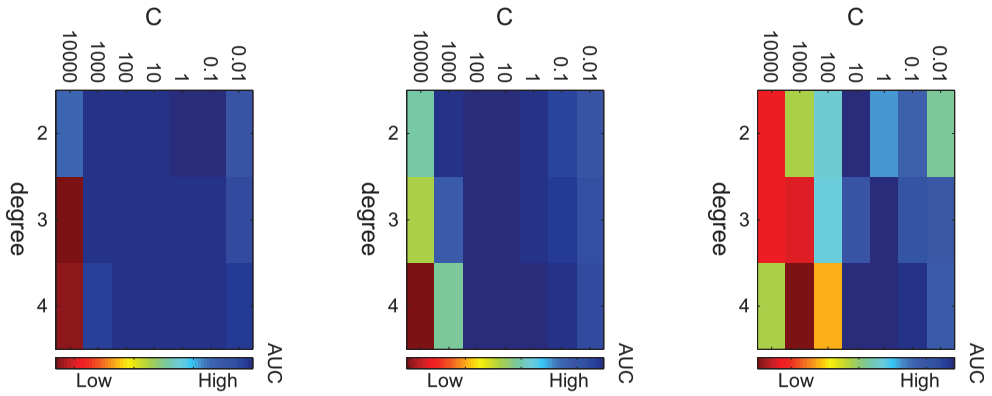
To improve the job recommendation made by each attribute, we use the SVM algorithm to develop models that aggregate the scores of the 15 attributes to make job recommendation: this is similar to the idea used in [Malherbe et al., 2014]. We call these SVM-based recommender systems *Engine-8*. Once again, we use three kernels for SVMs (see eq. (4.24)): Linear-kernel, Poly-kernel and RBF-kernel. We optimize the hyper-parameters of SVMs using 10-fold Cross-validation and Grid search (see section 4.3.4) to ensure that our different models are correctly fit (see Figure 5.31, 5.32 and 5.33).

Figures 5.34c, 5.34b, 5.34a show that the use SVMs improves the quality of our job recommendation to social network users compared the heuristic-based methods (Engine-6a for instance). Note that non-linear SVMs slightly outperforms the linear models on our 3 datasets. The results of these experiments show that it is possible to successfully aggregate using machine learning techniques the matching scores of the different layers (abilities, skills, educations, experience, etc.) of user and job profiles to improve the quality of job recommendation made to social network users.



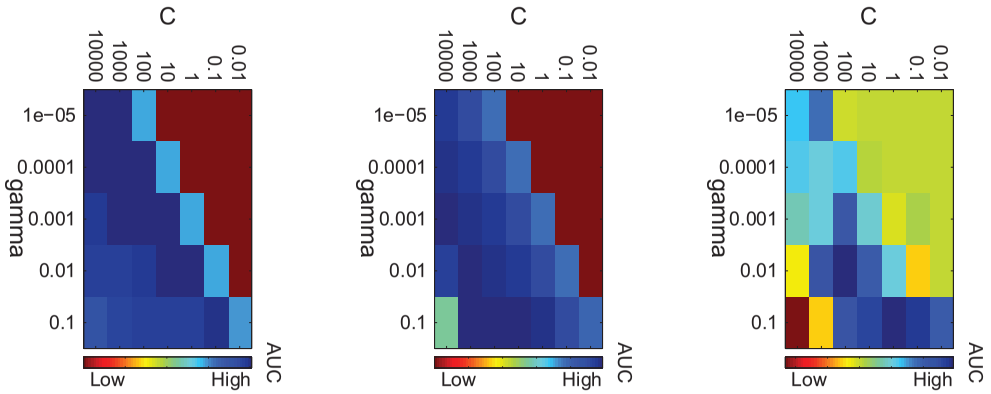

(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.31 – Optimization of hyper-parameters of Linear SVMs using our proposed multilayer vectors. Note Coef0 which corresponds to $r$ in eq (4.24a) and $C$ to the regularization parameter in eq (4.22). One can note the impact of Coef0 on the performance of our recommender systems is really very small compared to that of the regularization parameter $C$.



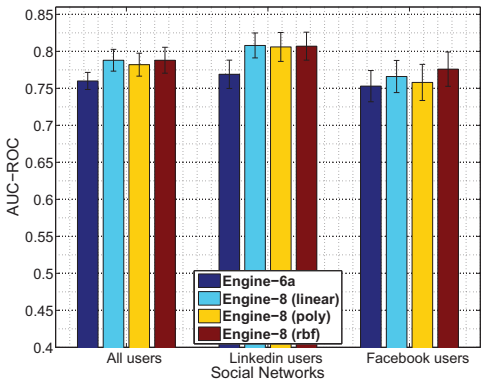

(a) Review dataset (b) Validation dataset (c) ALL dataset

Figure 5.32 – Optimization of hyper-parameters of Poly SVMs using our proposed multilayer vectors. Note Coef0 which corresponds to $r$ in eq (4.24b) is set to 1, degree corresponds to $d$ in eq (4.24b) and $C$ to the regularization parameter in eq (4.22).

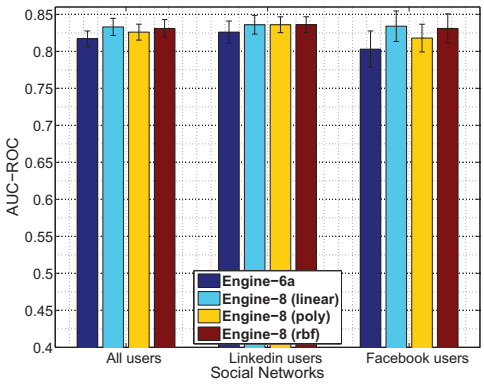



(a) Review dataset (b) Validation dataset (c) ALL dataset

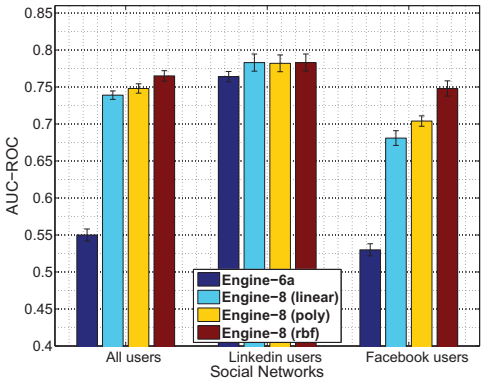
Figure 5.33 – Optimization of hyper-parameters of RBF SVMs using our proposed multilayer vectors. Note gamma corresponds to $\gamma$ in eq (4.24c) and $C$ to the regularization parameter in eq (4.22).




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.34 – Engine-8: comparison between different kernels of SVMs and Engine-6a (which yields the highest performance, see Table 5.3). One can note a slight out-performance of the RBF kernel over the other kernels.

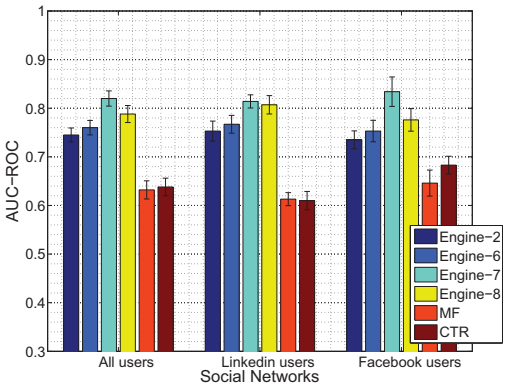
### 5.5.4 Comparison between the proposed recommender systems (based on O*NET taxonomy) and two state-of-the-art systems

We proposed several job recommender systems using O*NET taxonomy (Engine-6, Engine-7 and Engine-8) to address the problem of missing data and that of mismatching between users' and jobs' vocabulary. In this section, we analyze the performance of the proposed systems and compare them to other recommender systems from the literature.

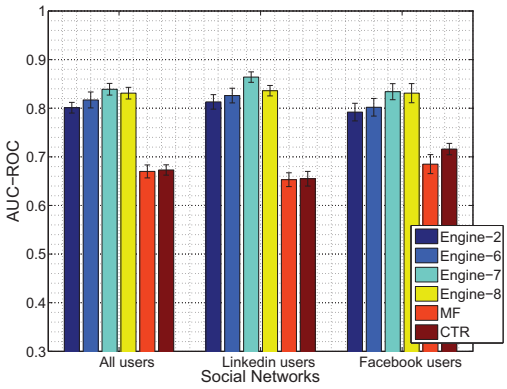
Our experiments revealed that the use of a taxonomy-based vector model (Engine-6) improves the quality of job recommendation to social network users compared to Engine-2 (see Figure 5.35): we observe a slight improvement on Review and Validation datasets and huge improvement on ALL dataset. This graph also shows that the quality of job recommendation using LinkedIn data is higher than using Facebook but the use of our O*NET vector model drastically reduces the difference of quality of data between the two social networks in the task of job recommendation on our datasets (mainly on ALL dataset): we can conclude that the use of taxonomy-based vector model clearly improves our results.

We can observe on Figure 5.35 that the use of learnt statistical models from data using our taxonomy-based vector model (Engine-7) improves the quality of job recommendation compared to heuristic-based recommendation (Engine-2 and Engine-6). However the use of SVM to aggregate the different matching criteria scores (Engine-8) yields results slightly lower than those of Engine-7.

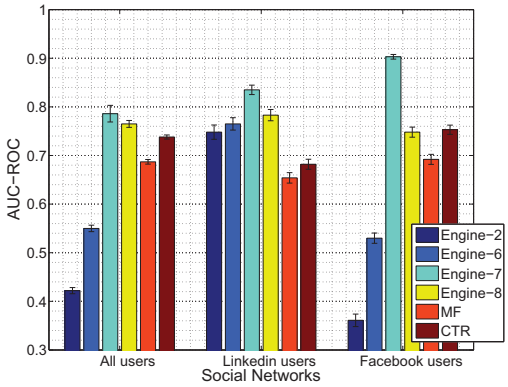
The comparison of Engine-2, Engine-6, Engine-7 and Engine-8 to Matrix Factorization (MF) and collaborative Topic Regression (CTR) (see Figure 5.35) showed that Engine-2 and Engine-6 outperform MF and CTR on the Review and Validation datasets but they are outperformed by MF and CTR on ALL dataset; this is due to the fact that Engine-2 and Engine-6 have some difficulties to correctly find the right labels in Candidate dataset. As for Engine-7 and Engine-8, they clearly outperform MF and CTR on the 3 datasets: the use SVM allows a significant improvement of the quality of job recommendation using a taxonomy-based vector model.




(a) Review dataset.




(b) Validation dataset.




(c) ALL dataset.

Figure 5.35 – Comparison between Engine-2, Engine-6, Engine-7, Engine-8, MF and CTR. Engine-2 (see section 5.4.2) Note that Engine-6 is using cosine similarity and Engine-7 and Engine-8 are using the RBF kernel since this kernel outperforms the others (see Figures 5.29 and 5.34).

## 5.6 Conclusion

We studied the literature about recommender systems, data mining and machine learning to propose several job recommender systems on social networks.

Our first job recommender systems are based on the bag-of-word model. The first series of experiments concluded that TF-IDF weighting function together with cosine similarity outperforms the other combinations of weighting and similarity functions (see Figure 5.9): we chose TF-IDF as weighting function and cosine as heuristic similarity. Lemmatization has been chosen as preprocessing technique to reduce the dimensionality of our problems. We showed that some fields of social network users and jobs are more important than others in the task of job recommendation (see Figure 5.11), taking this into account significantly improves the results (compared to Engine-1) but the quality of recommendations were still poor (see sections 5.4.2 and 5.4.6).

Our experiments on SVM revealed that the use of models based on machine learning together with TF-IDF vectors (Engine-3) drastically improves the quality of recommendations compared to heuristic-based recommender systems (Engine-1 and Engine-2). Engine-3 also outperforms two state-of-the-art recommendation techniques CTR and MF proposed by [Wang and Blei, 2011] (see Figure 5.23).

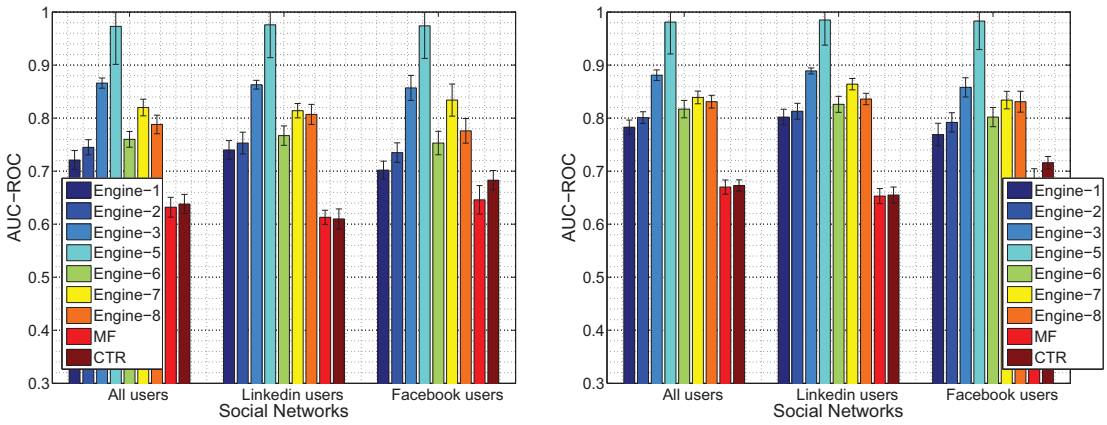
Our attempt to use of basic methods of social recommendation (Engine-4) failed to improve our results (compared to Engine-1), we could not use complex methods of social recommendation due to the nature of our data (privacy preservation). However, we showed that the use of relevance feedback (Engine-5) mitigates the problem of missing data, and therefore, drastically improves the quality of recommendations; 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 (see section 5.4.4 and Figure 5.23).

User term space could be not completely equal to the job term space, to address this problem, we use the O*NET taxonomy (see appendix B) to develop a new taxonomy-based vector model for social network users and job descriptions suited to the task of job recommendation. Our experiments (see Figure 5.25) concluded that the cosine similarity yields results slightly better than the proposed fuzzy logic-based similarity functions using our proposed O*NET vector model; they revealed that the use of our taxonomy-based vector model improves the performance of our job recommender systems compared to the TF-IDF and drastically reduces the difference of quality between our Facebook and LinkedIn data in the task of job recommendation (see Figure 5.35). We showed that using models based on machine learning (SVM) leads to job recommender systems (Engine-7) that outperforms our heuristic-based systems and two methods of the literature (CTR and MF) (see section 5.5.4).

We also proposed a multilayer vector model combining heterogeneous data from Facebook users, LinkedIn users and Job descriptions, based on the taxonomy O*NET to tackle the multiple criteria aspect of job recommendation. Our experiments revealed that the direct

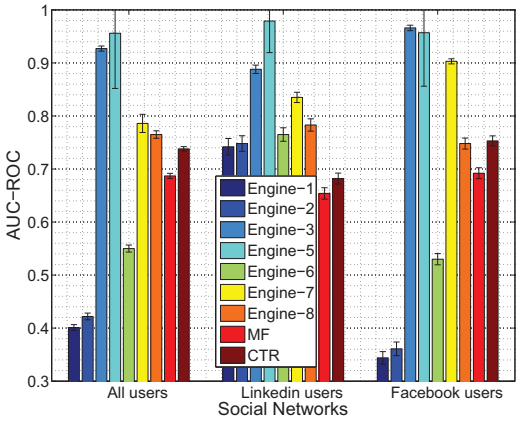
matching between users and job country, language, TF-IDF, O*NET, O*NET related vectors gives results that can be improved by combining them into a model learnt using Support Vector Machines algorithms (Engine-8). They also showed that the use of non-linear kernels for SVMs gives a slight improvement compared to the models based on linear SVM (see section 5.5.3).

Globally, the quality of job recommendation to social network users using Facebook data is lower than using LinkedIn data, this can be explained by the fact that contrary to Facebook, LinkedIn users use this platform for a professional purpose, as a result, they publish correct and detailed information about their work and education histories. We present a global conclusion about job recommendation to social network users in the section 7.1.




(a) Review dataset.

(b) Validation dataset.




(c) ALL dataset.

Figure 5.36 – Comparison between Engine-1, Engine-2, Engine-3, Engine-5, Engine-6, Engine-7, Engine-8, MF and CTR. Engine-3, Engine-7 and Engine-8 are using the RBF kernels (since we obtained highest performance for this kernel).

# 6 Prediction of the Audience of Job Advertisements on Social Networks

*"Big data is mostly about taking numbers and using those numbers to make predictions about the future. The bigger the data set you have, the more accurate the predictions about the future will be."*

- Anthony Goldbloom, Founder and CEO of Kaggle

**Contents**



## 6.1 Introduction

In the chapter 5, we developed and studied systems to make direct job recommendation to social network users. These systems require to have data about target users but social network users are more and more reluctant to let third party applications having access their profile data. Based on this observation, Work4 has developed applications (see chapter 2) that allow

organizations to post their job advertisements on social networks like Facebook [Facebook, 2015], LinkedIn [LinkedIn, 2015] and Twitter [Twitter, 2015], any social network user can then click on, explore and apply to posted ads. It worth noting that we cannot access the data of users who explore, click on or apply to jobs posted on social networks using these applications.

Posting their ads on social networks is offering to organizations a great opportunity to reach thousands or even millions of users at the same time, reason why social networks are capturing a growing part of the advertisement market. Advertising their job opportunities may allow organizations to reduce their recruitment costs by speeding up their hiring process.

We need to find out what make a given job advertisement (posted on a social network) popular in order to optimize the process of posting job ads on social networks. In this context, we propose a set of decision support systems called Work4Oracle that predict an estimation of the number of clicks a given advertisement should obtain, we applied them to predict the audience of job advertisements posted on social networks. Our system has been designed by combining heterogeneous data from different sources about jobs to be advertised, their organizations (age, income, revenue, industry, ...) and countries (unemployment rate). The descriptions of jobs (stored in the databases of Work4) generally contain information about the related positions, organizations' and countries' names, language of jobs, type of contracts (full time for instance) and requirements for the positions. We show how external sources of information, like Wikipedia or specialized websites can be used to enrich the profiles of ads. In our application, adding information (from external sources) about organizations and countries proved to be effective for the audience prediction. The main objective of our proposed system is to help recruiters optimizing the process of advertising their job offers to social network users by finding the right moments to post, the right persons to post a job, the right jobs to post for a specific poster, etc.

In this chapter, we focus on modeling job ads, learning models to predict their audience using machine learning and collaborative filtering techniques and quantifying the factors impacting the popularity of job ads on social networks. We fit predictive models using machine learning techniques together with the data of more than 150,000 job ads posted on social networks by Work4[1] on behalf of its customers between January 2013 and June 2014, and compare the findings to those of studies done in the literature of Human Resource Management.

## 6.2 Problem Statement

We call *audience* of a job advertisement, the number of clicks of social network users on this post. As stated earlier, we aim at developing a decision support system called Work4Oracle, able to predict a good estimation of the audience of job ads on social networks. Using Work4's applications, job ads can be posted on 3 social networks namely Facebook, LinkedIn and Twitter but for Facebook, posters can post their job advertisements either on their own

[1] http://www.work4labs.com

Facebook walls (Facebook-profile) or on Facebook page (Facebook-page) of their organizations; this leads us to define our set of networks as $N = \{Facebook-page, Facebook-profile, LinkedIn, Twitter\}$. The problem we tackle in this thesis can be formally stated as follows:

$$\forall n \in N,\ \Gamma_n : P \times J \times D \rightarrow \mathbb{N} \tag{6.1}$$

where $\Gamma_n$ is an audience function for the network $n$, $P$, $J$ are respectively the set of posters (users) and jobs, $D$ is the set of dates and $\mathbb{N}$ is the set of all natural numbers. $\Gamma_n(p, j, d)$ represents the audience of the job $j$ posted on the social network $n$ by the poster $p$ at the date $d$.

After finding the optimal $\Gamma_n$ for each network $n$, we can then apply them to the following problems:

1. For a given a network $n$, a poster $p$, a job $j$ and a date $d$, find an estimation of the number of clicks the post should obtain:

$$\text{number of clicks} = \Gamma_n(p, j, d) \tag{6.2}$$

2. For a given poster $p$, find the right network to post a job $j$ at a date $d$:

$$n^\star = \arg\max_{n \in N} \Gamma_n(p, j, d) \tag{6.3}$$

3. For a given job $j$, a network $n$ and a date $d$, find the right poster:

$$p^\star = \arg\max_{p \in P'} \Gamma_n(p, j, d) \tag{6.4}$$

4. For a given poster $p$, a network $n$ and a date $d$, find the right job to post:

$$j^\star = \arg\max_{j \in J'} \Gamma_n(p, j, d) \tag{6.5}$$

5. For a given poster $p$ and a network $n$, find the right moment to post a job $j$:

$$d^\star = \arg\max_{d \in D'} \Gamma_n(p, j, d) \tag{6.6}$$

where $P'$ is the set of potential posters, $J'$ is the set of available jobs and $D'$ is the set of desired dates.

## 6.3 Attractiveness of organizations for applicants

Many studies have been dedicated to the identification of the factors impacting the attractiveness of organizations for people seeking jobs (which may impact the popularity of their job advertisements), most of them are from the human resource management literature and are generally based on subjective analysis which makes difficult their generalization. One of the goals of our study is to adapt the previous work in the special case of social media-based recruitments (recruitments on new social media like Linkedin, Facebook and Twitter) using machine learning methods on real-world data collected by the company Work4.

Reference [Séguela, 2012] investigated factors impacting the performance of ads campaigns in recruitment context and highlighted some interesting factors: job message, target of job ads (locally-targeted recruitment ads generally perform better than global ads according to [Rafaeli et al., 2005]), type of jobs (location, salary, industry, category, type of contract, etc.), reputation, size and image as employer of the organizations that propose the jobs and organizations' recruitment websites.

Roberson et al. studied the impact of recruitment messages on applicant attraction to organizations: they found that specific recruitment messages would lead to higher perceptions of their organizations' attractiveness. Williamson et al. conclude that web sites' orientation influences organizational attractiveness perceptions which supports the findings of [Cober et al., 2000] who argued that the information displayed to potential applicants on recruitment web sites influence their attractiveness. Stone posited that organizations' web sites would be an efficient tool for recruitment. Note that organizations' web sites target active candidates (people who are seeking jobs). To better target passive candidates, organizations are increasingly using web-based recruitment services mainly social media like Facebook and LinkedIn.

References [Chapman et al., 2005; Ehrhart and Ziegert, 2005] posited that the image and size of an organization impacts its attractiveness while [Turban et al., 1998] studied the influence of organizations' reputation on applicants' attraction to firms and showed that the perception of jobs and organizational attributes influences applicants' attraction as well as the organization reputation. They also showed that applicants' attraction could be impacted by the behaviors of recruiters. We can assume that the image of an organization and its reputation can impact the audience of its jobs posted on Internet and especially on social networks. [Lievens and Highhouse, 2003; Mathews and Redman, 1998] showed that the location, the salary and the description of jobs can impact their attraction for future applicants.

Recently, [Bernstein et al., 2013] studied the correlation between different parameters and the audience of any posts on social networks, they found that social media users generally underestimate their audience size. Their study also indicates that only friend counts of social network users are not very accurate in estimating the audience of their posts. In the particular case of posting of jobs on social networks, we can also assume that the friend count of the posters and the interests of their friends for the posted jobs can impact the audience of posts.

Reference [Wang et al., 2013] studied how to detect the moment of a career switch for a specific user in order to recommend relevant jobs to him at that moment using hierarchical proportional hazards model [Fine and Gray, 1999; Lin and Wei, 1989; Therneau, 2000] together with data collected by LinkedIn.

## 6.4 Study of the dataset about job advertisements

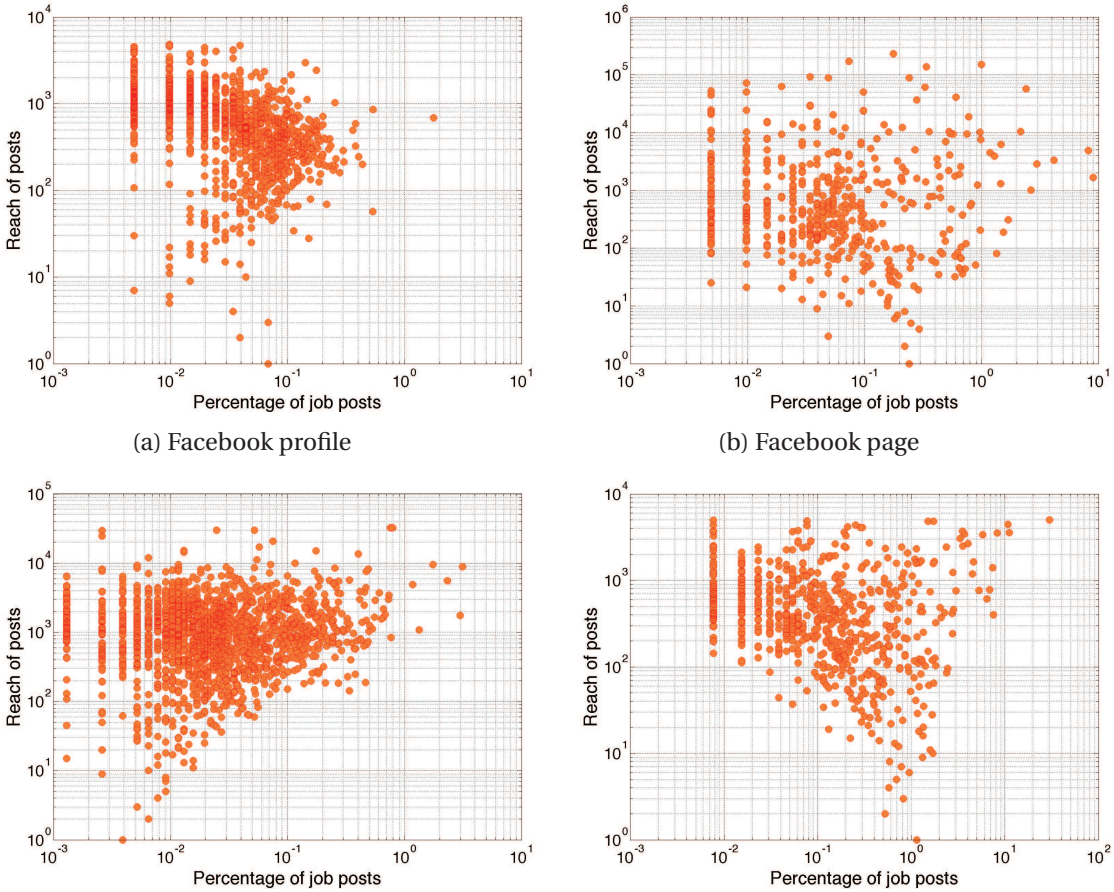
We evaluate the performance of our proposed systems on a dataset collected by Work4 between January 2013 and June 2014, this dataset is fully anonymized and is a subset of the company's collected data. Each entry in the dataset is a 5-tuple $(u, j, n, d, y)$ where $u$ and $v$ are a social network user (poster) and job respectively and $n \in \{Facebook-page, Facebook-profile, LinkedIn, Twitter\}$ is the network on which the job has been posted, $d$ is the date of the post and $y$ is the number of clicks on the post. We clean up our dataset by removing posts from Work4's developers and testers, finally we obtain a dataset with 152,382 job ads. Table 6.1 reports the summary statistics of our dataset.

| | LinkedIn | Facebook-Page | Facebook-Profile | Twitter | **Total** |
|---|---|---|---|---|---|
| Number of users | 3,205 | 512 | 3,376 | 1,856 | **5,729** |
| Number of jobs | 34,423 | 14,691 | 9,030 | 29,256 | **71,545** |
| Number of job pages | 1,002 | 490 | 1,084 | 948 | **1,824** |
| Number of posts in 2013 | 37,683 | 7,810 | 4,418 | 19,031 | **68,942** |
| Number of posts in 2014 | 39,150 | 12,583 | 8,555 | 23,152 | **83,440** |
| Total number of posts | 76,833 | 20,393 | 12,973 | 42,183 | **152,382** |
| 1st quintile | 1 | 1 | 1 | 1 | Number of clicks |
| 2nd quintile | 2 | 2 | 1 | 1 | |
| 3rd quintile | 3 | 4 | 1 | 2 | |
| 4th quintile | 5 | 9 | 2 | 3 | |

Table 6.1 – Statistics extracted from our dataset; the numbers of posts, posters and jobs for different social networks and for 2013 and 2014. We computed the quintiles of the number of clicks obtained by our job advertisements.

As presented in the section 6.5, each job ad is associated to a social network account that has a specific reach (number of persons who can see the posts of this account). Figure 6.1 shows the distribution of reach of jobs posted on Facebook, LinkedIn and Twitter. We note that jobs posted on organizations' Facebook pages (Facebook-page) obtain the highest reach on average (mean=7,277) and the highest variability of the reach (std=22,707), they are followed by job ads posted on LinkedIn, Twitter and on Facebook personal walls (Facebook-profile). Observing Figure 6.2, we note that our jobs are mainly posted in the day between 9am and 9pm, this is a specificity of the Work4's job posting algorithm.

We call half-life of job ads, the number of hours (after posting it on a social network) required



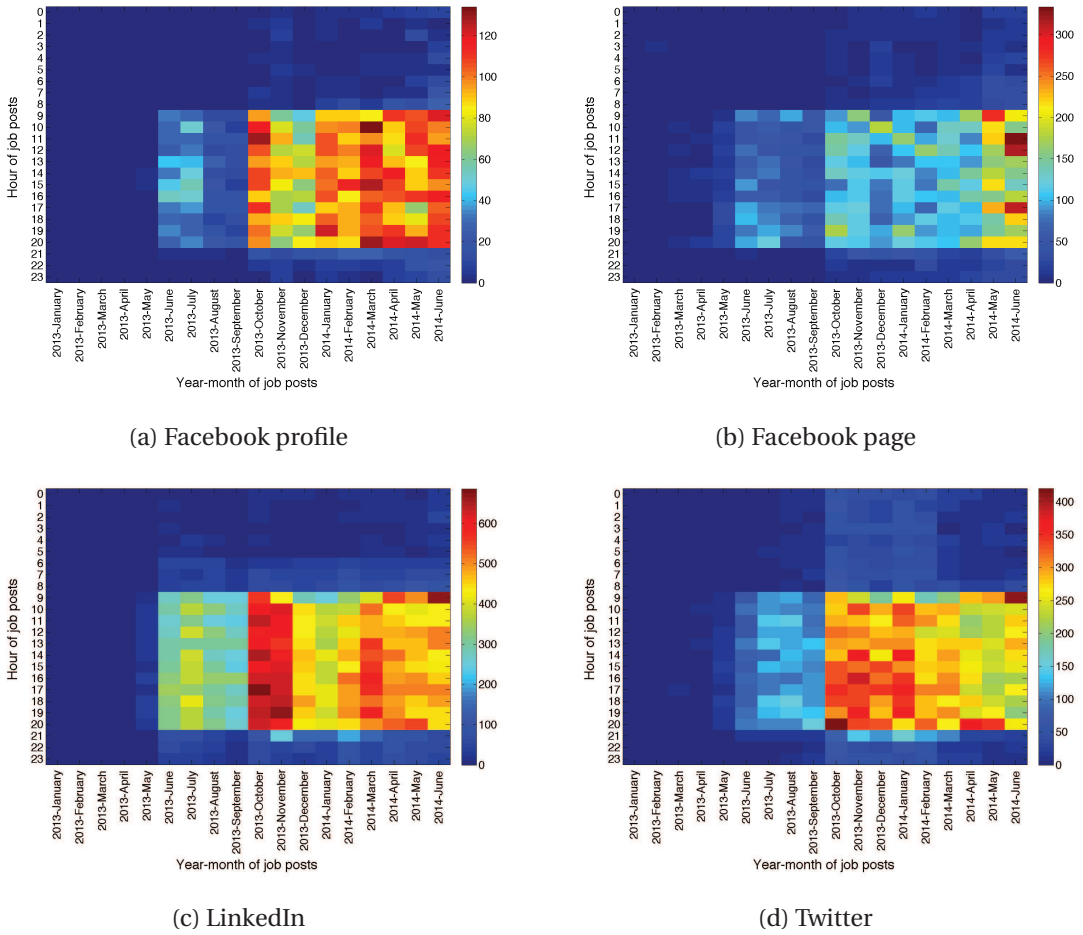

(a) Facebook profile

(b) Facebook page

(c) LinkedIn

(d) Twitter

Figure 6.1 – Distribution of the reach (number of persons who can see a given job advertisement) in our dataset.




(a) Facebook profile

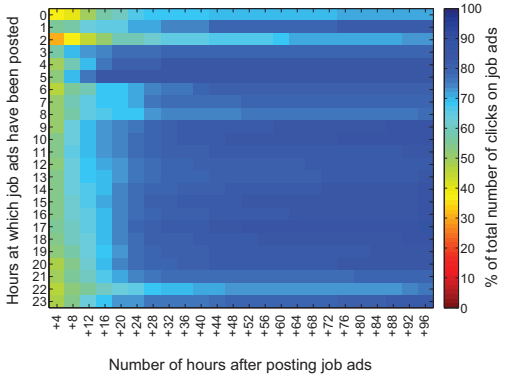
(b) Facebook page

(c) LinkedIn

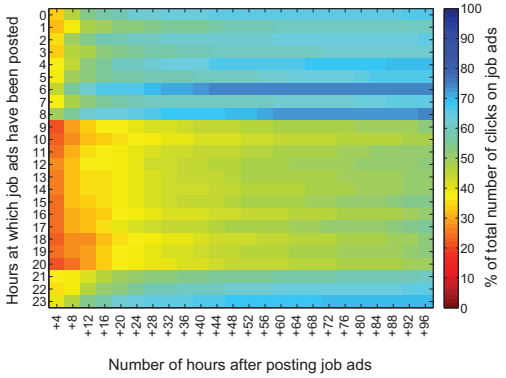
(d) Twitter

Figure 6.2 – Distribution of posts (job ads) in our dataset from January 2013 to June 2014.

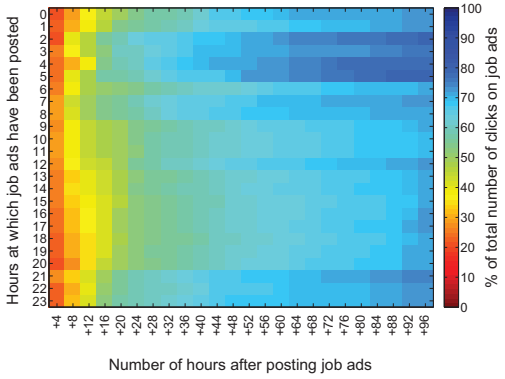
to get the half of the total number of clicks it obtained. Figure 6.3 compares the half-life of job advertisements posted on Facebook users' walls to those of job ads posted on LinkedIn, Twitter and organizations' Facebook pages. It reveals that the half-life of job posts on Facebook users' walls is very short (about 4 hours) compared to the half-life of those posted on organizations' Facebook pages (more than 96 hours), half-lives of job ads posted on LinkedIn and Twitter are similar (about 36 hours). The fact that the half-life of posts on Facebook users' walls is so short is probably due to the fact that posts on Facebook users' walls are rapidly "buried" by the posts of users' friends.




(a) Job ads posted on walls of Facebook users.




(b) Job ads posted on organizations' Facebook pages.




(c) Job ads posted on linkedIn.


(d) Job ads posted on Twitter.

Figure 6.3 – Evolution of percentage of total number of clicks on job ads posted on Facebook. By defining the half-life of job ads as the number of hours required to get the half of the total number of clicks it obtained, we note that the half-life of job ads posted on the walls of Facebook users is by far shorter than the half-life of those posted on LinkedIn and Twitter which are shorter than those of job ads posted on organizations' Facebook pages.

## 6.5 Modeling of job advertisements

Each job advertisement is a 5-tuple (poster, job, network, date_of_post, number_of_clicks), our goal is to learn a model to predict the variable number_of_clicks for each social network. For each job post defined by (poster, job, network, date, number_of_clicks), we extract a profile (mainly based on binary weighting function (see section 4.2.2)) using the information about the poster, job, network on which the job ad has been posted and the date of the post, this step is known as vectorization. The extracted profile will characterize the job post and will be used to predict its audience. Formally, we define the profile of a job ad as a set of:

1. The profile of its poster (with *1,046 dimensions*).

2. The profile of its job (with *1,744 dimensions* for information in the databases of Work4 and *157 dimensions* for information retrieved from Internet).
3. The profile of the date at which it has been or will be posted (with *45 dimensions*).
4. The matching vector between the poster's and job's O*NET vectors (with *1,040 dimensions*).

Now let us see how the profiles of posters, jobs and dates are extracted in our proposed systems.

#### Profiles of posters

Each social network user is related to some occupations and has an associated social network account. We call reach of an account the number of persons who can see the posts of this account, it corresponds to:

1. The friend count of a Facebook user, if the post is done on his Facebook wall.
2. The number of persons who liked the organization's Facebook page on which the job post has been done.
3. The number of followers on Twitter.
4. The number of relations on LinkedIn.

Due to privacy concerns, we cannot generally access the data of the users connected to a specific social network user, that's why we only use the reach of accounts in this study. We could not profile more finely the users to whom the jobs are advertised but based on the principle of homophily in social networks ("Birds of a feather flock together") [McPherson et al., 2001], the profile a poster (user) can give a clue on the profiles of his social connections (generally friends).

We define the *profile of a user/poster* (see Figure 6.4) as the set of:

- His *O*NET vector* (see section 5.5).
- The *vector of the reach* of his associated account.

*Note that the O*NET vector of posters are only interesting for ads posted on LinkedIn and Facebook users' walls since the Facebook accounts (data of which are used to extract O*NET vectors) allowing to post ads on Twitter and organizations' Facebook pages are generally that of recruiters.*

We propose the following encoding for the reach of job posts:

$$\forall i \in \{0,...,d-1\}\ v_i(r,d) = \begin{cases} 1 & \text{if r} > 0 \text{ and i} = \lfloor \log_{10}(r) \rfloor \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

where $v(r,d) = (v_0(r,d),...,v_{d-1}(r,d))$ is the vector associated to the reach $r$ ($\geq 0$) for a number of dimensions $d$. After analyzing the distribution of reaches in our dataset, we noticed that the reach values are ranging from $10^0$ to $10^6$ (exclusive), so we set the number of dimensions $d$ of the vectors of reach to 6. To understand how this encoding works, let us see the following examples: 9 is encoded as $[1,0,0,0,0,0]$, 10 as $[0,1,0,0,0,0]$ and $999,999$ as $[0,0,0,0,0,1]$.

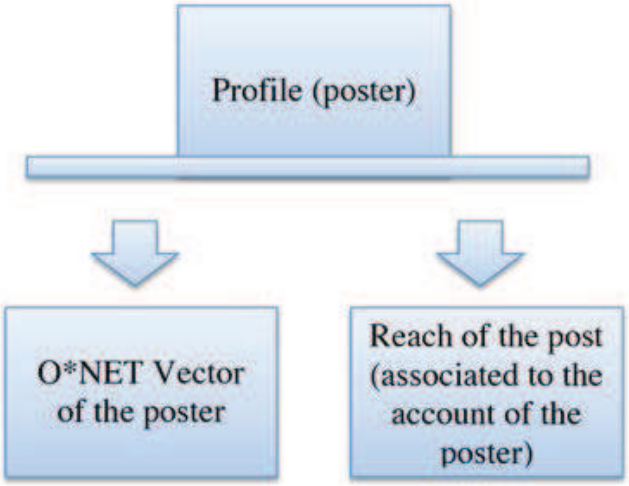



Figure 6.4 – Representation of the profile of posters (with *1,046 dimensions*) using the O*NET Taxonomy and the vector space model with binary weighting function. We have *1,040 dimensions* for O*NET vectors and *6* for the reach of the associated social network account. It is important to note that the value of each component is between 0 and 1.

### Profiles of jobs

Related work (see section 6.3) showed that the attractiveness of a job generally depends on some factors like its organization's name and reputation, the salary, title and industry of the job. We extract the profile a job based on two types of data:

1. Information about the job and its organization stored in the databases of Work4.
2. Additional information about the organizations and countries available on Internet.

**Modeling of data about jobs in the databases of Work4:** for information about jobs and their organizations stored in the databases of Work4, we define 4 sub-vectors for a specific job (see Figure 6.5): the *O*NET vector* (see section 5.5) linked to its title, *contract type*, *country* and *company name*.

Our 4 types of contracts (Full Time, Temporary, Internship and Part Time) are encoded on 4 dimensions using binary vector model. Example: vector("Full Time") = {("Full Time", 1)}. Similarly, our 500 distinct organizations' names and 200 targeted country names are respectively encoded on 500 and 200 dimensions using binary weighing function.

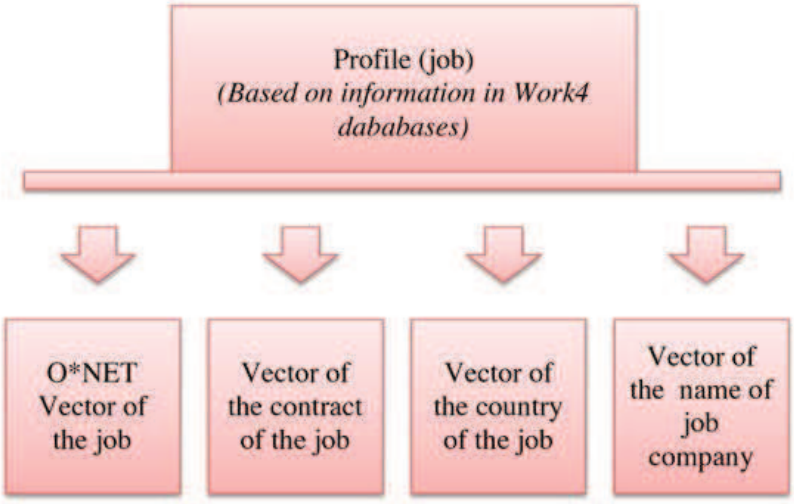



Figure 6.5 – Representation of the profile of a job based on the information provided by Work4 (with *1,744 dimensions*) using the O*NET taxonomy and the vector space model with binary weighting function. We have *1,040 dimensions* for O*NET vectors, *4* for the contract type, *200* for the country names, *500* for the company names. The value of each component is between 0 and 1.

**Modeling of data about jobs retrieved from Internet:** we retrieve additional information about organizations and countries available on Internet from some websites (mainly on Wikipedia.org [2] and on the Organization for Economic Co-operation and Development web site (OECD) [3]) and encode them in a vector with *157 dimensions*.

The *job country unemployment rate*[4] at the period (year-month) of a job post: this represents 1 dimension and we normalize its values by dividing by the max unemployment rate in our dataset, this ensures us to have values between 0 and 1.

The *age of a job organization* (based on the organization creation date on Wikipedia.org) is encoded by 5 dimensions (quintiles) using binary weighting function. On organizations' Wikipedia pages, we can find their *area-served*, *types* and *industries*. Our 20 distinct area-served, 20 types and 80 industries are respectively encoded on 20, 20 and 80 dimensions using binary weighting function. Wikipedia also generally gives the *number of employees* of an organization and its financial information (*revenue*, *income*, *operating income*, *asset* and *equity*). We encode the *number of employees* as the reach of posts on 6 dimensions (see

[2] http://en.wikipedia.org
[3] http://www.oecd.org
[4] The unemployment rates used in this thesis have been retrieved from the OECD web site (only concerning OECD countries but almost all our job ads have been done for OECD countries).

section 6.5), each of the financial information is encoded by 5 dimensions (quintiles) using binary weighting function.

### Profiles of dates of job ads

The date of job posts have 6 components: year, month, day, hour, minute, second. In our study, we ignored the minute and second components of posts since they might not impact the audience of job posts (see Figure 6.6). One notes that we have at most 31 days in a month

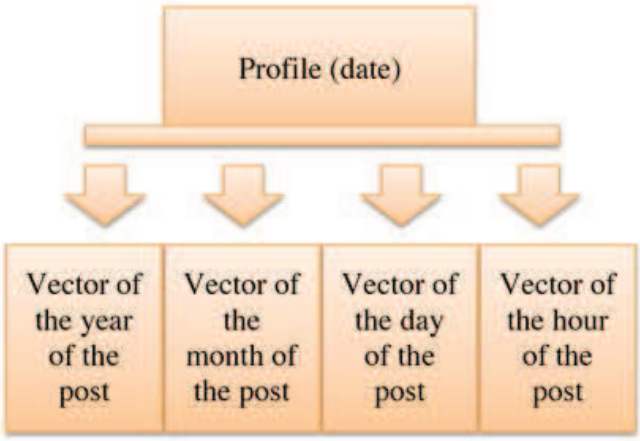



Figure 6.6 – Representation of the profile of a date (with *45 dimensions*) using the binary weighting function. We have *2 dimensions* for years of posts, *12* for months, *7* for days and *24* for hours. It is important to note that the value of each component is between 0 and 1.

but we decided to use the name of days instead since we are interested in finding the impact the day in a week on audience of job ads. We then encode a given date as a set of the vectors of its year, month, day and hour using binary vector model. We have 2 years (2013 and 2014), 12 months in a year, 7 days in a week and 24 hours in a day, so dates are encoded using *45 dimensions*. Example: knowing that *2014-07-21* corresponds to Monday and 11pm to 23 hours, vector(2014-07-21:11pm) = $\{(2014,1),(07,1),(\text{Monday},1),(23,1)\}$.

### Matching vector between a poster's and job's O*NET vectors

To efficiently measure the impact of the matching between a poster's and job's profiles on the performance of the posted job advertisement, we encode the matching vector $v$ as follows:

$$\forall i \in \{0,...,d-1\}\ v_i(v^u, v^j) = \min(v_i^u, v_i^j) \tag{6.8}$$

where $v(v^u, v^j) = (v_0(v^u, v^j),..., v_{d-1}(v^u, v^j))$ is the matching vector between $v^u$ and $v^j$, $d = 1,040$ is the number of distinct O*NET occupations (described in the section 5.5) and $v^u$ and $v^j$ are respectively the O*NET vectors of a user $u$ and job $j$.

*Note that the matching vectors between posters and jobs are only interesting for ads posted on LinkedIn and Facebook users' walls (for the same reasons presented in the section 6.5).*

## 6.6 Work4Oracle

We propose 4 methods to estimate the audience of job advertisements on social networks: CF-Work4Oracle, sPoly-Work4oracle, sRBF-Work4oracle and Lasso-Work4Oracle (described in the next sections).

After modeling job ads and extracting their vectors, we proposed a method inspired from collaborative filtering techniques (described in section 3.2) called *CF-Work4Oracle*. Recall that in collaborative filtering systems, the utility of an item for a user is computed using the utility of similar users for this item. After analyzing our dataset, we notice the date of posts impact their audience, so in CF-Work4Oracle, the audience of a job advertisement is computed as a weighted sum of the audience of similar jobs posted at the same year, month and hour and on the same network as the active job ad. Formally the audience of job ads are computed as follows:

$$\Gamma_n(\rho) = \begin{cases} 0 & \text{if } N_\rho^n \text{ is empty} \\ \dfrac{\sum_{\rho' \in N_\rho^n} \cos(\rho, \rho') \times \Gamma_n(\rho')}{\sum_{\rho' \in N_\rho^n} \cos(\rho, \rho')} & \text{otherwise} \end{cases} \tag{6.9}$$

where $\rho$ and $\rho'$ are vectors of job posts, $\Gamma_n(\bullet)$ is the audience of $\bullet$, $N_\rho^n$ is a set of all job posts similar (similarity $> 0$) to $\rho$ posted on the network $n$ at the same year, month and hour and cos is the cosine similarity defined by Eq. (4.15). We note that the computed $\Gamma_n(\rho)$ is positive since all values of components of job ads' vectors are positive. CF-Work4Oracle could suffer from the limitations of collaborative filtering mainly the scalability issue, we use it as our *baseline method*.

After defining the baseline method, we design and study 3 systems based on regression algorithms: $\epsilon$-SVM-Regression (Polynomial and Radial Basis Function kernels) and Lasso regression (see section 4.3):

1. *sPoly-Work4Oracle* is based on models learnt from our data using $\epsilon$-SVR (see Eq.4.23) with Polynomial kernel (see Eq.4.24b).
2. *sRBF-Work4Oracle*: this job ads' audience estimator is based on models learnt from our data using $\epsilon$-SVR (see Eq.4.23) with RBF kernel (see Eq.4.24c).
3. *Lasso-Work4Oracle* model uses the Lasso regression (see Eq.4.34) to learn models to predict the audience of jobs on social networks.

The numbers of clicks on job ads in our dataset are ranging from 0 to 2086. Our preliminary experiments have showed that it is to difficult to fit a model to accurately predict target values

between 0 and 2086. We decide to use the log-scaled number of clicks (log(1+number of clicks) where log is the natural logarithm) to fit our models, in that case target values are ranging from 0 to 7.64. Since we use log-scaled number of clicks to fit our models, the predictions (integer) are obtained as follows:

$$\hat{y}(x) = \max(0, \lfloor -1 + \exp^{\text{model}(x)} \rceil) \tag{6.10}$$

where $\lfloor \bullet \rceil$ is the nearest integer to $\bullet$ and $model$ is a model learnt from our data using log-scaled number of clicks. In the particular case of a linear model (Lasso regression), the predictions are made as follows:

$$\hat{y}(x) = \max(0, \lfloor -1 + \exp^{\alpha} \prod_{i=1}^{d} \exp^{w_i x_i} \rceil) \tag{6.11}$$

where $\alpha$ and $w$ are the parameters of the learnt linear model, $x$ is the vector for a job ad and $d$ is the number of dimensions of the vectors of job ads. We can note that even though the learnt model is linear, the predictions are not.

Recall that a linear learnt model (see Eq. 6.11) is defined by:

$$\left( \alpha, \ (w_i^f)_{\substack{1 \leq f \leq n_f \\ 1 \leq i \leq k_f}} \right) \tag{6.12}$$

where $\alpha$ is the bias, $w_i^f$ is the learnt weight for the component $i$ of the feature $f$, $n_f$ is the number of different features in the model and $k_f$ is the number of dimensions associated to the feature $f$. We note the target output for a vector $x$ in a linear model defined by Eq. (6.12) is calculated as:

$$\alpha + \sum_{f=1}^{n_f} \sum_{i=1}^{k_f} w_i^f x_i^f \tag{6.13}$$

For each feature, we are interested in its contribution to the audience of job advertisements, we want to answer the following question: how does it contribute to the audience of job ads?

Based on the equation (6.13), we calculate the contribution $c_f$ of a feature $f$ as follows:

$$c_f = \begin{cases} \dfrac{\sum_{i=1}^{k_f} |w_i^f|}{\sum_{i=1}^{k_f} 1_{w_i^f \neq 0}} & \text{if } (\sum_{i=1}^{k_f} 1_{w_i^f \neq 0}) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.14}$$

where $1_{\text{condition}} = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise} \end{cases}$

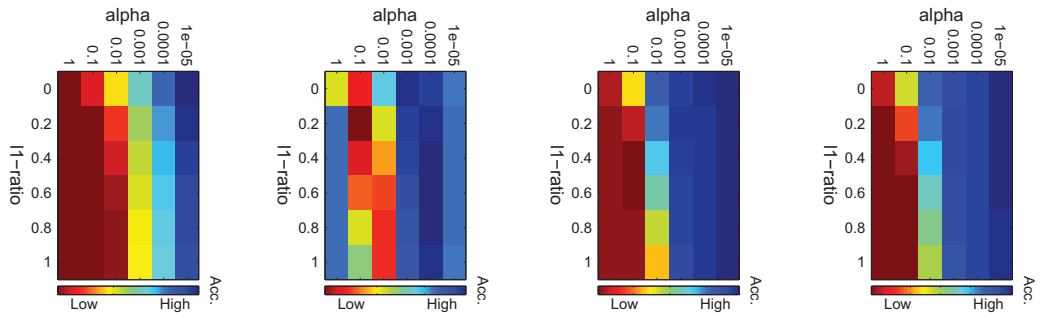
Finally, the percentage $pc_f$ of the contribution $c_f$ of a feature $f$ is obtained by:

$$pc_f = \begin{cases} 100\dfrac{c_f}{\sum_{j=1}^{n_f} c_j} & \text{if } (\sum_{j=1}^{n_f} c_j) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.15}$$

where $c_j$ is the contribution of the feature $j$.

### 6.6.1 Calibration of algorithms and parameter settings

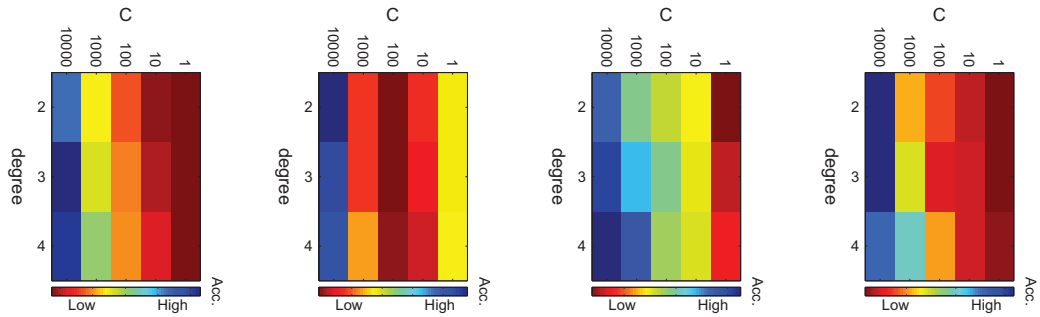
Our scripts are written in Python and are mainly based on scikit-learn[5] [Pedregosa et al., 2011] implementation of different regression algorithms and performance metrics (the implementation in the class Lasso uses coordinate descent as the algorithm to fit models while the implementation of SVR is based on LIBSVM [Chang and Lin, 2011]). The different experiments have been run on Intel Xeon 2.00GHz (with 12 cores). For all proposed methods based on machine learning algorithms (sPoly-Work4oracle, sRBF-Work4oracle and Lasso-Work4Oracle), we optimize the hyper-parameters of used algorithms using Accuracy (*Acc.*) as performance metric, Cross-validation (see section 4.3.4) and Grid search (see section 4.3.4), as shown by Figures 6.7, 6.8, 6.9. For CF-Work4Oracle, we precompute the similarities between job ads in order to speed up our experimentations. We then learn models and test them for different social networks separately using 5-fold cross-validation.




(a) Facebook-profile (b) Facebook-page (c) Twitter (d) LinkedIn

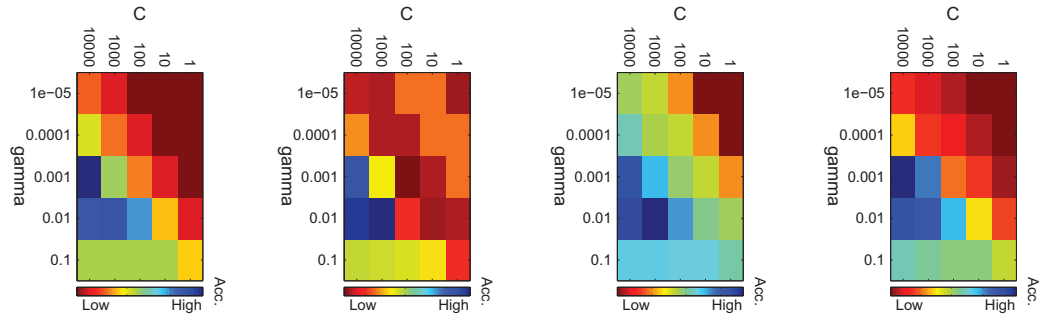
Figure 6.7 – Optimization of hyper-parameters of Elastic-Net using our job advertisements dataset. Note L1-ratio and alpha respectively corresponds to $\rho$ and $\alpha$ in eq (4.35).

[5] http://scikit-learn.org




(a) Facebook-profile (b) Facebook-page (c) Twitter (d) LinkedIn

Figure 6.8 – Optimization of hyper-parameters of Poly SVMs using our job advertisements dataset. Note Coef0 which corresponds to $r$ in eq (4.24b) is set to 1, degree corresponds to $d$ in eq (4.24b) and $C$ to the regularization parameter in eq (4.22).




(a) Facebook-profile (b) Facebook-page (c) Twitter (d) LinkedIn

Figure 6.9 – Optimization of hyper-parameters of RBF SVMs using our job advertisements dataset. Note gamma corresponds to $\gamma$ in eq (4.24c) and $C$ to the regularization parameter in eq (4.22).

### 6.6.2 Factors impacting the audience of job advertisements on Facebook users' walls

Tables 6.2 and 6.3 depict the training times, RMSE, MAE, Accuracy, Precisions, Recalls and F1s of our proposed methods for job ads posted on Facebook users' walls. The models based on Lasso take more time fit models than SVM-based methods. We can note that sPoly-Work4Oracle obtains the lowest RMSE and MAE and the highest accuracy. A fine analysis of the table 6.3 shows that sPoly-Work4Oracle has a higher value of F1 for job ads with at most 1 click (third quintile of the number of clicks) but a very low F1 on job ads with at least 2 clicks. Lasso-Work4Oracle makes a better trade-off between precision and recall. CF-Work4Oracle is globally outperformed by the variants of Work4Oracle based on machine learning.

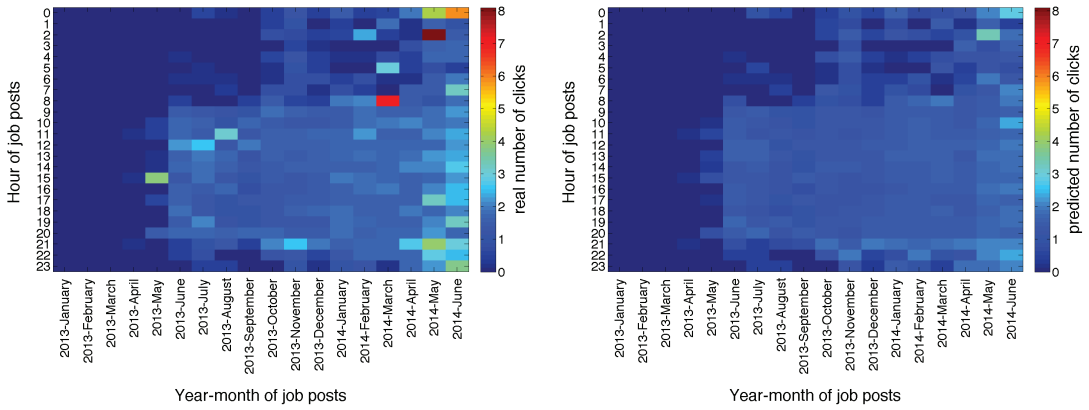
| Methods | Training time (seconds) | RMSE | MAE | Accuracy |
|---|---|---|---|---|
| CF-Work4Oracle | - | 2.47±0.12 | 1.14±0.01 | 0.49±0.01 |
| sPoly-Work4Oracle | 31.65±0.29 | **2.33±0.18** | **0.81±0.03** | **0.66±0.01** |
| sRBF-Work4Oracle | 31.94±0.52 | 2.36±0.16 | 1.15±0.03 | 0.53±0.01 |
| Lasso-Work4Oracle | 1988.54± 7.92 | 2.41±0.40 | 0.93±0.04 | 0.59±0.01 |

Table 6.2 – Training times, RMSE, MAE and Accuracy of Work4Oracle for different regression methods for job advertisements on Facebook users' walls using 5-fold cross-validation.

| | Number of clicks | ≤ 1 (1: 3rd quintile) | ≥ 2 |
|---|---|---|---|
| Precision | CF-Work4Oracle | 0.69±0.01 | 0.33±0.01 |
| | sPoly-Work4Oracle | 0.70±0.01 | 0.45±0.03 |
| | sRBF-Work4Oracle | 0.71±0.01 | 0.35±0.01 |
| | Lasso-Work4Oracle | 0.73±0.01 | 0.39±0.02 |
| Recall | CF-Work4Oracle | 0.45±0.01 | 0.57±0.01 |
| | sPoly-Work4Oracle | 0.90±0.01 | 0.18±0.02 |
| | sRBF-Work4Oracle | 0.51±0.02 | 0.57±0.02 |
| | Lasso-Work4Oracle | 0.62±0.02 | 0.52±0.02 |
| F1 | CF-Work4Oracle | 0.54±0.01 | 0.42±0.01 |
| | sPoly-Work4Oracle | **0.78±0.01** | 0.25±0.02 |
| | sRBF-Work4Oracle | 0.59±0.02 | **0.44±0.01** |
| | Lasso-Work4Oracle | 0.67±0.01 | **0.45±0.02** |

Table 6.3 – Precision, Recall and F1 of Work4Oracle for different regression methods for job advertisements on Facebook users' walls.

Figure 6.10 shows a comparison between the real number of clicks on job advertisements posted on Facebook users' walls and the predicted number of clicks (using Lasso-Work4Oracle). After ensuring that our different models are correctly fit, let us analyze the contributions of




(a) Real intensity of clicks on job ads.

(b) Predicted intensity of clicks on job ads.

Figure 6.10 – Comparison between real and predicted intensities of clicks on job ads posted on Facebook users' walls using 5-fold cross-validation with Lasso-Work4Oracle.

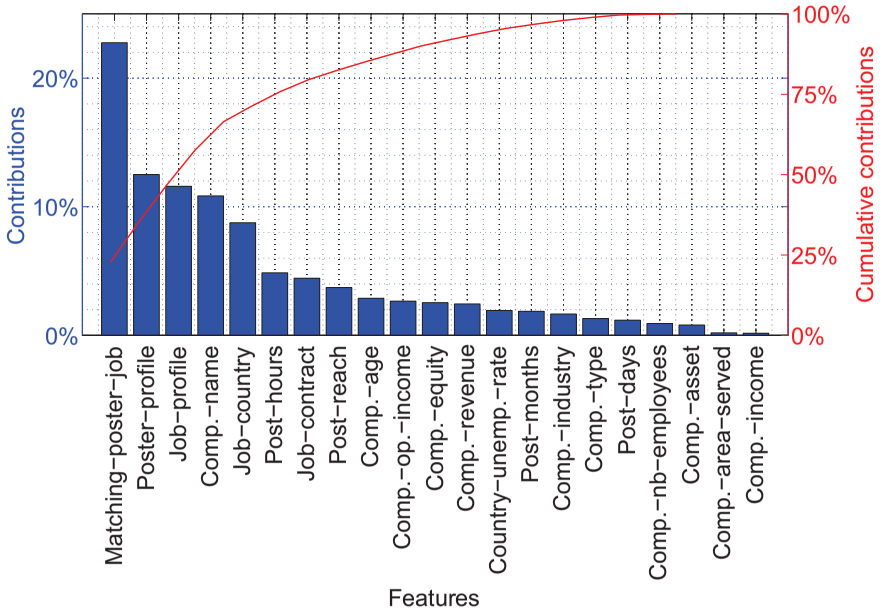



Figure 6.11 – Contributions of different factors (obtained using Lasso-Work4Oracle) to the performance of job ads posted on Facebook users' walls. "Comp.", "unemp.", "op."are respectively the abbreviations of *company*, *unemployment* and *operating*.

different factors. First of all, we obtain a better trade-off between precision and recall with Lasso-Work4oracle, we then compute the contributions of different factors (see Figure 6.11) using the equation (6.15). The analysis of the different contributions of factors reveals that the audience of job advertisements posted on Facebook users' walls depends on different factors, the most important of which are:

- The profiles of posters/matching between the profiles of posters and jobs: from our results, it seems that the performance of a job posted on Facebook users' walls depends on who posted it. Due to privacy concerns we have no information about the connections or relations of the posters but the profiles of posters can give some clues about the profiles of their social connections since users generally make friendships with those who are similar to them.
- The profiles of advertised jobs (type and industry).
- The name of companies of the jobs, this could be extends to the reputation of companies since the two concepts are somehow linked.
- The countries (and their unemployment rates) for which the job ads have been done.
- The hours at which ads have been posted.
- The type of contract of the job ads.

Other interesting factors are the number of persons who can see the advertisements and the age of companies. We also can notice the hour at which a job ad has been posted on Facebook users' walls impacts more than the day and month of the post.

### 6.6.3 Factors impacting the audience of job advertisements on organizations' Facebook pages

We measure the training times, RMSE, MAE, Accuracy, Precisions, Recalls and F1s of our proposed methods for job ads posted on organizations' Facebook pages (see Tables 6.4 and 6.5). We obtain lowest RMSE and MAE and highest accuracy for sRBF-Work4Oracle and Lasso-Work4Oracle; in terms of F1, the two methods obtain the same results. We can note that we have higher F1 scores for job ads posted on organizations' Facebook pages than for those posted on Facebook users' walls (see section 6.6.2).

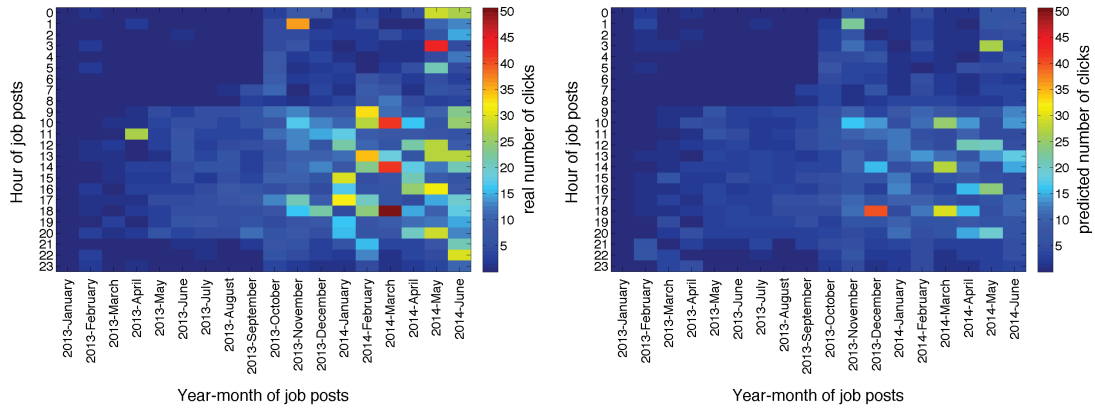
Figure 6.12 compares the real number of clicks on job advertisements posted on organizations' Facebook pages to the predicted number of clicks using sRBF-Work4Oracle, this allows us to make sure that the models have been properly fit. Using the lasso learnt models we estimate the contributions of different factors to the audience of job ads (see Figure 6.13), we find out that the most important factors are:

| Methods | Training time (second) | RMSE | MAE | Accuracy |
|---|---|---|---|---|
| CF-Work4Oracle | - | 50.85±7.09 | 15.50±0.85 | 0.49± 0.01 |
| sPoly-Work4Oracle | 63.39±1.32 | 39.71±6.96 | 8.76±0.42 | 0.72±0.02 |
| sRBF-Work4Oracle | 94.70±1.46 | 38.91±8.55 | **7.84±0.66** | **0.78±0.01** |
| Lasso-Work4Oracle | 2,696.29±13.52 | **37.48±7.54** | **7.87±0.55** | **0.78±0.01** |

Table 6.4 – Training times, RMSE, MAE and Accuracy of Work4Oracle for different regression methods for job advertisements on organizations' Facebook pages using 5-fold cross-validation.

| | Number of clicks | ≤ 4 (4: 3rd quintile) | ≥ 5 |
|---|---|---|---|
| Precision | CF-Work4Oracle | 0.73±0.01 | 0.40±0.00 |
| | sPoly-Work4Oracle | 0.78±0.02 | 0.62±0.03 |
| | sRBF-Work4Oracle | 0.81±0.01 | 0.72±0.02 |
| | Lasso-Work4Oracle | 0.82±0.01 | 0.70±0.01 |
| Recall | CF-Work4Oracle | 0.33±0.01 | 0.79±0.00 |
| | sPoly-Work4Oracle | 0.79±0.03 | 0.60±0.02 |
| | sRBF-Work4Oracle | 0.86±0.01 | 0.65±0.01 |
| | Lasso-Work4Oracle | 0.84±0.01 | 0.68±0.01 |
| F1 | CF-Work4Oracle | 0.45±0.01 | 0.53±0.00 |
| | sPoly-Work4Oracle | 0.78±0.02 | 0.61±0.02 |
| | sRBF-Work4Oracle | **0.84±0.01** | **0.68±0.01** |
| | Lasso-Work4Oracle | **0.83±0.00** | **0.69±0.02** |

Table 6.5 – Precision and Recall of Work4Oracle for different regression methods for job advertisements on organizations' Facebook pages.




(a) Real intensity of clicks on job ads.

(b) Predicted intensity of clicks on job ads.

Figure 6.12 – Comparison between real and predicted intensity of clicks on job ads posted on organizations' Facebook pages using sRBF-Work4Oracle.

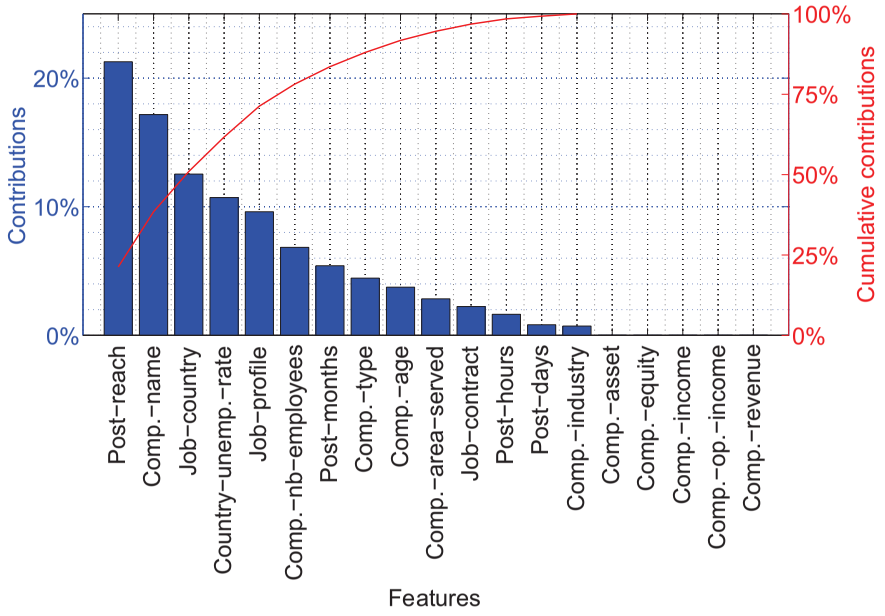



Figure 6.13 – Contributions of different factors (obtained using Lasso-Work4Oracle) to the performance of job ads posted on organizations' Facebook pages. "Comp.", "unemp.", "op."are respectively the abbreviations of *company*, *unemployment* and *operating*.

- The number of persons who can see the job advertisements (reach of posts): the reach of job ads is the most important factors that impact the audience of job ads on organizations' Facebook pages. Only persons who liked an organization's Facebook page can see its job ads, so we can conclude that the popularity of organizations on Facebook affects the audience of its job ads.
- The name/reputation of organizations of the jobs.
- The countries (and their unemployment rates) for which the jobs have been advertised.
- The profiles of advertised jobs.

Other impacting factors are the age of companies, the size of companies (number of employees), the months, hours and days of posts and the contract types of jobs. We can see that the hours of job posts impact more its audience than the days at which it has been posted.

### 6.6.4 Factors impacting the audience of job advertisements on LinkedIn

The analysis of the training times shows that the learning times of SVM-based models are much more shorter than those of Lasso-based models (see Table 6.6). This table also shows the RMSE, MAE and Accuracy scores for our proposed methods: Lasso-Work4Oracle outperforms the others for these 3 metrics as well as for the Precisions, Recalls and F1s metrics (see Table 6.7).

| Methods | Training time (second) | RMSE | MAE | Accuracy |
|---|---|---|---|---|
| CF-Work4Oracle | - | 6.54±1.12 | 3.29±0.05 | 0.44±0.00 |
| sPoly-Work4Oracle | 941.28±40.45 | 6.68±1.09 | 3.28±0.09 | 0.58±0.02 |
| sRBF-Work4Oracle | 1,089.50±29.22 | 6.13±1.18 | 2.74±0.06 | 0.61±0.01 |
| Lasso-Work4Oracle | 11,646.07± 45.33 | **5.97±1.21** | **2.48±0.04** | **0.67± 0.01** |

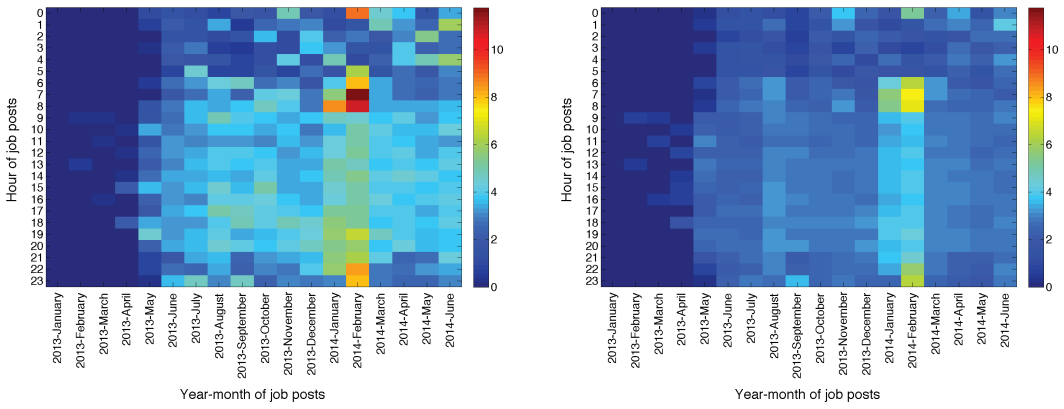
Table 6.6 – Training times, RMSE, MAE and Accuracy of Work4Oracle for different regression methods for job advertisements on LinkedIn using 5-fold cross-validation.

Figure 6.14 compares the real number of clicks on job advertisements posted on LinkedIn to the predicted number of clicks using Lasso-Work4Oracle. The analysis of this Figure ensures us that the different models are well fit, we then calculate the contributions of different factors (see Figure 6.15). We find out the following important factors:

- The profiles of posters/matching between the profiles of posters and jobs.
- The name/reputation of organizations of the jobs.
- The number of persons who can see the job advertisements (reach of posts).

| | Number of clicks | $\leq 2$ (2: 2nd quintile) | $\geq 3$ |
|---|---|---|---|
| Precision | CF-Work4Oracle | 0.68±0.01 | 0.43±0.00 |
| | sPoly-Work4Oracle | 0.66±0.01 | 0.50±0.02 |
| | sRBF-Work4Oracle | 0.67± 0.00 | 0.54 ± 0.02 |
| | Lasso-Work4Oracle | 0.73 ±0.00 | 0.59 ±0.01 |
| Recall | CF-Work4Oracle | 0.08±0.00 | 0.95±0.00 |
| | sPoly-Work4Oracle | 0.59±0.04 | 0.58±0.02 |
| | sRBF-Work4Oracle | 0.65± 0.03 | 0.55± 0.02 |
| | Lasso-Work4Oracle | 0.66±0.01 | 0.67±0.00 |
| F1 | CF-Work4Oracle | 0.14±0.01 | 0.59±0.00 |
| | sPoly-Work4Oracle | 0.62±0.03 | 0.54±0.01 |
| | sRBF-Work4Oracle | 0.66±0.01 | 0.55±0.01 |
| | Lasso-Work4Oracle | **0.70±0.01** | **0.63±0.01** |

Table 6.7 – Precision and Recall of Work4Oracle for different regression methods for job advertisements on LinkedIn.




(a) Real intensity of clicks on job ads.

(b) Predicted intensity of clicks on job ads.

Figure 6.14 – Comparison between real and predicted intensity of clicks on job ads posted on LinkedIn using Lasso-Work4Oracle.

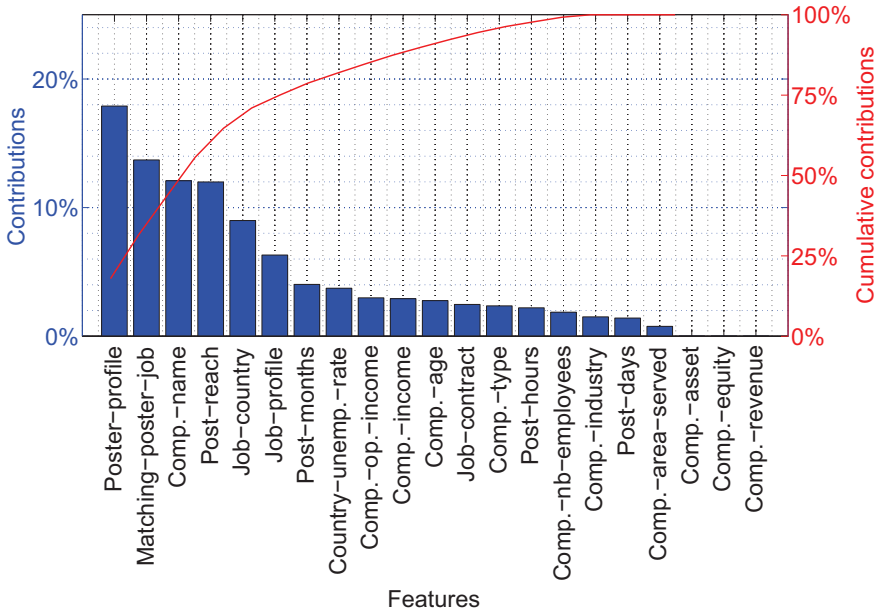



Figure 6.15 – Contributions of different factors (obtained using Lasso-Work4Oracle) to the performance of job ads posted on LinkedIn. "Comp.", "unemp.", "op."are respectively the abbreviations of *company*, *unemployment* and *operating*.

- The countries (and their unemployment rates) for which the jobs have been advertised.
- The profile of advertised jobs.

Other important factors are the months, hours and days of posts, the age of companies, the contract type of jobs and the size of organizations (number of employees).

### 6.6.5 Factors impacting the audience of job advertisements on Twitter

Fitting models to predict the audience of job ads posted on Twitter using Lasso-Work4Oracle took much more time than SVM-based systems (see Table 6.8): we have the same observations for job ads on Facebook and LinkedIn (see sections 6.6.2, 6.6.3, 6.6.4). We note an out-performance of Lasso-Work4Oracle and sRBF-Work4Oracle on Twitter in terms of RMSE, MAE, Accuracy and F1s (see Tables 6.8 and 6.9).

Figure 6.16 compares the real number of clicks on job advertisements posted on Twitter to the predicted number of clicks using Lasso-Work4Oracle while Figure 6.17 shows the different factors impacting the audience of job advertisements posted to Twitter. Among the most important factors, we find out:

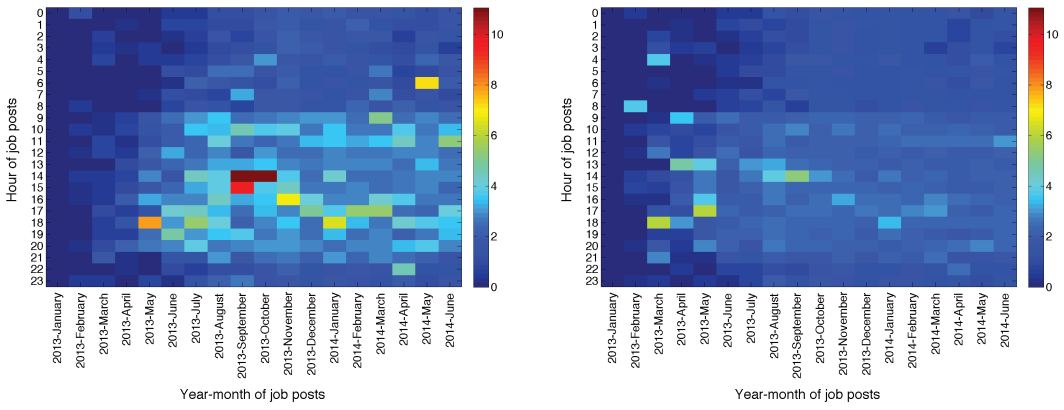
- The name/reputation of organizations of jobs.

| Methods | Training time (second) | RMSE | MAE | Accuracy |
|---|---|---|---|---|
| CF-Work4Oracle | - | 8.52±0.93 | 2.93±0.11 | 0.49±0.01 |
| sPoly-Work4Oracle | 305.79±6.68 | 7.59±0.90 | 2.40±0.14 | 0.56±0.01 |
| sRBF-Work4Oracle | 446.93±2.32 | 7.77±0.93 | **2.00±0.11** | **0.60±0.01** |
| Lasso-Work4Oracle | 6,214.61±33.26 | **7.54 ±0.95** | **2.02±0.10** | **0.59±0.00** |

Table 6.8 – Training times, RMSE, MAE and Accuracy of Work4Oracle for different regression methods for job advertisements on Twitter using 5-fold cross-validation.

| | Number of clicks | ≤ 1 (1: 2nd quintile) | ≥ 2 |
|---|---|---|---|
| Precision | CF-Work4Oracle | 0.67±0.01 | 0.47±0.01 |
| | sPoly-Work4Oracle | 0.62±0.02 | 0.52±0.01 |
| | sRBF-Work4Oracle | 0.67±0.02 | 0.55±0.00 |
| | Lasso-Work4Oracle | 0.70±0.01 | 0.54±0.01 |
| Recall | CF-Work4Oracle | 0.09±0.00 | 0.95±0.00 |
| | sPoly-Work4Oracle | 0.44±0.02 | 0.69±0.01 |
| | sRBF-Work4Oracle | 0.49±0.01 | 0.73±0.01 |
| | Lasso-Work4Oracle | 0.40±0.01 | 0.80± 0.01 |
| F1 | CF-Work4Oracle | 0.16±0.00 | 0.63±0.01 |
| | sPoly-Work4Oracle | 0.51±0.02 | 0.59±0.01 |
| | sRBF-Work4Oracle | **0.57±0.01** | 0.63±0.01 |
| | Lasso-Work4Oracle | 0.51±0.00 | **0.65±0.00** |

Table 6.9 – Precision and Recall of Work4Oracle for different regression methods for job advertisements on Twitter.




(a) Real intensity of clicks on job ads

(b) Predicted intensity of clicks ads

Figure 6.16 – Comparison between real and predicted intensity of clicks on job ads posted on Twitter using Lasso-Work4Oracle.

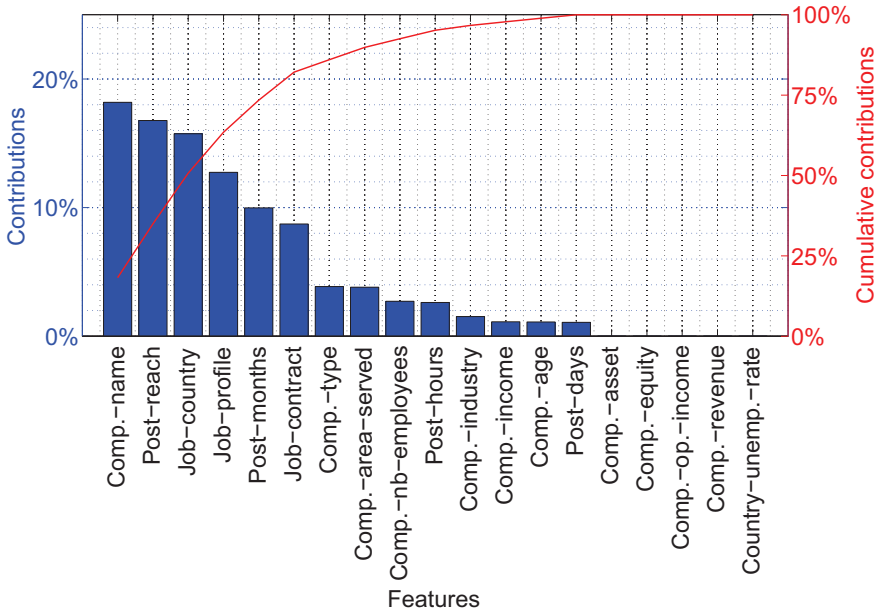



Figure 6.17 – Contributions of different factors (obtained using Lasso-Work4Oracle) to the performance of job ads posted on Twitter. "Comp.", "unemp.", "op."are respectively the abbreviations of *company*, *unemployment* and *operating*.

- The number of persons who can see the job advertisements.
- The countries for which the jobs have been advertised.
- The profiles of jobs posted on Twitter.
- The months of posts.
- The contract type of jobs.

## 6.7 Conclusion

In this chapter, we applied data mining, recommender systems and machine learning techniques to a significant large datasets from real-world data collected by the Work4 to find out and quantify the factors impacting the audience of job advertisements on Facebook, LinkedIn and Twitter. We combined heterogeneous data from Work4, Wikipedia and OECD websites and defined a list of features that could be important in the task estimating the attractiveness of organizations for applicants and use them to propose a vector model for job ads based on the taxonomy O*NET (see appendix B) and the vector space model with the binary weighting function and making sure that the values of all components are ranging from 0 to 1. Our proposed models for job ads can be easily enrich with additional information like meta-data about organizations, social networks, friends of posters and countries.

We used linear and non-linear regression algorithms to learn models that estimate the log of audience: real predictions are the exponential of the predictions made by the models (this allows us to always make non-linear predictions). The use of linear methods on our models for job ads allows us to quantify the factors impacting the audience of job advertisements on Facebook, Linkedin and Twitter. We can use the obtained results to select interesting features and then speed up our future computations and optimize the performance of job ads on social networks.

Our results show that the performance of job advertisements posted on social networks depend on some hidden factors, the most important are the profiles of posters, the number of persons who can see the ads, the name of the companies of jobs (which give a clue about their reputations), the size of job companies, the countries (and their unemployment rates) for which the job ads have been posted, the profiles of jobs (their functions, industries and categories) and the months, hours and days at which job ads have been posted. We find out that the months of posts generally impact more than the performance of job ads than the hours of posts which have a higher impact than the days at which job ads have been posted. These results confirm the findings of [Chapman et al., 2005; Ehrhart and Ziegert, 2005; Lievens and Highhouse, 2003; Mathews and Redman, 1998; Turban et al., 1998] who support that location, salary and description of jobs, the reputation, size and image as employer of the organizations may impact their attractiveness for applicants: our results extend these previous work to the social media-based recruitments.

The number of persons who can see the job ads (reach of posts) is the most important factor that impacts the audience of job ads on organizations' Facebook pages. Only persons who liked an organization page can see its job ads, so we can conclude that the popularity of an organization on Facebook affects the audience of its job ads. The notion of popularity of an organization may be closely linked to its reputation: an organization with a good reputation can be popular for future applicants. Our experiments show that the hour of posts impact more for job ads posted on the walls of Facebook users than for jobs posted on organizations' Facebook pages. This could be explained by the fact that the half-life of job posts on the walls of Facebook users is very short (see Figure 6.3). We find out that the months of posts generally impact more the performance of job ads than the hours of posts which have a higher impact than the days at which job ads have been posted. We noticed that the most important factors impacting the audience of job ads are almost the same for the three studied social networks, the only difference being the contributions of factors which vary over social networks.

Our results reveals that it is much more easier to accurately predict the performance of job advertisements posted on organizations' Facebook pages and LinkedIn (see Tables 6.5 and 6.7) than those posted on Facebook users' walls (see Tables 6.3 and Twitter 6.9). The results of this study allow us to explain why some job advertisements posted by our customers perform better than others. We used a simplified version of Work4Oracle based on Decision Trees [Quinlan, 1986] to extracted insights for the product team of the company Work4.

# 7 Conclusion and Future Directions

*"The stairs of science are like Jacob's ladder, they only end at the feet of God."*

- Albert Einstein, Nobel Prize in Physics (1921)

## Contents



When I started this thesis three years ago, I was looking for something, an adventure during which I hoped to learn more, understand how to analyze social networks, to develop systems to automatically recommend jobs to social network users, to develop predictive systems based on machine learning and artificial intelligence and to learn about startups. Today, concluding this thesis, I would say it has been a long and difficult journey but I have learnt a lot.

For a reminder, this thesis has been done in an industrial context (a collaboration between the laboratory L2TI and the company Work4) which defines the methodologies used during this thesis: most of the studied topics in this document have been proposed by Work4 and validated on its collected data. During the 3 years of this thesis, we developed several systems and explored many topics and algorithms to adapt our studies to the changes in the company business model.

The next two sections summarize the global conclusions we have drawn from our experiments on recommending jobs to social network users and predicting the audience of job advertisements posted on social networks.

## 7.1 Job recommendation to Facebook and LinkedIn users

Recommender systems have emerged as an independent research area in the mid-1990s and for the last two decades, a great research effort has been made to improve the quality of recommended items. In this thesis, we developed job recommender systems for social network users. Our studies have been done in an industrial context, as a result, the proposed systems have been tested and validated on real-world data collected by the company Work4.

To design our systems, we needed to study the literature of recommender systems to understand how they work and to identify the strengths and weaknesses of different categories of recommendation techniques. We constantly surveyed the literature to keep up-to-date to the latest algorithms of recommender systems, data mining and social network analysis throughout the period of this thesis.

The analysis of the datasets about job recommendation provided by Work4 has revealed that to make efficient job recommendation to social network users, one needs to *deal with missing and noisy data*:

- Social network users can publish fake information about them and they do use informal vocabulary like abbreviations and teen text terms.
- Social network users do not completely fill their profiles and are more and more reluctant to let third party applications having access their profile data.
- Social networks like Facebook are increasingly restricting the access to the profiles of their users by constantly changing their privacy settings.
- The vocabulary of job descriptions is formal but varies a lot from a company to another, making the task of mining job descriptions very difficult.
- Some job descriptions are poorly written and difficult to understand, even for a human.
- Some job descriptions contain a section describing the related companies (their products and their customers), this section is interesting for applicants and candidates but it generally brings noise when automatically mining job descriptions.

In this thesis, we proposed 2 families of methods to recommend jobs to social network users: algorithms based on the bag-of-words model and those based on O*NET taxonomy.

For the recommendation algorithms based on the bag-of-words model, we showed by comparing different weighting functions with different similarity measures that a TF-IDF weighting function combined with cosine similarity yields better results than the others in the task of job recommendation. However the results with TF-IDF were disappointing when recommending jobs to Facebook users. The use of lemmatisation as preprocessing did not significantly improve the performance of our systems but has reduced the dimensionality of the problem.

The profiles of social network users and the description of jobs are composed of different fields, we estimated the importance of these fields in the task of job recommendation. This allows us to figure out the difference between the vocabulary (set of terms) of Facebook/LinkedIn users and the vocabulary of job descriptions (in our datasets). LinkedIn users' vocabulary seems closer to the vocabulary of jobs than that of Facebook users. Not surprisingly, our study has revealed that the most relevant fields for users are "Work history" (Facebook) and "Headline and positions" (LinkedIn), the "Title" brings the most important information for jobs. These results make sense since if one tries to tell whether a given Facebook or LinkedIn user matches with a job, he will probably first compare the user's work history (Facebook) or headline/positions (LinkedIn) field information to the title of the job description.

We showed that the use of machine learning algorithms (especially SVMs) to learn models to recommend jobs to social network users significantly improves the quality of recommendations. However, it involves the collection of a dataset containing a significant number of positive instances (users matching jobs) and negative instances (users who do not match jobs) for different families and categories of jobs.

Statistics from our datasets (see Table 5.2) also showed that a vast majority of Facebook users fields are almost empty. This has raised a big problem: we cannot accurately make recommendations to users whose profiles are almost empty using the proposed recommender systems. This has led us to enrich the profiles of users with data from their friends: this is known as social recommendation. Due to privacy concern, we only have access to partial data of friends, as a result, we have only used basic methods of social recommendation. Unfortunately the use of these basic methods of social recommendation failed to improve our results. However we showed that the use of relevance feedback drastically improves the quality of job recommendation, this is very interesting and shows that we can improve the performance of heuristic-based job recommender systems by enriching social network users' profiles using their feedback.

Social network users generally do not give feedback about the quality of job recommendation (made to them), this leads us to use knowledge databases like O*NET taxonomy to deal with missing data in the task of job recommendation.

For the recommendation algorithms based on O*NET taxonomy, we showed the cosine similarity yields results slightly better than both our proposed fuzzy logic-based similarity functions and Pearson Correlation Coefficient. The use of our taxonomy-based vector model has dramatically reduced the difference of quality between our Facebook and LinkedIn data in the task of job recommendation. Our study has also revealed that the use machine learning (SVMs) improves the quality of job recommendation compared to the heuristic-based recommender systems.

Since we know that matching a user with a job is a multidimensional problem (see section 1.1), we proposed a multilayer vector model combining heterogeneous data from Facebook users, LinkedIn users and job descriptions, based on the taxonomy O*NET, the related experiments

revealed that the direct matching between users' and jobs' country, language, TF-IDF, O*NET and O*NET related vectors gives results that can be improved by combining them into a model learnt using SVMs.

During this thesis, we proposed, designed and tested various job recommender systems using complex, semi-structured data from Facebook/LinkedIn users and the description of jobs posted by the customers of the company Work4. Several of our proposed recommender systems outperforms two state-of-the-art methods of recommender systems (CTR and MF) (see section 5.6). We showed that the quality of job recommendation to social network users using Facebook data is lower than using LinkedIn data. This can be explained by the fact that contrary to Facebook, LinkedIn users use this platform for a professional purpose (as a result, they publish correct and detailed information about their work and education histories).

After studying the literature of recommender systems, developing several job recommender systems for social network users and analyzing their results, we draw the following key conclusions:

- Recommender systems using the bag-of-words model combined with similarity functions have poor performance when recommending jobs to Facebook users but yield a fair performance for LinkedIn users. The performance for Facebook users can be improved using knowledge databases such as taxonomies (O*NET taxonomy for instance).
- Recommender systems using either the bag-of-words model or a taxonomy-based vector model combined with machine learning-based models have high performance when recommending jobs to both Facebook and LinkedIn users.
- Recommender systems using the bag-of-words model combined with similarity functions and the relevance feedback mechanism have high performance when recommending jobs to both Facebook and LinkedIn users.
- The use of data of users' social connections can mitigate the problem of missing data (an issue on social networks like Facebook) and therefore improve the performance of job recommender systems but the data of social connections are not accessible most of the time due to the privacy preservation.

## 7.2 Prediction of the audience of job ads posted on social networks

During the second part of this thesis, we tackled the problem of estimating the audience of job advertisements posted on social networks. To develop our different systems, we first studied the literature of recommender systems, data mining, machine learning and human resource management. Previous work in the human resource management found out some important factors impacting the attractiveness of organizations for applicants but these studies were based on small sets of data and their conclusions are generally based on subjective analysis which makes it difficult to generalize.

We applied data mining, recommender systems and machine learning techniques to a significant large dataset from real-world data collected by the company Work4 to find out and quantify the factors impacting the audience of job advertisements on 3 popular social networks (Facebook, LinkedIn and Twitter). Our studies revealed that it is possible to learn from our data models that are able to accurately predict the audience of job ads posted on social networks (see sections 6.6.2, 6.6.3, 6.6.4, 6.6.5). However, they showed that it is considerably easier to accurately predict the performance of job advertisements posted on organizations' Facebook pages and LinkedIn than those posted on Facebook users' walls and Twitter. *Note that additional studies can be done to improve the prediction quality of job advertisements posted on Facebook users' walls and Twitter.*

We showed that the performance of job advertisements posted on social networks depend on some hidden factors, particularly the profiles of posters, the number of persons who can the ads[1], the name of the companies of jobs (which give a clue about their reputations), the size of job companies, the countries (and their unemployment rates) for which the job ads have been posted, the profiles of jobs (their functions, industries and categories) and the months, hours and days when job ads have been posted. We find out that the months of posts generally contribute more to the performance of job ads than the hours of the posts. These results confirm the findings of [Chapman et al., 2005; Ehrhart and Ziegert, 2005; Lievens and Highhouse, 2003; Mathews and Redman, 1998; Turban et al., 1998] who claim that location, salary and description of jobs, the reputation, size and image as employer of the organizations contribute to the attractiveness of their job offers: our results extend these previous work to the social media-based recruitments. We noticed that the most important factors contributing to the audience of job ads are almost the same for the three social networks (though, the contributions of factors may vary over social networks).

The systems we developed in this thesis to estimate the number of clicks on job advertisements posted on social networks can be easily extended to predict the popularity of any type of advertisements on social networks. The proposed solutions can be used in on-line systems since their models can be fit off-line and the prediction step is very fast (once the models are available).

## 7.3 Future Directions

We have explored various topics and we hope to have improved the knowledge about the process of recommending and advertising jobs to social network users but many challenges remain unexplored.

First of all, we did not consider the temporal aspect of job recommendation in this study because we did not have the related data. We assumed static preferences of users for jobs, in other words, if a job matches a user today, the same job will match the same user in the future

[1] On Facebook page, this represents the number of persons who liked the Facebook page of the company that posts the job ad, so in that case, this is gives a clue to the popularity of the company (at least on Facebook).

(in ten years for instance). This assumption is not entirely justified but we have to make it due to the nature of our data: we only have data about static preferences of users for jobs. It would be interesting to extend our methods to take into account the temporal aspect of job recommendation/advertising to social network users.

We focused our studies on content-based recommendation techniques since we had not enough jobs linked to social network users: it has been very tedious for us to obtain the information about social network users' interests for jobs. Almost all our datasets have been collected based manual annotations. Companies like LinkedIn can automatically know the interests of users for jobs (jobs they explore, click on or apply to): using those data, our work can be extended to collaborative filtering systems and hybrid recommendation which could lead to much more efficient job recommender systems.

Due to privacy concerns, we only worked with partial views of social network users' profiles (Facebook, LinkedIn and Twitter) by accessing information in only some of the fields of their profiles. Companies like Facebook, LinkedIn and Twitter can access the full profiles of users, it would very interesting and exciting to work with full profiles to make much more accurate job recommendations/advertisings to users. Privacy concerns have also limited us in the use of data of users' friends/social connections on social networks to enrich their profiles in order to retrieve missing information in users' profiles to make much more accurate job recommendation. Additional studies can extend our work by using the data of Facebook, LinkedIn and Twitter full profiles to make job recommendation.

Some social network users publish fake information in their profiles, and this makes it difficult to recommend jobs to them. Thus, detecting fake information in social network users' profiles could be very interesting to improve the performance of the recommender systems.

We presented in this section a list of some future directions of our work, this list is not exhaustive, there are many additional studies that can be performed to extend our work.

# A Publications

## Journals, Conferences and Workshops

1. **Mamadou Diaby** and Emmanuel Viennet. Quantifying the hidden factors impacting the audience of job advertisements posted on Facebook. In Proceedings of the 15$^{th}$ Industrial Conference on Data Mining ICDM 2015.

2. **Mamadou Diaby** and Emmanuel Viennet. Job recommendations on social networks using a multilayer vector model. *Workshop on Heterogeneous Information Access* - WSDM 2015.

3. Daniel Bernardes, **Mamadou Diaby**, Raphaël Fournier, Françoise Fogelman Soulié and Emmanuel Viennet. A Social Formalism & Survey for Recommender Systems. SIGKDD Explorations, vol. 16, no. 2, Dec. 2014.

4. **Mamadou Diaby**, Emmanuel Viennet, and Tristan Launay. Exploration of methodologies to improve job recommender systems on social networks. Social Network Analysis and Mining Journal, 2014.

5. Emmanuel Malherbe, **Mamadou Diaby**, Mario Cataldi, Emmanuel Viennet, and Marie-Aude Aufaure. Field selection for job categorization and recommendation to social network users. In IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'14), pages 588-595, August 2014.

6. **Mamadou Diaby** and Emmanuel Viennet. Taxonomy-based job recommender systems in facebook and linkedin. In Proceedings of the 2014 IEEE Eighth International Conference on Research Challenges in Information Science RCIS 2014, pages 237-244. IEEE, may 2014.

7. **Mamadou Diaby** and Emmanuel Viennet. Developpement d'une application de recommandation d'offres d'emploi aux utilisateurs de Facebook et Linkedin. In Atelier Fouille de Données Complexes de la 14e Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC'14), Rennes, Jan 2014.

8. **Mamadou Diaby**, Emmanuel Viennet, and Tristan Launay. 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, pages 821-828, Aug 2013.

## Submissions under review

1. **Mamadou Diaby** and Emmanuel Viennet. Quantifying the hidden factors impacting the audience of job advertisements posted on Facebook, LinkedIn and Twitter.

2. Jean Thierry Stephen Avocanh and **Mamadou Diaby**. Learning methods for Semi-Persistent Scheduling in the uplink of LTE networks. To be submitted to IEEE Transactions on Wireless Communications, 2015.

# B Description of O*NET Databases

After exploring the bag-of-words models to recommend jobs to social network users, we needed to study how ontology-based models for users and jobs can be interesting in the task of job recommendation to social network users. Since the company (Work4) was focused on the US market, we used O*NET-SOC (Occupational information NETwork-Standard Occupational Classification) taxonomy [National Center for O*NET Development, 2010; O*NET, 2015; Peterson et al., 2001]. O*NET-SOC is a taxonomy that defines the set of occupations across the world of work, it is being developed under the sponsorship of the US Department of Labor/Employment and Training Administration (USDOL/ETA) through a grant to the North Carolina Department of Commerce[1].

O*NET-SOC taxonomy has been initially released in 1998 (O*NET 98) [O*NET, 2015]. We used for our experiments the version 18.0 of O*NET-SOC taxonomy (released in July 2013) [National Center for O*NET Development, 2013], which contains 1,110 occupational titles, 974 of which represent O*NET data-level occupations (those data have been collected from job incumbents, occupation experts and occupational analysts); we focused on 1,040 occupations for our studies. Figure B.1 shows the structure of the current version of O*NET-SOC taxonomy.

O*NET-SOC taxonomy is based on SOC (Standard Occupational Classification)[2] in which we have four levels of aggregation [National Center for O*NET Development, 2010; O*NET, 2015]:

- 23 major groups.
- 97 minor groups.
- 461 broad occupations.
- and 840 detailed occupations.

SOC occupations are encoded with 6 digits: $d_1d_2$-$d_3d_4d_5d_6$ where $d_1d_2$, $d_3$, $d_4d_5$ and $d_6$

[1] http://www.onetcenter.org/about.html
[2] http://www.bls.gov/soc/

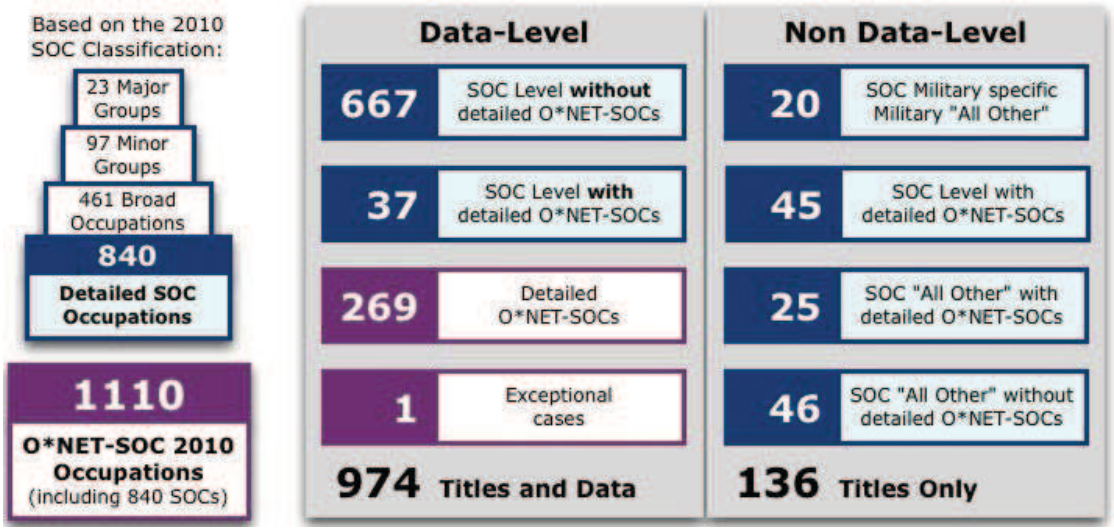



Figure B.1 – Structure of O*NET-SOC taxonomy extracted from [O*NET, 2015].

respectively represent the major group, minor group, broad occupation and detailed occupation [National Center for O*NET Development, 2010]. The latest version of SOC (released in 2010) contains 23 major groups:

1. **11-0000**: Management Occupations.
2. **13-0000**: Business and Financial Operations Occupations.
3. **15-0000**: Computer and Mathematical Occupations.
4. **17-0000**: Architecture and Engineering Occupations.
5. **19-0000**: Life, Physical, and Social Science Occupations.
6. **21-0000**: Community and Social Services Occupations.
7. **23-0000**: Legal Occupations.
8. **25-0000**: Education, Training, and Library Occupations.
9. **27-0000**: Arts, Design, Entertainment, Sports, and Media Occupations.
10. **29-0000**: Healthcare Practitioners and Technical Occupations.
11. **31-0000**: Healthcare Support Occupations.
12. **33-0000**: Protective Service Occupations.
13. **35-0000**: Food Preparation and Serving Related Occupations.
14. **37-0000**: Building and Grounds Cleaning and Maintenance Occupations.

15. **39-0000**: Personal Care and Service Occupations.
16. **41-0000**: Sales and Related Occupations.
17. **43-0000**: Office and Administrative Support Occupations.
18. **45-0000**: Farming, Fishing, and Forestry Occupations.
19. **47-0000**: Construction and Extraction Occupations.
20. **49-0000**: Installation, Maintenance, and Repair Occupations.
21. **51-0000**: Production Occupations.
22. **53-0000**: Transportation and Material Moving Occupations.
23. **55-0000**: Military Specific Occupations.

Since O*NET-SOC taxonomy is based on Standard Occupational Classification (SOC), its occupations are encoded as SOC occupations but they have a 2 digit extension $.e_1e_2$ ($d_1d_2$-$d_3d_4d_5d_6.e_1e_2$). The extension represents the different variants of SOC detailed occupation used in O*NET. Example: the O*NET occupations *19-4051.01* and *19-4051.02* are 2 variants of the SOC occupation *19-4051* (Nuclear technicians). O*NET contains the same 23 major groups as SOC. O*NET occupations (version 18.0 of O*NET databases) are stored in the table/collection **Occupation Data** (which contains 3 fields: title, O*NET-SOC code and the description associated to the occupation). The table/collection **Occupation Level Metadata** allows a better understanding of **Occupation Data**.

The version 18.0 of O*NET-SOC taxonomy contains other tables/collections, the most interesting of which are:

- **Abilities** contains the abilities required for different O*NET occupations. Examples: "Oral comprehension", "Written comprehension"and "Oral expression".
- **Education, Training, and Experience Categories** and **Education, Training, and Experience** contain the education level, training and experience required for occupations.
- **Interests**: contains the Interest data associated with each O*NET-SOC occupation; "Realistic", "Artistic", "Enterprising" are some examples of interest data.
- **Job Zone Reference** and **Job Zones**: contains information about the "zone"of occupations; we have 5 job zones: from 1 (usually require a high school diploma) to 5 (require graduate school: Master, Ph.D, etc.).
- **Knowledge**: provides a mapping of O*NET-SOC occupations to Knowledge ratings. "Law and Government", "Telecommunications", "Communications and Media" are some examples of Knowledge.

- **Skills**: contains the skills required for an O*NET occupation. Examples: "Active Listening", "Speaking", "Writing".
- **Task Categories**, **Task Ratings** and **Task Statements** provide a description of Task categories and task statements.
- **Work Activities**: contains the Work Activity data associated with each O*NET-SOC occupation. Examples: "Getting Information", "Monitor Processes, Materials, or Surroundings", "Judging the Qualities of Things, Services, or People".
- **Work Context**: contains the Work Context data associated with each O*NET-SOC occupation. **Work Context Categories**: contains the categories associated with the Work Context content area.
- **Work Styles** and **Work Values** respectively contain the Work Styles data and Work Values data associated with each O*NET-SOC occupation. "Persistence", "Initiative", "Leadership" are some of examples of Work Styles while Work Values can be "Achievement", "Relationships", "Independence".

It is important to note that O*NET-SOC taxonomy is continuously being improved by adapting it to the changes in the world of Work [National Center for O*NET Development, 2010; O*NET, 2015].

As conclusion, this section briefly presented the O*NET-SOC taxonomy, for further explanations and details, please refer to [National Center for O*NET Development, 2010, 2013; O*NET, 2015; Peterson et al., 2001].

# C Additional Explorations

As this thesis has been done in an industrial context, we did many other explorations, the topics of which have been given up by the company Work4 (because of a change in the its global strategy/business model) before obtaining interesting results or conclusions. Among them, we present the job categorization and job summarization problems in this section.

## C.1 Job categorization

The goal of this exploration was to be able to better categorize jobs that our clients post using our platform. The description of our jobs contain textual information and are divided into 3 fields as presented in the section 1: Title, Description and Responsibilities fields. We collected a small data set (called job categorization dataset) containing 940 entries, each of them is defined by a couple (job, label), we have 23 distinct labels (manually assigned to jobs) corresponding to the 23 O*NET families (see appendix B). Table C.1 shows the summary statistics of our collected dataset and reveals that we have not enough entries for some categories like Healthcare support, Physical security and legal.

Our jobs have been modeled using the bag-of-words models together with weighting functions (see section 4.2.2) and preprocessing techniques (see section 4.2.1). Then we use SVMs to fit models to categorize jobs using 5-fold cross-validation and optimizing hyper-parameters of different kernels. We find that the results for Linear SVMs are slightly better than those for RBF and Poly SVMs. Using Linear SVM together with TF-IDF (as weighting function) and lemmatization (as preprocessing), we obtain 67% of global accuracy. An analysis of the impact of thresholds reveals that we can achieve 75% of global accuracy by adjusting used thresholds (which slightly lowers the recall of the system).

We then investigate the use of WordNet [Miller et al., 1993; Vossen, 1997] to enrich TF-IDF vectors of jobs with semantic relations (hypernyms and synonyms) of terms:

$$\text{Enriched-weight}_v(t') = \max_{(t,w)\in v} (\text{semantic-similarity}(t, t') * w) \tag{C.1}$$

| Categories (O*NET labels) | Number of instances |
|---:|:---:|
| Sales (41) | 187 |
| Tech (15) | 186 |
| Business & Finance (13) | 147 |
| Food Preparation (35) | 105 |
| Administrative (43) | 88 |
| Engineering (17) | 39 |
| Transportation (53) | 28 |
| Education (25) | 23 |
| Healthcare (29) | 22 |
| Management (11) | 20 |
| Production (51) | 19 |
| Personal Service (39) | 19 |
| Creative (27) | 14 |
| Maintenance (49) | 14 |
| Science (19) | 9 |
| Cleaning and Gardening (37) | 7 |
| Construction (47) | 5 |
| Legal (23) | 3 |
| Physical Security (33) | 3 |
| Healthcare Support (31) | 2 |
| **TOTAL** | **940** |

Table C.1 – Summary statistics extracted from our job categorization dataset.

where $t'$ and $t$ are terms, $v$ is a TF-IDF vector of a job, $w$ is the weight of the term $t$ in $v$ and semantic-similarity is the semantic similarity between 2 terms calculated using a semantic database like WordNet. Using enriched vectors of jobs together with SVM to categorize jobs leads to results (56% of global accuracy) lower than those previously obtained with TF-IDF vectors.

The use of dimensionality reduction methods (LDA and LSA described in the Section 4.2.4) fails to improve the results obtained with *SVM + TF-IDF + Lemmatization*.

Finally, we used SVMs to combine the predictions of different methods of categorization (including the *SVM + TF-IDF + Lemmatization* described above) developed by the members of my research team and I, to improve the accuracy of our job categorization system in production: this system has achieved 80% of global accuracy on our job categorization dataset.

## C.2 Job summarization

The exploration of this topic has been motivated by the need to be able to automatically extract keywords from job descriptions that sum up them the best. We use the same dataset as for the problem of job categorization (job categorization dataset). Here, jobs have been

modeled (as in section C.1) using bag-of-words models together with weighting functions (see section 4.2.2) and lemmatization (to filter out typos, abbreviations, etc.). The following basic summarizers have been proposed:

1. TF-IDF based job summarizer: returns the top-N terms with the highest TF-IDF (see Eq (4.6)) scores.
2. Log-Entropy based job summarizer: returns the top-N terms with the highest Log-Entropy (see Eq (4.7)) scores.
3. SVM-based job summarizer:
    - Fit linear SVM models to categorize jobs: learnt models contain the relevant terms for each job category.
    - For each job to summary, categorize it using learnt SVM models and return the top-N keywords (with highest scores) associated to its category in SVM models, only considering terms appearing in the job vector.

The extracted keywords have been used to categorize jobs which allows to measure the quality of extracted keywords: the results showed that TF-IDF job summarizer and SVM-based job summarizer sightly outperform the Log-Entropy based job summarizer. However, SVM-based job summarizer needs many training points to be efficient. Our results also reveal that using the top-20 extracted keywords generally yields results as good as using all the terms, this leads to lower the computation costs.

We did not do any further investigations about this topic since Work4 has changed its global strategy/business model and summarizing jobs was not a priority anymore.

# Bibliography

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99. URL http://dx.doi.org/10.1109/TKDE.2005.99.

Fabio Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 273–280, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507189. URL http://doi.acm.org/10.1145/2507157.2507189.

Ashish Anand, Ganesan Pugalenthi, Gary B Fogel, and P N Suganthan. An approach for classification of highly imbalanced data using weighting and undersampling. *Amino Acids*, 39(5):1385–91, 2010. ISSN 1438-2199. URL http://www.biomedsearch.com/nih/approach-classification-highly-imbalanced-data/20411285.html.

Jorge Aranda, Inmar Givoni, Jeremy Handcock, and Danny Tarlow. An online social network-based recommendation system. *Toronto, Ontario, Canada*, 2007.

Sujeevan Aseervatham. *Apprentissage à base de Noyaux Sémantiques pour le Traitement de Données Textuelles*. PhD thesis, Université Paris 13 – Institut Galililée, Villetaneuse, France, 12 2007.

Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommender system application: Moleskiing. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 1589–1593, New York, NY, USA, 2005. ACM. ISBN 1-58113-964-0. doi: 10.1145/1066677.1067036. URL http://doi.acm.org/10.1145/1066677.1067036.

Ricardo A. Baeza-Yates and Ribeiro-Neto Berthier. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 020139829X.

Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Comm. ACM*, 40(3):66–72, Mar. 1997.

Horace Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin. *communications of the acm*, 35(12):29–38, 1992.

Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN 1935-8237. doi: 10.1561/2200000006. URL http://dx.doi.org/10.1561/2200000006.

Yoshua Bengio. Deep learning of representations: Looking forward. In *Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

Daniel Bernardes, Mamadou Diaby, Raphaël Fournier, Françoise Fogelman Soulié, and Emmanuel Viennet. A social formalism & survey for recommender systems. 16(2), December 2014.

Michael S. Bernstein, Eytan Bakshy, Moira Burke, and Brian Karrer. Quantifying the invisible audience in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 21–30, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0.

Michael W Berry and Malu Castellanos. Survey of text mining. *Computing Reviews*, 45(9):548, 2004.

Chris Biemann. Ontology learning from text: A survey of methods. *LDV-Forum*, 20(2):75–93, 2005.

Yolanda Blanco-Fernández, Martín López-Nores, Alberto Gil-Solla, Manuel Ramos-Cabrer, and José J Pazos-Arias. Exploring synergies between content-based filtering and spreading activation techniques in knowledge-based recommender systems. *Information Sciences*, 181(21):4823–4846, 2011.

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

David M. Blei and John D. Lafferty. *Topic Models*, pages 71–94. CRC Press, 2009.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

David Bollier and Charles M Firestone. *The promise and peril of big data*. Aspen Institute, Communications and Society Program Washington, DC, USA, 2010.

Danah M Boyd and Nicole B Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2008.

J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Uncertainty in Artificial Intelligence, Proc. 14th Conf., July 1998.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

David Carmel, Naama Zwerdling, Ido Guy, Shila Ofek-Koifman, Nadav Har'el, Inbal Ronen, Erel Uziel, Sivan Yogev, and Sergey Chernov. Personalized social search based on the user's social network. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1227–1236, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1646109. URL http://doi.acm.org/10.1145/1645953.1646109.

Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, pages 33–40. Citeseer, 2005.

Peter J Carrington, John Scott, and Stanley Wasserman. *Models and methods in social network analysis*, volume 28. Cambridge university press, 2005.

Juan Luis Castro. Fuzzy logic controllers are universal approximators. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):629–635, 1995.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

Derek S. Chapman, Krista L. Uggerslev, Sarah A. Carroll, Kelly A. Piasentin, and David A. Jones. Applicant attraction to organizations and job choice: A meta-analytic review of the correlates of recruiting outcomes. *Journal of Applied Psychology*, 90:928–944, September 2005.

Kristina Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

Christina Christakou, Spyros Vrettos, and Andreas Stafylopatis. A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools*, 16 (05):771–792, 2007.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

Vincent Claveau. Vectorisation, okapi et calcul de similarité pour le tal: pour oublier enfin le tf-idf. In *Actes de la conférence Traitement Automatique des Langues Naturelles (TALN)*, 2012.

Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper, 1999.

Richard T. Cober, Douglas J. Brown, Alana J. Blumental, Dennis Doverspike, and Paul Levy. The quest for the qualified job surfer: It's time the public sector catches the wave. *Public Personnel Management*, 29(4):479–496, 2000. doi: 10.1177/009102600002900406. URL http://ppm.sagepub.com/content/29/4/479.abstract.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

T. M. Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *Electronic Computers, IEEE Transactions on*, EC-14(3): 326–334, 1965.

BD Craven and Sardar MN Islam. *Ordinary least-squares regression*. SAGE Publications, 2011.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

Luis M. de Campos, Juan M. Fernández-Luna, and Juan F. Huete. A collaborative recommender system based on probabilistic inference from fuzzy observations. *Fuzzy Sets Syst.*, 159 (12):1554–1576, June 2008. ISSN 0165-0114. doi: 10.1016/j.fss.2008.01.016. URL http://dx.doi.org/10.1016/j.fss.2008.01.016.

Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *Int. J. Approx. Reasoning*, 51(7):785–799, 2010.

Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek*, volume 27. Cambridge University Press, 2011.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

Zoran Despotovic and Karl Aberer. A probabilistic approach to predict peers? performance in p2p networks. In Matthias Klusch, Sascha Ossowski, Vipul Kashyap, and Rainer Unland, editors, *CIA*, volume 3191 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2004. ISBN 3-540-23170-6.

Mamadou Diaby and Emmanuel Viennet. Developpement d'une application de recommandation d'offres d'emploi aux utilisateurs de Facebook et Linkedin. In *Atelier Fouille de Données Complexes de la 14e Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC'14)*, Rennes, jan 2014a.

Mamadou Diaby and Emmanuel Viennet. Taxonomy-based job recommender systems on Facebook and Linkedin. In *Proceedings of the 2014 IEEE Eighth International Conference on Research Challenges in Information Science RCIS 2014*, pages 237–244. IEEE, may 2014b.

Mamadou Diaby and Emmanuel Viennet. Quantifying the hidden factors impacting the audience of job advertisements posted on facebook. In *15th Industrial Conference on Data Mining ICDM 2015*, 2015a.

Mamadou Diaby and Emmanuel Viennet. Quantifying the hidden factors affecting the audience of job advertisements posted on facebook, linkedin and twitter. *ACM Transactions on the Web*, 2015b.

Mamadou Diaby and Emmanuel Viennet. Job recommendations on social networks using a multilayer vector model. In *Workshop on Heterogeneous Information Access at WSDM 2015 (HIA'15)*, Shanghai, February 2015c.

Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 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*, pages 821–828, Aug 2013.

Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. Exploration of methodologies to improve job recommender systems on social networks. *Social Network Analysis and Mining*, 4(1):227, 2014. ISSN 1869-5450. doi: 10.1007/s13278-014-0227-z. URL http://dx.doi.org/10.1007/s13278-014-0227-z.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Karen Holcombe Ehrhart and Jonathan C. Ziegert. Why are individuals attracted to organizations? *Journal of Management*, 31(6):901–919, 2005.

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on Artificial Intelligence and Statistics*, pages 153–160, 2009.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Levent Ertoz, Michael Steinbach, and Vipin Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*, pages 105–115, 2002.

Facebook, June 2015. URL http://newsroom.fb.com/company-info/.

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge University Press, 2007.

Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, 11(2):11–34, 2006.

Jason P Fine and Robert J Gray. A proportional hazards model for the subdistribution of a competing risk. *Journal of the American Statistical Association*, 94(446):496–509, 1999.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

Fengrong Gao, Chunxiao Xing, Xiaoyong Du, and Shan Wang. Personalized service system based on hybrid filtering for digital library. *Tsinghua Science & Technology*, 12(1):1 – 8, 2007. ISSN 1007-0214.

Mike Gartrell, Xinyu Xing, Qin Lv, Aaron Beach, Richard Han, Shivakant Mishra, and Karim Seada. Enhancing group recommendation by incorporating social relationship interactions. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, pages 97–106, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0387-3. doi: 10.1145/1880071.1880087. URL http://doi.acm.org/10.1145/1880071.1880087.

Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the 4th International Conference on Trust Management*, iTrust'06, pages 93–104, Berlin, Heidelberg, 2006. Springer-Verlag. doi: 10.1007/11755593_8.

Jennifer Ann Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, College Park, MD, USA, 2005. AAI3178583.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992. ISSN 0001-0782. doi: 10.1145/138859.138867. URL http://doi.acm.org/10.1145/138859.138867.

Gene. H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970. ISSN 0029-599X. doi: 10.1007/BF02163027. URL http://dx.doi.org/10.1007/BF02163027.

Georg Groh and Christian Ehmig. Recommendations in taste related domains: Collaborative filtering vs. social filtering. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, GROUP '07, pages 127–136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-845-9. doi: 10.1145/1316624.1316643.

R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988727. URL http://doi.acm.org/10.1145/988672.988727.

Mangesh Gupte and Tina Eliassi-Rad. Measuring tie strength in implicit social networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 109–118. ACM, 2012.

Ido Guy and David Carmel. Social recommender systems. In *Proceedings of the 20th international conference companion on World wide web*, pages 283–284. ACM, 2011.

Nizar Habash, Owen Rambow, and Ryan Roth. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt*, pages 102–109, 2009.

Marc J Hadley. Web application description language (wadl). 2006.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

Lutz H Hamel. *Knowledge discovery with support vector machines*, volume 3. John Wiley & Sons, 2011.

Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *ISMIR*, pages 339–344. Utrecht, The Netherlands, 2010.

Jiawei Han. Data mining techniques. *SIGMOD Rec.*, 25(2):545–, June 1996. ISSN 0163-5808.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009. URL http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

Anne-Claire Haury. *Feature selection from gene expression data : molecular signatures for breast cancer prognosis and gene regulation network inference*. Theses, Ecole Nationale Supérieure des Mines de Paris, December 2012. URL https://pastel.archives-ouvertes.fr/pastel-00818345.

Jianming He and WesleyW. Chu. A social network-based recommender system (snrs). In Nasrullah Memon, Jennifer Jie Xu, David L. Hicks, and Hsinchun Chen, editors, *Data Mining for Social Network Data*, volume 12 of *Annals of Information Systems*, pages 47–74. Springer US, 2010. ISBN 978-1-4419-6286-7. doi: 10.1007/978-1-4419-6287-4_4. URL http://dx.doi.org/10.1007/978-1-4419-6287-4_4.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems,*, 22(1):5–53, Jan. 2004.

Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: 10.1145/223904.223929.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196, 2001.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62, 2005.

Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2011.

Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-64417-2.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

Paul B. Kantor. *Recommender systems handbook*. Springer, New York; London, 2009. ISBN 9780387858197 0387858199.

Henry Kautz, Bart Selman, and Mehul Shah. Referral web: Combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245123. URL http://doi.acm.org/10.1145/245108.245123.

Przemysław Kazienko, Katarzyna Musiał, and Tomasz Kajdanowicz. Multidimensional Social Network in the Social Recommender System. 41(4):746–759, July 2011.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, pages 81–93, 1938.

Kimmo Kettunen, Tuomas Kunttu, and Kalervo Järvelin. To stem or lemmatize a highly inflectional language in a probabilistic ir environment? *Journal of Documentation*, 61(4): 476–496, 2005.

Eunju Kim, Myungwon Kim, and Joungwoo Ryu. Collaborative filtering based on neural networks using similarity. In *Proceedings of the Second International Conference on Advances in Neural Networks - Volume Part III*, ISNN'05, pages 355–360, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-25914-7, 978-3-540-25914-5. doi: 10.1007/11427469_57. URL http://dx.doi.org/10.1007/11427469_57.

Irwin King, Michael R Lyu, and Hao Ma. Introduction to social recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 1355–1356. ACM, 2010.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245126. URL http://doi.acm.org/10.1145/245108.245126.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263.

Dieter Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

Bruce Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI magazine*, 18(2):37, 1997.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

Matthieu Latapy, Clémence Magnien, and Raphaël Fournier. Quantifying paedophile activity in a large p2p system. *Information Processing & Management*, 49(1):248–263, 2013.

Tristan Launay. *Bayesian methods for electricity load forecasting*. Theses, Université de Nantes, December 2012. URL https://tel.archives-ouvertes.fr/tel-00766237.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

Benoît Lemaire. Limites de la lemmatisation pour l'extraction de significations. *Actes des 9e Journées internationales d'Analyse Statistique des Données Textuelles (JADT'2008)*, pages 725–732, 2008.

Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

Filip Lievens and Scott Highhouse. The relation of instrumental and symbolic attributes to a company's attractiveness as an employer. *Personnel Psychology*, 56(1):75–102, 2003. ISSN 1744-6570.

Danyu Y Lin and Lee-Jen Wei. The robust inference for the cox proportional hazards model. *Journal of the American Statistical Association*, 84(408):1074–1078, 1989.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003. ISSN 1089-7801. doi: 10.1109/MIC.2003.1167344. URL http://dx.doi.org/10.1109/MIC.2003.1167344.

LinkedIn, June 2015. URL http://press.linkedin.com/about.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988. ISSN 0885-6125. doi: 10.1023/A:1022869011914. URL http://dx.doi.org/10.1023/A%3A1022869011914.

Haibin Liu, Tom Christiansen, William A Baumgartner Jr, and Karin Verspoor. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *J. Biomedical Semantics*, 3:3, 2012.

Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011. ISBN 978-0-387-85819-7.

Julie Beth Lovins. Development of a stemming algorithm. *Mechanical TRanslation and Computational Linguistics*, 11(2):22–31, June 1968.

David G Luenberger. *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA, 1973.

Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 203–210, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1571978. URL http://doi.acm.org/10.1145/1571941.1571978.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 287–296, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935877. URL http://doi.acm.org/10.1145/1935826.1935877.

Emmanuel Malherbe, Mamadou Diaby, Mario Cataldi, Emmanuel Viennet, and Marie-Aude Aufaure. Field selection for job categorization and recommendation to social network users. In *IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM'14)*, pages 588–595, August 2014.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. *Lecture Notes in Computer Science*, 3290:492–508, October 2004. URL http://www.metapress.com/link.asp?id=8BAJ2BP1HATVFGKC.

Brian P. Mathews and Tom Redman. Managerial recruitment advertisements – just how market orientated are they? *International Journal of Selection and Assessment*, 6(4):240–248, 1998. ISSN 1468-2389.

Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. In *Annual Review of Sociology*, volume 27, pages 415–444. Annual Reviews, 2001.

Prem Melville and Vikas Sindhwani. Recommender systems. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 829–838. Springer, 2010. ISBN 978-0-387-30768-8.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to wordnet: An on-line lexical database. Technical report, Computer Centrum Letteren Universiteit van Amsterdam, Aug. 1993.

Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

National Center for O*NET Development. Updating the o*net-soc taxonomy: Incorporating the 2010 soc structure. Technical report, National Center for O*NET Development, December 2010.

National Center for O*NET Development. O*net 18.0 database. Technical report, National Center for O*NET Development, July 2013.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 689–696, 2011.

Blaise Ngonmang, Savaneary Sean, and Rémi Kirche. Monetization and services on a real online social network using social network analysis. In *Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops*, ICDMW '13, pages 185–193, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-1-4799-3142-2. doi: 10.1109/ICDMW.2013.78. URL http://dx.doi.org/10.1109/ICDMW.2013.78.

Ian Soboroff. Charles Nicholas and Charles K. Nicholas. Combining content and collaboration in text filtering. In *In Proceedings of the IJCAI 99 Workshop on Machine Learning for Information Filtering*, pages 86–91, 1999.

Jorge Nocedal and Stephen J Wright. *Conjugate gradient methods*. Springer, 2006.

John O'Donovan and Barry Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, pages 167–174, New

York, NY, USA, 2005. ACM. ISBN 1-58113-894-6. doi: 10.1145/1040830.1040870. URL http://doi.acm.org/10.1145/1040830.1040870.

Zanifa Omary and Fredrick Mtenzi. Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. *International Journal for Infonomics (IJI)*, 3, Sept. 2010.

O*NET, June 2015. URL http://www.onetcenter.org/taxonomy.html.

Marjorie Paternostre, Pascal Francq, J Lamoral, D Wartel, and M Saerens. Carry, un algorithme de désuffixation pour le français. *Rapport technique du projet Galilei*, 2002.

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

Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, December 1999. ISSN 0269-2821. doi: 10.1023/A:1006544522159. URL http://dx.doi.org/10.1023/A:1006544522159.

Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-72078-2.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Norman G. Peterson, Michael D. Mumford, Walter C. Borman, P. Richard Jeanneret, Edwin A. Fleishman, Kerry Y. Levin, Michael A. Campion, Melinda S. Mayfield, Frederich P. Morgeson, Kenneth Pearlman, Marilyn K. Gowing, Anita R. Lancaster, Marilyn B. Silver, and Donna M. Dye. Understanding work using the occupational information network (o* net): Implications for practice and research. *Personnel Psychology*, 54(2):451–492, 2001. ISSN 1744-6570. doi: 10.1111/j.1744-6570.2001.tb00100.x. URL http://dx.doi.org/10.1111/j.1744-6570.2001.tb00100.x.

John Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods—support vector learning*, 3, 1999.

Carlos Porcel, A Tejeda-Lorente, MA Martínez, and Enrique Herrera-Viedma. A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Information Sciences*, 184(1):1–19, 2012.

M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5. URL http://dl.acm.org/citation.cfm?id=275537.275705.

Martin Porter. Snowball: A language for stemming algorithms, 2001.

Martin F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Anat Rafaeli, Ori Hadomi, and Tal Simons. Recruiting through advertising or employee referrals: Costs, yields, and the effects of geographic focus. *European Journal of Work and Organizational Psychology*, 14(4):355–366, 2005. doi: 10.1080/13594320500183709. URL http://dx.doi.org/10.1080/13594320500183709.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

Pradeep Ravikumar, Ambuj Tewari, and Eunho Yang. On ndcg consistency of listwise ranking methods. 2011. URL http://www.cs.utexas.edu/users/ai-lab/?RTY11.

Lei Ren, Liang He, Junzhong Gu, Weiwei Xia, and Faqing Wu. A hybrid recommender approach based on widrow-hoff learning. In *Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on*, volume 1, pages 40–45. IEEE, 2008.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. doi: 10.1145/192844.192905.

Elaine Rich. User modeling via stereotypes*. *Cognitive science*, 3(4):329–354, 1979.

Matt Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70. ACM, 2002. ISBN 1-58113-567-X. URL http://dblp.uni-trier.de/db/conf/kdd/kdd2002.html#RichardsonD02.

Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 351–368. Springer Berlin / Heidelberg, 2003.

C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 0408709294.

QuinettaM. Roberson, ChristopherJ. Collins, and Shaul Oreg. The effects of recruitment message specificity on applicant attraction to organizations. *Journal of Business and Psychology*, 19(3):319–339, 2005. ISSN 0889-3268. doi: 10.1007/s10869-004-2231-1. URL http://dx.doi.org/10.1007/s10869-004-2231-1.

J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.

Foster Provost Rong Zheng and Anindya Ghose. Social network collaborative filtering: Preliminary results. In *Sixth Workshop on eBusiness (WeB2007)*, 2007.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. 2008.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975. ISSN 0001-0782.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Helmut Schmid. Probabilistic part-of-speech tagging using decision trees, 1994.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer, 1997.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko. *High performance MySQL: Optimization, backups, and replication*. " O'Reilly Media, Inc.", 2012.

Julie Séguela. *Fouille de données textuelles et systèmes de recommandation appliqués aux offres d'emploi diffusées sur le web*. PhD thesis, Conservatoire National des Arts et Métiers (CNAM), Paris, France, May 2012.

Shang Shang, Pan Hui, Sanjeev R. Kulkarni, and Paul W. Cuff. Wisdom of the crowd: Incorporating social influence in recommendation models. In *Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems*, ICPADS '11, pages 835–840, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4576-9. doi: 10.1109/ICPADS.2011.150. URL http://dx.doi.org/10.1109/ICPADS.2011.150.

Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: 10.1145/223904.223931. URL http://dx.doi.org/10.1145/223904.223931.

Subhash K Shinde and Uday Kulkarni. Hybrid personalized recommender system using centering-bunching based clustering algorithm. *Expert Systems with Applications*, 39(1): 1381–1387, 2012.

Rashmi R Sinha and Kirsten Swearingen. Comparing recommendations made by online systems and friends. In *DELOS workshop: personalisation and recommender systems in digital libraries*, volume 1, 2001.

Lindsay I Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

J.F. Sowa, Nov. 2010. URL http://www.jfsowa.com/ontology/.

Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing & Management*, 36(6):779–840, 2000.

V Srividhya and R Anitha. Evaluating preprocessing techniques in text categorization. *International Journal of Computer Science and Application*, 47(11), 2011.

Dianna L. Stone. E-recruiting : online strategies for attracting talent. *The brave new world of eHR : human resources management in the digital age.*, 2005.

Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009. ISSN 1687-7470. doi: 10.1155/2009/421425. URL http://dx.doi.org/10.1155/2009/421425.

Roshan Sumbaly, Jay Kreps, and Sam Shah. The big data ecosystem at linkedin. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1125–1134, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2037-5. doi: 10.1145/2463676.2463707. URL http://doi.acm.org/10.1145/2463676.2463707.

Gero Szepannek, Matthias Gruhne, Bernd Bischl, Sebastian Krey, Tamas Harczos, Frank Klefenz, Christian Dittmar, and Claus Weihs. Perceptually based phoneme recognition in popular music. In *Classification as a Tool for Research*, pages 751–758. Springer, 2010.

Jiliang Tang, Xia Hu, and Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013. ISSN 1869-5450. doi: 10.1007/s13278-013-0141-9. URL http://dx.doi.org/10.1007/s13278-013-0141-9.

Don Tapscott. *Grown Up Digital: How the Net Generation is Changing Your World HC*. Mcgraw-Hill, 1 edition, 2008. ISBN 0071508635, 9780071508636.

Terry M Therneau. *Modeling survival data: extending the Cox model*. Springer, 2000.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

Brendon Towle and Clark Quinn. Knowledge based recommender systems using explicit user models. In *Proceedings of the AAAI Workshop on Knowledge-Based Electronic Markets*, pages 74–77, 2000.

Daniel B. Turban, Monica L. Forret, and Cheryl L. Hendrickson. Applicant attraction to firms: Influences of organization reputation, job and organizational attributes, and recruiter behaviors. *Journal of Vocational Behavior*, 52(1):24 – 44, 1998. ISSN 0001-8791. doi: http://dx.doi.org/10.1006/jvbe.1996.1555. URL http://www.sciencedirect.com/science/article/pii/S0001879196915559.

Twitter, June 2015. URL https://about.twitter.com/company.

Vladimir Vapnik. *The nature of statistical learning theory*. springer, 2000.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Piek Vossen. Eurowordnet: a multilingual database for information retrieval. Technical report, Computer Centrum Letteren Universiteit van Amsterdam, 1997.

Manolis G Vozalis and Konstantinos G Margaritis. Using svd and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, 177(15):3017–3037, 2007.

Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7.

Jian Wang, Yi Zhang, Christian Posse, and Anmol Bhasin. Is it time for a career switch? In *Proceedings of the 22nd international conference on World Wide Web*, pages 1377–1388. International World Wide Web Conferences Steering Committee, 2013. URL http://www2013.org/proceedings/p1377.pdf.

Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974a.

P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974b.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

Ian O. Williamson, David P. Lepak, and James King. The effect of company recruitment web site orientation on individuals' perceptions of organizational attractiveness. *Journal of Vocational Behavior*, 63(2):242 – 263, 2003. ISSN 0001-8791. doi: http://dx.doi.org/10.1016/S0001-8791(03)00043-5. URL http://www.sciencedirect.com/science/article/pii/S0001879103000435. Special Issue on Technology and Careers.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

Zhonghang Xia, Yulin Dong, and Guangming Xing. Support vector machines for collaborative filtering. In *Proceedings of the 44th Annual Southeast Regional Conference*, ACM-SE 44, pages 169–174, New York, NY, USA, 2006. ACM. ISBN 1-59593-315-8. doi: 10.1145/1185448.1185487. URL http://doi.acm.org/10.1145/1185448.1185487.

Bo Xiao and Izak Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *MIS Q.*, 31(1):137–209, March 2007. ISSN 0276-7783. URL http://dl.acm.org/citation.cfm?id=2017327.2017335.

Allen Y Yang, Shankar S Sastry, Arvind Ganesh, and Yi Ma. Fast l1-minimization algorithms and an application in robust face recognition: A review. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1849–1852. IEEE, 2010.

B Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

Forrest W Young, Jan De Leeuw, and Yoshio Takane. Regression with qualitative and quantitative variables: An alternating least squares method with optimal scaling features. *Psychometrika*, 41(4):505–529, 1976.

L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. ISSN 0019-9958.

ANK Zaman, Pascal Matsakis, and Charles Brown. Evaluation of stop word lists in text retrieval using latent semantic indexing. In *Digital Information Management (ICDIM), 2011 Sixth International Conference on*, pages 133–136. IEEE, 2011.

Jerrold H Zar. Spearman rank correlation. *Encyclopedia of Biostatistics*, 1998.

Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 478–481. IEEE, 2010.

Rong Zheng, Dennis Wilkinson, and Foster Provost. Social network collaborative filtering. *Stern, IOMS Department, CeDER, Vol*, 2008.

Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, December 2005. ISSN 1387-3326. doi: 10.1007/s10796-005-4807-3. URL http://dx.doi.org/10.1007/s10796-005-4807-3.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.