Cold-start recommendation : from Algorithm Portfolios to Job Applicant Matching

François Gonard

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Cold-start recommendation: from Algorithm Portfolios to Job Applicant Matching

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 31/05/2018, par

François Gonard

Composition du jury :

M. Patrick Gallinari Professeur des universités, Université Pierre et Marie Curie	Rapporteur
M. Amaury Habrard Professeur des universités, Université Jean Monnet Saint-Étienne	Examinateur
M. Jin-Kao Hao Professeur des universités, Université d'Angers	Examinateur
M. Holger H. Hoos Professor, Universiteit Leiden	Rapporteur
M. Marc Schoenauer Directeur de recherche, INRIA, Université Paris-Sud	Directeur de thèse
Mme Michèle Sebag Directrice de recherche, LRI, Université Paris-Sud	Co-encadrante de thèse
Mme Anne Vilnat Professeure des universités, LIMSI, Université Paris-Sud	Examinatrice, Présidente

Abstract

The need for personalized recommendation has considerably increased with the advent of huge databases, whose size exceeds by far the human scale. Such an overabundance of information (commercial products, potential connections in social networks, ...) has driven the development of recommender systems (RS). A RS typically learns the user interests from past recorded activities, and consistently processes new incoming items to include them (or not) in the user-specific recommendations.

Recommendation is also desirable when estimating the relevance of an item requires complex reasoning based on experience: supervised machine learning techniques are good candidates to simulate experience from large amounts of data.

In both cases, the accuracy of decisions to recommend or not items is heavily dependent on the ability to generalize from past data.

The present thesis focuses on the *cold-start* context in recommendation, *i.e.* the situation where either a new user desires recommendations, or a brand-new item is to be recommended. Cold-start is specially challenging to RSs since no past interaction data is available to infer the user interests or the item peculiarities. Instead, RSs have to base their reasoning for recommendations on side descriptions. Two recommendation problems are investigated in this work, and two recommender systems are proposed to handle the cold-start issue.

The problem of choosing an optimization algorithm in a portfolio can be cast as a recommendation problem. In order to both reduce the optimization cost of a brand-new problem instance and mitigate the risk of optimization failure, we propose a two-components system. First, an algorithm selector is used to leverage the description of problem instances, thus enabling peak per-instance performances to be achieved. Second, this selector is combined with a sequential scheduler, that enforces the generalization ability at a moderate cost. Both components are trained on past data to simulate experience, and alternatively optimized to enforce their cooperation. The resulting system won the Open Algorithm Selection Challenge 2017, focusing on algorithm portfolios for combinatorial optimization.

Automatic job-applicant matching (JAM) has recently received considerable attention in the recommendation community as online recruitment platforms and social networks are now common media used to connect job seekers and companies. We develop specific natural language (NL) modeling techniques and combine them with standard recommendation procedures to leverage past user interactions and the (mostly) textual descriptions of job positions. The NL and recommendation aspects of the JAM problem are studied on two real-world datasets. The appropriateness of various RSs on applications similar to the JAM problem are discussed.

Synthèse

L'avènement de bases de données de grande taille, impossibles à appréhender par des humains, a rendu indispensable la mise à disposition de recommandations personnalisées. Cette surabondance d'information (de produits de consommation, de relations potentielles dans les réseaux sociaux, etc.) a été le moteur du développement des systèmes de recommandation. L'activité d'un utilisateur est enregistrée et utilisée par ces systèmes pour apprendre ses centres d'intérêt. Ces systèmes doivent également traiter en permanence les données concernant de nouveaux objets, afin d'éventuellement les inclure dans les recommandations personnalisées.

Il est également intéressant de disposer de recommandations lorsqu'estimer la pertinence d'un objet est complexe, et repose essentiellement sur le jugement d'experts. Les techniques d'apprentissage automatique supervisé sont alors d'excellents moyens de simuler l'expérience requise par le traitement de grandes quantités de données.

Dans les deux cas, la décision de recommander ou non des objets avec précision est grandement influencée par la capacité à étendre à de nouveaux exemples le raisonnement qui a été appris à partir des données passées.

Cette thèse examine le problème du *démarrage à froid* en recommandation, c'est-à-dire une situation dans laquelle soit un tout nouvel utilisateur désire des recommandations, soit un tout nouvel objet est proposé à la recommandation. Le démarrage à froid est un défi pour les systèmes de recommandation puisqu'aucune donnée d'intéraction n'est disponible pour inférer les souhaits de l'utilisateur ou les caractéristiques de l'objet. Le raisonnement qui aboutit aux recommandations repose alors sur des descriptions externes de l'utilisateur et/ou de l'objet. Deux problèmes de recommandation de ce type sont étudiés ici, pour lesquels des systèmes de recommandation spécialement conçus pour le démarrage à froid sont présentés.

En optimisation, il est possible d'aborder le problème du choix d'algorithme dans un portfolio comme un problème de recommandation. Notre première contribution concerne un système à deux composants qui vise à réduire le coût de l'optimisation d'une nouvelle instance tout en limitant le risque d'un échec de l'optimisation. Le sélecteur d'algorithme est le composant clé pour atteindre d'excellentes performances pour l'instance courante : il reproduit le jugement expert à partir de la description de l'instance. Afin que le sélecteur concentre ses efforts sur les instances difficiles à traiter, un ordonnanceur d'algorithmes est utilisé pour résoudre les instances faciles pour un faible surcoût. Les deux composants sont entrainés sur les données du passé afin de simuler l'expérience, et sont alternativement optimisés afin de les faire coopérer. Le système ainsi formé a remporté l'*Open Algorithm Selection Challenge* 2017, qui se focalisait sur l'optimisation combinatoire et les problèmes de décision.

L'appariement automatique de chercheurs d'emploi et d'offres est un sujet d'étude majeur depuis que les plateformes de recrutement en ligne et les réseaux sociaux constituent un moyen privilégié de mettre en contact candidats et entreprises. Une seconde contribution concerne le développement de réseaux de neurones articifiels pour la modélisation du langage naturel et leur combinaison avec des techniques de recommandation classiques. Ainsi, les systèmes proposés tirent profit à la fois des intéractions passées des utilisateurs et des descriptions textuelles des annonces. Le problème d'appariement d'offres et de chercheurs d'emploi est étudié à travers le prisme du langage naturel et de la recommandation sur deux jeux de données tirés de contextes réels. Les systèmes de recommandation proposés sont ensuite évalués sur ces données propriétaires et comparés à l'état de l'art sur

des données publiques. Une discussion sur la pertinence des différents systèmes de recommandations pour des applications similaires est proposée.

Acknowledgments

I used to think that doing a PhD was above all a personal adventure. But this is not entirely true: anyone among your relatives or in your professional circle can affect the way you live this experience and how you move forward. I met the following people, discussed with them or just felt that they were standing by my side; they all contributed to my adventure in one way or the other:

First and foremost I am deeply grateful to my advisors Marc and Michèle for their kindness, the discussions, the ideas, their guidance and their support during these 3+ years. I truly think this was a chance to work with you.

I deeply thank Patrick Gallinari and Holger Hoos for carefully reviewing my thesis and for the valuable comments and suggestions to improve this work. I also would like to thank Anne Vilnat, Amaury Habrard and Jin-Kao Hao for accepting being members of the Jury and for the interest you showed during the defence. It was a honor to have you all assess my work.

I want to thank all my friends at IRT SystemX and in the TAO (and later TAU) team. Special thoughts to Nacim and Thomas with whom I closely worked; to Sarah, Tim, Félix, Paul & Paul, Pierre, Yves, Gabriel, Jean-Patrick, Karim, Virginie, Nathalie, Julie, Gauthier, Romain, Mouadh, Mathieu & Mathieu, Michel, Ming, Sullivan for jogging and football sessions, beers, discussions, advices, or just for laugh out loud moments. Once I have been trapped with some of them inside a bar because there was a suspicious package outside: best excuse ever to drink one or two (more) pints.

Thanks to Simon, Yoann, Laurent and my friends from Supaero who came at my defence or sent messages to support me.

I don't forget that this PhD was funded by IRT SystemX and was made possible thanks to quite a few people; in particular, I would like to thank Yves Tourbier, who was my internship supervisor at Renault and asked me if I would be interested in doing a PhD at SystemX. I would like to use this opportunity to thank Emmanuel Rachelson who was one of my professors at Supaero and first introduced me to machine learning.

I am also grateful to the administrative and IT services at SystemX and TAU for making the life of a PhD student easier; to Stéphanie Delestre from Qapa, the Association Bernard Gregory and the COSEAL group for making the data I used in this work available to us.

Finally, I want to warmly thank my family for their support since much before I started this PhD whatever I chose and Alexia for supporting me every day and forgiving me for not coming home on my 25^{th} birthday because, on that night, there was a deadline.

Contents

Chapter 1

Selection and recommendation in a cold-start context

Selection is defined as the process of choosing one out of a set of alternatives, retaining the best one and discarding all other alternatives that are perceived as less satisfying. Selection is an active process that reflects both the utility function and the intelligence (or the computational power) of the decision maker. Selection can be viewed as a key building block in artificial intelligence systems, either supporting complex processes (*e.g.* feature or example selection in machine learning [Blum and Langley, 1997]) or delivering the output of the AI system.

Recommendation can be viewed as a preparatory phase of selection, where the recommender system selects and ranks a small subset of alternatives out of a (usually very large) set of alternatives. A main difference between recommendation and selection is that in recommendation setting the final decision – *i.e.* the actual *selection* – is not made by the system. The recommender system only supplies the actual decision maker, called the user in the following, with a restricted set of alternatives. Typically, the system is not fully informed about the interestingness function of the user.

1.1 Context of the thesis

The presented work is concerned with *automatic* selection and recommendation. The calibration of the selection/recommendation system (SRS) is achieved by machine learning from a previously recorded dataset. In the following, only the offline setting will be considered, *i.e.* there is no "user in the loop" cooperating with the SRS; and further information gained by the system after the learning phase is not exploited − as opposed to the online setting also referred to as lifelong learning. Following common practice, alternatives to be recommended or selected will be referred to as *items*.[1]

1.1.1 A machine learning perspective

Selection and recommendation systems mostly differ regarding who will make the eventual decision: the system itself (selection setting) or the human being referred to as *user* (recommendation setting). Naturally, a good recommender system is expected to provide high-quality suggestions, matching the selections that would have been made if the user had infinite time and computational resources. In other words, the ultimate goal of a

[1] Though in some contexts, such as social networks, items to be recommended may be users.

recommender system is to learn and implement the utility function, or function of interestingness, of the particular user. The main distinction thus becomes that the system is allowed to deliver one choice in the selection setting *vs* a set of choices in the recommendation setting. The human-machine interaction aspects of SRS, the user's trust in the system and how to favor it, are outside the scope of the presented work, and selection and recommendation will be used interchangeably in the following.

An SRS usually benefits from data related to a number of users; therefore it has a substantially wider view of the application domain than any single user (who, in the general case, only has a partial view of the available items). The point of SRS will thus be to exploit the dataset, or community archive, to support the profiling of user interests, of item features, and their correspondence. This knowledge is used in a second phase to form the prediction (*e.g.* list of recommendations or set of selected items). Such an approach based on past data typically is cast as a machine learning problem. The SRS methods usually involve little prior knowledge about the domain of application, making them fairly general and applicable – under the usual assumption data of good quality (user and item descriptions, feedback).

1.1.2 The cold-start context

The item *cold-start* problem (I-CS) is stated as the problem of recommending items for which there does not yet exist any feedback [Schein et al., 2002]. Such a situation commonly occurs in e-commerce whenever a new product is listed for sale. A recommender system must then rely on an external source of information about the item, typically its description. Naturally, the item description might not be as informative regarding the user preferences as the feedback itself.

Likewise, the user *cold-start* problem (U-CS) is stated as the problem of recommending items to a brand new user who did not give any feedback yet, *e.g.* a user that has just subscribed to a website and whose interests are yet unknown. The recommender system must also rely on side information (age, gender...or user profile) to make recommendations.

The I-CS and U-CS problem, though similar, differ as users and items do not play symmetrical roles in the recommendation problem. Typically, a user will make her selection among a set of items (and the desirability of a given item might thus depend on the other items); in contrast, the user behaviors are assumed to be independent of each other.[2]

For the sake of completeness, two other settings are introduced. The recommendation of known items to known users, referred to as warm-start recommendation problem (or recommendation problem for short), has been extensively studied in the recommender system literature, particularly so since the Netflix challenge [Bennett et al., 2007; Weimer et al., 2008; Koren et al., 2009; Salakhutdinov et al., 2007]. The recommendation of new items to new users, referred to as full cold-start recommendation problem, has received little attention in the recommender systems community. In most cases this very specific setting is handled as a two-stage process, *e.g.* recommending the globally top-ranked items first and thereafter gradually refining the user profile based on her feedback. As an alternative, full cold-start recommendation can be tackled as an information retrieval problem, matching a query (the user profile) with documents (items, through their description).

In the following, the *item-oriented* (I-CS) and *user oriented* (U-CS) cold-start problems will be referred to as *semi cold-start* when there is no ambiguity. It is emphasized

[2]Except through the "taste propagation" achieved by the SRS and/or the commercial biases implemented in the recommendation platform.

that the "cold-start" term refers to the recommendation of *brand-new* items or to *brand-new* users (also termed cold-start in the "narrow sense" in [Gantner et al., 2010] or *hard cold-start*), *i.e.* of items or users with not a single known interaction.

1.2 Motivation

As previously mentioned, recommender and selection systems have a wide range of applications. Thanks to decades of effort especially in the recommendation area, a number of large data resources have been made publicly accessible; they foster the design and enable the rigorous assessment of recommender systems, spanning a variety of applications and research goals. Two specific application domains will be considered in this work, illustrating the variety of research questions, of algorithmic approaches, and of validation criteria involved in SRSs.

1.2.1 Selection supporting the expert: algorithm portfolios for optimization

The first application regards algorithm selection in order to solve a particular problem instance. In many contexts, several algorithms are likely to solve a problem instance, though the provided solutions might be more or less satisfactory for the problem owner. An algorithm portfolio is a set of algorithms that are all applicable to the same problem instances, and desirably such that any problem instance is satisfactorily solved by at least one algorithm in the portfolio. The selection problem at hand thus is to select the best algorithm in the portfolio i) for the new problem instance at hand and ii) w.r.t. the considered performance criteria. In this context, problem instances will be referred to as *users* and portfolio algorithms as *items*. By construction, the SRS essentially aims at brand new problem instances, and selecting the best algorithm for each one. This goal is cast as a U-CS problem.

Part I is devoted to algorithm portfolios in the context of optimization. The number of considered algorithms is typically a few dozens; the experienced optimization practitioner usually knows them all, with a good expertise of several of them. The difficulty is to predict the performances of a particular algorithm on the new instance: algorithms and instances are complex objects. After pre-selecting the algorithms appropriate for the considered range of problem instances (the pre-selection aspect is not addressed here), algorithm selection has long relied on manual expertise. We consider here automatic algorithm selection methods, which have been studied and used for over a decade [Gomes and Selman, 2001; Leyton-Brown et al., 2003]. The goal is to see how machine learning-based algorithms can be leveraged to achieve algorithm selection, exploiting significant expertise and datasets about the portfolio behaviors on former problem instances.

1.2.2 Recommendation for society: job advertisements recommendation

The second application focuses on the automatic recommendation of job advertisements (items, referred to as jobs in the following) to applicants (users). This problem, tackled in Part II, is cast as an I-CS problem (more in Chap. 6). The number of items is generally large; users can hardly review even a small fraction thereof and thus use filters. Currently, the search for job ads appropriate to a given user proceeds by combining search query (information retrieval) and hard filtering (along specific criteria like sector, type of contract, location...). The use of machine learning for job recommendation is a recent and

promising direction for online recruitment platforms [Malherbe et al., 2014; Carpi et al., 2016].

1.3 Main contributions and organization of the work

The first SRS problem tackled in this manuscript concerns algorithm selection for optimization; the goal is to "cross the chasm" of a purely automatic algorithm selection setup and deliver the best performances of the algorithm portfolio for the problem instance at hand.

The second part tackles the automatic recommendation of job ads to applicants. Beyond efficient recommendation, we shall see that the matching learning approach opens promising perspectives beyond job recommendation (typically in the domain of continuing education).

1.3.1 Per-instance algorithm selection within a portfolio

The most common and simplest approach consists of: i) determining the most efficient algorithm on average in a first stage (benchmark phase); ii) always using it in the second stage, referred to as production or operational phase. Still, the most effective approach relies on the *per-instance* selection of one or a small set of appropriate algorithms. This manuscript investigates two ways of combining portfolio algorithm to achieve per-instance efficient resolution:

Algorithm Selector And Pre-scheduler, version 1 The per-instance selection of a single algorithm is meant to determine for each instance the algorithm that performs best. This approach might however be adversely affected due to the inappropriate or insufficient description of the problem instances, occasionally leading to catastrophic failures and compromising the average performance of the approach. Inspired by other algorithm portfolio techniques [Xu et al., 2008; Kadioglu et al., 2011; Xu et al., 2012b; Malitsky et al., 2013; Hoos et al., 2015], our first contribution within the ASAP.V1 system is to combine algorithm selection with another portfolio method, referred to as pre-scheduler. In comparison to similar algorithm selection approaches [Xu et al., 2008; Kadioglu et al., 2011], careful attention is paid to the trade-off between performance and robustness on a wide range of optimization problems. ASAP.V1 also introduces a simple and elegant new way of improving the global performances by enhancing the algorithm selector with information gained during solving with the pre-scheduler. ASAP.V1 received a honourable mention during the ICON challenge on Algorithm selection (2015) [Kotthoff, 2015].

Algorithm Selector And Pre-scheduler, version 2 ASAP.V1 was upgraded and yielded ASAP.V2, based on the alternating optimization of ASAP components. The main limitation faced by ASAP.V2 is *overfitting*. Several methodologies have been designed and investigated to avoid overfitting. Their extensive empirical assessment shows that ASAP.V2 preserves the robustness of ASAP.V1 while achieving improved performances. ASAP.V2 won the Open Algorithm Selection Challenge in 2017 [Kotthoff et al., 2017].

1.3.2 Contributions to the job-applicant matching problem

The job-applicant matching problem defines a complex recommendation problem at the crossroad of natural language processing, representation learning and pure recommender

systems. The key difficulties regard the representation of users and items on the one hand, and the scalability w.r.t. large-scale datasets on the other hand. Two main research directions have been explored.

Toward a better understanding of the job-applicant matching problem The core of the study is based on two proprietary databases; these are analyzed to inspect their structures and their quality, and determine how their information can be represented. A first remark regards the inconsistency between the user description (their resumes) and their behavior (their clicks on job ads). This inconsistency can be leveraged to characterize "difficult" users. Based on the lessons learned, an original classification of recommender systems is proposed, distinguishing monothetic and polythetic users.

Neural architectures for job advertisement recommendation Inspired from continuous language representation [Bengio et al., 2003; Le and Mikolov, 2014] and from similarity learning [Chopra et al., 2005], two neuronal architectures are designed for job recommendation:
LaJam learns a metric or similarity on the job ad space, which is leveraged using the collaborative data to select job ads similar to the job ads formerly selected by the user. Some key aspects regard the regularization of the similarity and its scalability.
MaTJam maps users and items alike onto a single continuous space, supporting the alignment of applicant skills and jobs specifics in a latent space. This approach relates to machine translation. Both architectures are assessed against the proprietary databases and compared with related methods on publicly available datasets.

1.3.3 Thesis outline

Chapter 2 provides an overview of the optimization field and focuses on the positioning of algorithm portfolios approaches with respect to other ensemble methods.

Chapter 3 introduces the most prominent portfolios techniques and reviews a selection of recent state-of-the-art portfolio systems to highlight good practices for designing efficient portfolios.

ASAP systems version 1 and 2 are described and analyzed in Chapter 4, with a special focus on the assessment of the new techniques introduced. Experimental results are described and discussed.

Chapter 5 summarizes the main lessons on algorithm selectors designs and provides perspectives of further research.

The job-applicant automatic matching problem is presented in Chapter 6. Formal background on recommender systems and Natural Language Processing (NLP) relevant to this application are introduced. Chapter 6 finally proposes a criteria to distinguish among and structure the recommender systems.

Chapter 7 details the real-world datasets at the core of our empirical study. Their analysis provides insight on the specifics of this application.

The LaJam and MaTJam systems are described and situated with respect to existing techniques in Chapter 8. They are assessed in terms of performance; for the sake of comparison with the state of the art, additional experiments on a public dataset are reported and discussed.

Chapter 9 presents our main conclusions about the job applicant matching problem and how it is currently handled by recommender systems. Several directions are described for further research.

Chapter 10 concludes the thesis with a summary of our contributions, a further discussion on the applications and the perspectives of bridging the gap between the two directions of research explored in this work.

Part I

Algorithm portfolios to support the optimization expert

Chapter 2

Background on optimization

Optimization has been for decades and centuries, and still is, at the core of problem solving in science, engineering and economics. It is not limited to the professional context, as many decisions of everyday life aim to "make the best" out of a given situation. This chapter briefly reviews the basics of computational optimization, referring the interested reader to a few key references for a more comprehensive presentation. Our objective is to present the optimization landscape in relation with algorithm portfolios which form the core of Part I of the thesis.

2.1 Background

Optimization is in essence the *selection of a best solution among a set of alternatives*. Let $\mathcal{S}$[1] denote the set of alternatives, and f denote a deterministic evaluation or objective function[2] on $\mathcal{S}$. An optimization problem is formalized as:

$$\text{"optimize"}_{x \in \mathcal{S}} \ f(x) \tag{2.1}$$

where f takes values in $\mathbb{R}$ (single-objective optimization) or $\mathbb{R}^d$ (multi-objective optimization). Only the single-objective optimization framework will be considered in the following, assuming that set $\{f(x)\}_{x \in \mathcal{S}}$ is endowed with a total order relationship. Assuming without loss of generality that $f(x)$ is to be minimized, any optimizer (or minimizer) x^* of f is classically defined as:

$$x^* \in \operatorname*{argmin}_x f(x) \tag{2.2}$$

Problem (2.1) is formally equivalent to the following constrained optimization problem:

$$\begin{aligned} &\min_{x \in \mathcal{X}} \ f(x) \\ &\text{subject to } x \in \mathcal{S} \end{aligned} \tag{2.3}$$

where $\mathcal{X}$ is referred to as *search space* and contains the set of alternatives $\mathcal{S}$: $\mathcal{S} \subseteq \mathcal{X} \subseteq \mathbb{R}^n$. The constraint $x \in \mathcal{S}$ is usually handled through appropriate elementary constraints $\{g_j(x) \leq 0\}_j$.

[1] We may restrict to $\mathcal{S} \subseteq \mathbb{R}^n \times [0,1]^p$ as we are interested in numerical and combinatorial optimization only. In all generality, optimization problems may be defined over more complex object sets – *e.g.* functions – which cannot always be encoded as such.

[2] The case of noisy or stochastic objective functions requires specific techniques [Cauwet et al., 2016] and is beyond the scope of this work.

2.1.1 Computational optimization

The main focus is on problems where x^* cannot be determined in closed form. Most approaches rely on the successive evaluation of a subset of alternatives, also called *candidate solutions*. For non-trivial optimization problems, the evaluation of all $\{f(x)\}_{x\in\mathcal{S}}$ is intractable for scalability reasons, in particular in the continuous case, exacerbating the need for smart strategies to select a relatively "small" number of alternatives to be evaluated before returning one. Such computational optimization strategies, implemented as algorithms, are assessed along specific issues:

- the quality of the returned alternatives with respect to the true optimum;
- the overall computational cost, including the strategy inner computation (*i.e.* resources spent to decide which candidate to evaluate next) and extraction of information about f (*e.g.* evaluation, approximate or natural gradient).

Both issues are often jointly addressed in the literature, as the quality of the returned alternative usually increases with the computational resources. In the black-box optimization benchmarks for instance [Auger et al., 2012; Hansen et al., 2016], algorithms are compared in terms of resources needed to find the optimizer or an ϵ-approximation thereof (*e.g.* an alternative $x \in \mathcal{S}$ such that $f(x) - f(x^*) < \epsilon$), or in terms of the quality of solutions obtained for a fixed resource budget.

Many research frameworks focus on either the general *black-box* setting (where objective f is only accessible through point evaluation, and its gradient is unknown), or on specific classes of functions. In both cases, algorithm performance is assessed based on the number of candidate solutions evaluated to deliver a solution of a prescribed quality. This performance measure is relevant to all problems where the evaluation cost far exceeds the inner computation of the optimization strategy.[3] Another performance measure is the *computational time* needed to solve the optimization problem (assuming of course that all algorithms run in the same carefully controlled environment). Competitions typically offer the fairest framework to evaluate algorithms using their running time (in short "runtime" in the following) as a measure of performance (see Sec. 2.1.3).

2.1.2 Optimization problems and algorithm classification

A coarse classification of optimization problems is proposed as follows:

- Is the objective single of multi-valued? Hereinafter we focus on single-objective optimization.
- Is the objective deterministic or stochastic ? Hereinafter we focus on deterministic (noise-free) optimization.
- Is the search space continuous, discrete or mixed?
- Is the optimization constrained or unconstrained? Constraint satisfaction problems (CSPs) aim at finding any solution that satisfies a given set of constraints; all such solutions, referred to as admissible solutions, are deemed equally good. Although CSPs do not involve any explicit objective function, they nevertheless fit the general optimization framework by considering a constant objective function.

[3] However, this measure is only meaningful when all candidates are equally *costly* to evaluate. In contrast, the detection of simple (*e.g.* bound) constraints violation is easy and allows to skip further computations [Booker et al., 1999].

Each category of problems defined from the above criteria is addressed by a specific community (*e.g.* multi-objective optimization, optimization under uncertainty); algorithms tailored to this problem category usually hardly extend "as is" beyond it.

The first task for the optimization practitioner thus consists in identifying the category of problems relevant to the problem instance at hand and hence the set of algorithms well-suited to solve it.

Exact *vs* approximation algorithms Besides the above problem categorization, one must consider the algorithm categorization. *Exact algorithms* provide the guarantee to return the optimal solution (or a solution arbitrarily close to it in the continuous case) when run to completion. They often come with a high computational cost, as worstcase guarantees require the ability to explore[4] the whole search space. As such, exact algorithms are also capable of proving that a given solution quality is unreachable, or that a constraint satisfaction problem is unsatisfiable. In such cases, the time needed to arrive at an unsatisfiability result is taken as the time needed to solve the problem. Exact algorithms however do not scale up in the worstcase: the computational budget required to reach completion[5] increases with the complexity of the problem instance (in a polynomial or most often exponential manner).[6] To overcome this limitation, *approximation algorithms* have been developed, amenable to tackle much harder[7] problem instances reliably (*i.e.* even in the worstcase) at the expense of the loss of guarantee of optimality.[8]

2.1.3 Empirical evaluation of optimization algorithms

As noted in Sec. 2.1.1, optimization algorithms are assessed using two criteria: their ability to find a good quality candidate solution and the amount of resources required. The most prominent computation resource is the runtime, though it varies with the machine and the implementation used. For these reasons, the number of point evaluations (each evaluation provides the objective and constraints evaluation of the current solution, as well as other measures such as the gradient in that point, if available) is sometimes preferred, especially for computationally costly objective functions where the evaluation cost surpasses by far the time required by the optimization algorithm itself. It is, however, not appropriate when the optimization algorithm requires only *partial* evaluation (such as evaluation of a subset on constraints) in each iteration.

The need for empirical evaluation comes with the following three observations. Firstly, whenever theoretical results exist about asymptotic convergence, they mostly deal with worstcase scenarios, whereas the optimization practitioner is in most cases interested in average-case performance. Secondly, algorithms can in practice be run on problem instances that do not fulfill the assumptions underlying the algorithm performance guarantees. Thirdly, there exist many efficient algorithms that do not come with theoretical guarantees, especially so for heuristic methods.

[4]This may involve effectively evaluate all candidates in a subspace or analyze the subspace and determine, *e.g.* that it cannot contain an optimal solution, that an optimal candidate has already been found, etc.

[5]One can argue that most often such algorithms are anytime, *i.e.* are able to return "good quality" solutions long before completion.

[6]This is however not always problematic as i) the asymptotic scaling involves constants that may make an exponential algorithm "efficient" and ii) that some algorithms with poor (worstcase) scaling are known to scale well in practice [Spielman and Teng, 2004].

[7]Determining the hardness of a problem is a research subject in itself, see, *e.g.* [Smith-Miles and Lopes, 2012] and references therein.

[8]Still, such algorithms come with guarantees on the quality of the solution returned.

To provide fair, reproducible evaluation of algorithms and stimulate algorithm designers, optimization competitions are organized by research communities on a regular basis (*e.g.* , SAT for satisfiability problems[9], BBComp for multi-objective black-box optimization[10]). They usually release materials (evaluation test bed and optimization performances of competing algorithms...) and thus serve as reference benchmarks for research.

2.1.4 Black-box optimization

Black-box optimization deals with optimization problems where no information on a problem is given but the search space and pointwise evaluation (if defined). This broad optimization framework includes the case where the evaluation is actually delegated to a third-party software. A typical evaluation framework measures the quality of the best possible solution for a predefined number of available queries, or the ability to reach a target quality with the minimum number of queries. An prominent example of a black-box optimizer is CMA-ES [Hansen et al., 2003], used in Sec. 4.5.1.

2.1.5 No Free Lunch theorem and implications

A fundamental negative result dubbed the *No-free lunch* (NFL) theorem for optimization [Wolpert and Macready, 1997] has been established in the context of black-box, unconstrained computational optimization. Under the assumption that all optimization problems are equally likely, the NFL states that all algorithms perform equally well on average. A corollary thereof, *i.e.* that the search for a perfect general-purpose black-box optimization algorithm is vain, is however at odds with the practical lessons from benchmarks, showing that some algorithms can consistently dominate others. For Wolpert and Macready [1997], the NFL theorem expresses that each algorithm is optimal for a particular distribution of problem instances. Along the same lines, the knowledge about the problem instance at hand is key to guide the selection of the (quasi) best algorithm for this instance, as done in practice: benchmarks focus on a particular classes of problems (*e.g.* SAT, TSP...). Moreover their setup often splits the competition into tracks (*e.g.* randomly generated problems, manually designed problems and application problems for the SAT11 competition) reflecting differences – not necessarily well-understood – between the problems in different tracks.

On the other hand, some authors [Culberson, 1998; Droste et al., 1999; Streeter, 2003] have argued that NFL theorems do not really apply to real-world optimization problem. Intuitively, such problems (more precisely their evaluation function) are represented by finite memory computer programs, which is a strong restriction compared to the set of problems considered in the NFL setting.

It is most interesting to note the correspondence between the NFL theorem implication and the algorithm selection framework[11] proposed by Rice [1976] two decades earlier. In this seminal work, Rice [1976] states that the choice of an algorithm critically depends on the problem instance at hand, and that the performance of an algorithm depending on the description of the problem instance can be learned. We shall return to per-instance algorithm selection in Sec. 3.2.

[9] `http://www.satcompetition.org/`

[10] `https://bbcomp.ini.rub.de/`

[11] This framework is not limited to optimization algorithms but also includes, *e.g.* game strategies or scheduling algorithms. As a consequence, the previously mentioned NFL theorem cannot apply in this setting.

NFL theorems have since found their way into the maths and computer science folklore as a principle stating that "no magical approach works the best in all cases". Variants of the NFL theorem exist for other domains, *e.g.* machine learning [Wolpert, 1996, 2002], that may be applicable to per-instance algorithm selection (see above).

2.2 Strategies to reduce the optimization cost

As mentioned in Sec. 2.1.3, a major driving force in optimization algorithm design is its practical use, focusing on the trade-off between the quality of the returned optimum and the optimization cost. In an Optimization-As-A-Service perspective, the cost includes the development cost and the usage cost, while the value comes from the usage phase. For this reason, one cannot think of designing one algorithm for each problem instance: the holy grail is to design breoadly scoped methods – that is, easily adjusted to tackle new variants of problems. This section introduces the main optimization approaches aimed at a decent usage cost while being widely applicable and adaptable.

2.2.1 Metaheuristics

The need for efficient methods to tackle large problem instances has driven the development of heuristics and metaheuristics. Heuristics are approximate methods, *i.e.* algorithms that rely on a strategy to find shortcuts during the optimization at the cost of optimality guarantees (to draw a parallel, taking shortcuts may drive the search close to the final optimum address but on the other side of a one-way street). Their goal is to deliver "good" solutions, *i.e.* with evaluation of quality close to the optimal one, allowing the practitioner to choose among them and possibly take into account extra implicit criteria [Michalewicz et al., 1996]. Metaheuristics are informally defined as "methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space" [Gendreau and Potvin, 2010]. Along this line metaheuristics are adaptable to a wide range of problems with moderate modifications; this explains why metaheuristics are widely used in application fields such as operations research [Gendreau and Potvin, 2005].

Metaheuristics often take inspiration from natural processes: evolutionary algorithms, simulated annealing, particle swarm optimization respectively take inspiration from biology, physics and ethology. Their strategies are usually designed to avoid stagnation in local optima, through the use of population of candidate solutions or mechanisms to accept a deterioration in the current solution. Such strategies are often controlled by hyperparameters (*e.g.* the initial population size, or the temperature parameter controlling the deterioration acceptance rate). These parameters control the trade-off between exploration (favored by large and scattered populations, high deterioration acceptation rates) and exploitation (favored by small populations, low deterioration acceptance rates), possibly dynamically (when the hyperparameters evolve during the run as part of the strategy[12]).

2.2.2 Hybrid metaheuristics

Besides simple metaheuristics there exist hybrid approaches that combine different metaheuristics, or metaheuristics with exact methods. Such approaches aim to exploit the

[12]The classical exploration vs. exploitation dilemma in machine learning is often called diversification vs. intensification dilemma in the metaheuristics community.

complementary strengths of different algorithms. The simplest example thereof is the well-known restart strategy, where a local search method is used repeatedly with different random initializations. Talbi [2009] proposes a hierarchical classification of hybrid metaheuristics (Fig. 2.1):
The first distinction holds between *low-level* and *high-level* hybrids. In the low-level class, hybridization takes place within a given metaheuristics, borrowing operators from another meta-heuristics. In the high-level class, several metaheuristics cooperate and exchange information.
The second distinction regards the cooperation between metaheuristics. In *relay* hybrids, the different metaheuristics are organized in a pipeline, the output of a metaheuristics being used as the input of the next one. In *teamwork* hybrids, they are launched in parallel.

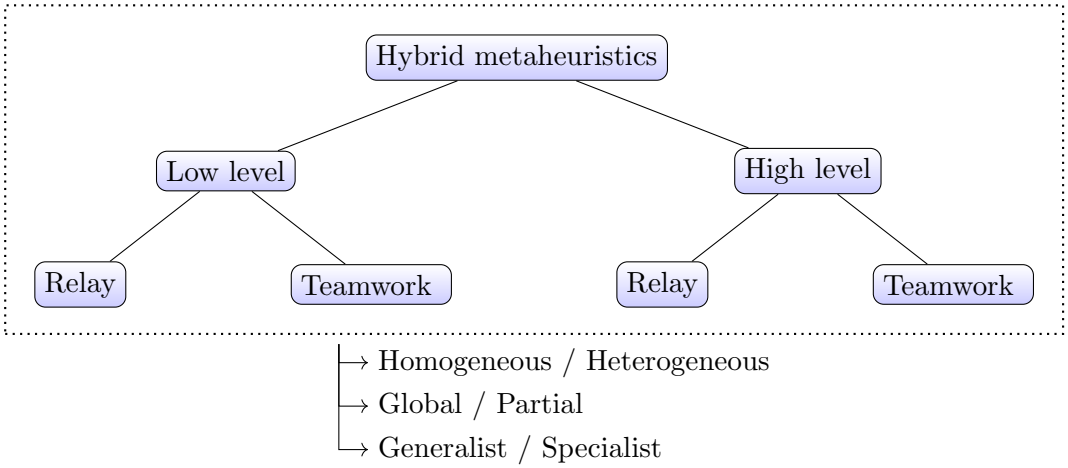

Figure 2.1: A hierarchical metaheuristics taxonomy, as proposed in [Talbi, 2009]

Additional traits are defined by Talbi [2009] to categorize hybrid metaheuristics.

Homogeneous vs. heterogeneous. Homogeneous hybrids combine several instances of the same metaheuristics, with different initializations (*e.g.* restart strategy), hyperparameters or component operators (*e.g.* different neighborhood). Heterogeneous hybrids combine different metaheuristics to take advantage of their respective strengths (*e.g.* combine the exploration properties of an ant colony optimization algorithm and the exploitation properties of local search [Stützle and Hoos, 1997]).

Global vs. partial. In global hybrids, all metaheuristics work on the entire search space. In contrast, metaheuristics components of a partial hybrid tackle (linked) subproblems with a reduced search space; they deliver partial solutions which are reconciled to deliver a global solution. Partial hybrids are thus highly related to decomposition techniques. A simple application of a partial hybrid is the case of a separable objective function $f(x, y) = f_1(x) \cdot f_2(y)$, in which case the optimization of x and y can be tackled independently (e.g., if f_1 and f_2 are both positive) or not (if f_1 and f_2 both may take positive and negative values).

Generalist vs. specialist. In generalist hybrids, all component algorithms solve the same target optimization problem, sharing information or competing against each other.

Specialist hybrids combine metaheuristics with different focus, *e.g.* one maximizing the exploration and the other that refines the solutions of the former one to find local minima. Interestingly, the objectives of the different metaheuristics may take into consideration aspects of the optimization problem which have not been explicitly modeled in the global objective, *e.g.* running time or properties of the returned solution.[13]

For a more comprehensive review of hybrid metaheuristics and examples, the interested reader is referred to [Talbi, 2009], Chapter 5.

2.2.3 Algorithm portfolios

Algorithm portfolios – which will be formally introduced in Chap. 3 – can be situated with respect to the above metaheuristics classification as follows:
Firstly, they combine several algorithms that share the same purpose; their goal may be to deliver a better result in terms of solution quality – be it because the portfolio algorithms are all non-exact methods or because the actual interest of the optimization practitioner does not match exactly the optimization problem criterion.[14] Another aspect typically tackled by algorithm portfolios is the optimization cost, typically in terms of runtime.

Secondly, w.r.t. the classification traits proposed in [Talbi, 2009], algorithm portfolios are: **high-level** (all component algorithms are left unchanged); and **global** (all component algorithms explore the same search space). Note that portfolio algorithms do not share their current solutions (or set of solutions) in the general case, neither sequentially nor in parallel.

Tab. 2.1 summarizes the positioning of algorithm portfolios variants:

Algorithm portfolio variant	**Properties (when applicable)**
Parallel scheduling	Teamwork, heterogeneous, generalist
Sequential scheduling	Relay, heterogeneous, generalist
Per-instance algorithm selection	Relay, heterogeneous, specialist
Per-instance algorithm configuration	Relay, homogeneous, specialist

Table 2.1: Characteristics of most prominent algorithm portfolios, seen as metaheuristics. For sequential scheduling (resp. per-instance algorithm selection, per-instance algorithm configuration), relay occurs between the selected components (resp. between the selector and the selected component, between the configurator and the configured algorithm).

Algorithm portfolios can be seen a a special case of hyper-heuristics; the interested reader is referred to [Burke et al., 2010] and references therein for a classification of hyper-heuristics.

2.2.4 Surrogate modeling-based optimization

When dealing with expensive optimization objectives, an approximate computationally efficient model of the objective, referred to as meta-model or surrogate model, is learned

[13] Alternatively, the global optimization problem can be stated as multiobjective to integrate such targets.

[14] For instance, in machine learning *training* defines an optimization problem on known data while machine learners are interested in the *trained* system performances on new data (which is in essentially a different problem). Besides, the underlying goal (*e.g.* classification accuracy, ranking-based indicators) often pertains to combinatorial optimization, which is hardly tractable given the high-dimensional search space; therefore, optimization is performed for a slightly different but more tractable optimization problem (*e.g.* using cross-entropy as the objective function).

and used *in lieu* of the true objective to save time. The surrogate model is typically built by regression from a few samples, *i.e.* candidate solutions for which the true objective has been computed. The number of initial samples as well as the choice of the regression approach and its hyper-parameters govern the efficiency of the surrogate optimization approach. They are often selected using expert knowledge about the system (*e.g.* uni- or multi-modality w.r.t. the parameters), while the initial samples are determined using standard Design of Experiment (DoE) approaches Fisher [1937].

Surrogate-based optimization proceeds by alternatively calling the surrogate model, and the expensive true objective, to estimate the objective value and/or constraints for new candidate solutions. Such an approach is typically used when surrogate-modeling is inexpensive and the evaluation of the surrogate model is negligible w.r.t. the true one (*e.g.* when a candidate solution is evaluated through an expensive simulator while the surrogate-model is an analytic function). The cost of surrogate optimization only takes into account the calls to the true objective function; the calls to the surrogate model are "for free".

Surrogate model-based optimization requires the surrogate model to accurately reflect the true objective, though inaccuracies in bad or unpromising regions do not harm. Accordingly, most surrogate-based approaches build an initial DoE of limited size and iteratively update the surrogate model to improve its accuracy in the regions of interest. In this perspective, surrogate-based optimization connects to Bayesian Optimization Hennig and Schuler [2012], where the surrogate model and its uncertainty are directly exploited to determine the best points to evaluate.

2.3 Summary

Optimization has evolved substantially in the last decades to benefit from fast computation to tackle ever more diverse and complex problems. Computational effort also compensates to some extent for the lack of knowledge about the problems to solve; would such knowledge be available, the user would often be able to solve the problem instances in an efficient manner through dedicated algorithms.

A few scientific communities have been built around categories of optimization problems in order to better understand their specifics and devise approaches taking advantage of these. Pursuing the same line introduced in [Wolpert and Macready, 1997] – and even though the applicability of the celebrated No Free Lunch theorems is limited as far as real-world problems are concerned – such "divide and conquer" strategies can be viewed as a mean to incorporate the specificity of different distributions of problem instances into the algorithms design. algorithms design.

In the meanwhile, much effort has also been put to provide optimization strategies that make as few assumptions as possible on the problem at hand, defining the so-called Black-Box Optimization (BBO) setting; naturally, BBO approaches are hardly the most appropriate ones to well-behaved optimization problems. Most generally, the design of optimization algorithms aims at a good balance between the efficiency of the brand new algorithm on a niche of problems, and the width of this niche, – in other words, a good balance between *efficiency* vs. *applicability*.

Still, quite a few real-world problems are both hard to characterize and too complex to be tackled using pure black-box optimization (*e.g.* for scalability reasons). Such problems have called for the deployment of approximate methods that combine efficient search strategies with wide applicability. These approximate methods have been further extended to include hybrid meta-heuristics. The rapid evolution of the optimization field sees the

come-back of old methods, and their use as ingredients of more complex strategies, facing two limitations. The first limitation concerns the cooperation of the previously mentioned ingredients. Another one is the substantial computational resources needed for such meta-heuristics.

Overall, in the computational setting considered in this thesis, significant expertise – regarding the problem at hand and its properties on the one hand, and the algorithms on the other hand – is required to deliver (quasi) peak performances on every problem instance. How to automatically build such an expertise is the bottleneck of algorithm selection.

Chapter 3

Algorithm portfolio techniques for optimization

Advances in optimization in the last two decades have shown that peak performances can be delivered by exploiting the variety and complementarity of already existing techniques. This chapter presents some prominent algorithm portfolio approaches and discusses their impact on optimization communities. Recent advances in portfolios are then reviewed, and lessons to learn are proposed.

3.1 Introduction to algorithm portfolios

Algorithm portfolios appear in domains where there exists no universal algorithm, in the sense of the No Free Lunch theorem [Wolpert and Macready, 1997] (Sec. 2.1.5). In such cases, the portfolios include diverse algorithms aimed at the same task, possibly developed along different perspectives, by different teams and with different applications in mind. The algorithm diversity is desirable: if they have complementary strengths, their ensemble can achieve better performances than each alone on a wider range of problems.

Our main focus here is on optimization portfolios in the black-box optimization setting. Each algorithm a achieves optimization on a problem instance x on a prescribed computational budget; the result is *a posteriori* assessed and defines the performance assigned to pair (a,x). Only the single-objective optimization task is considered in the following. Accordingly, the performance measure usually indicates the quality of the eventual result or the resources needed to achieve optimization. In the following, the problem instances range in instance space $\mathcal{I}$ ($x \in \mathcal{I}$) and "solving x with a" will be used interchangeably with "performing the optimization of x using a until it stops or resources are exceeded". Note that this framework can be extended to the multi-objective optimization setting whenever a (scalar) performance measure has been defined.

3.1.1 Per-Instance Algorithm Selection

The Algorithm Selection (AS) problem, first formalized by Rice in his seminal work [Rice, 1976], consists in a mapping from problem space $\mathcal{I}$ to algorithm space $\mathcal{A}$. A more involved setting includes the extraction of problem instance features. These features support the *per-instance* algorithm selection (PIAS, Fig. 3.1) defining for each instance x which algorithm to use. As noted by Rice [1976], algorithm selection occurs in various contexts, ranging from estimation to artificial intelligence. Note that PIAS also encompasses the selection of an algorithm for a distribution of problem instances (*i.e.* a constant mapping).

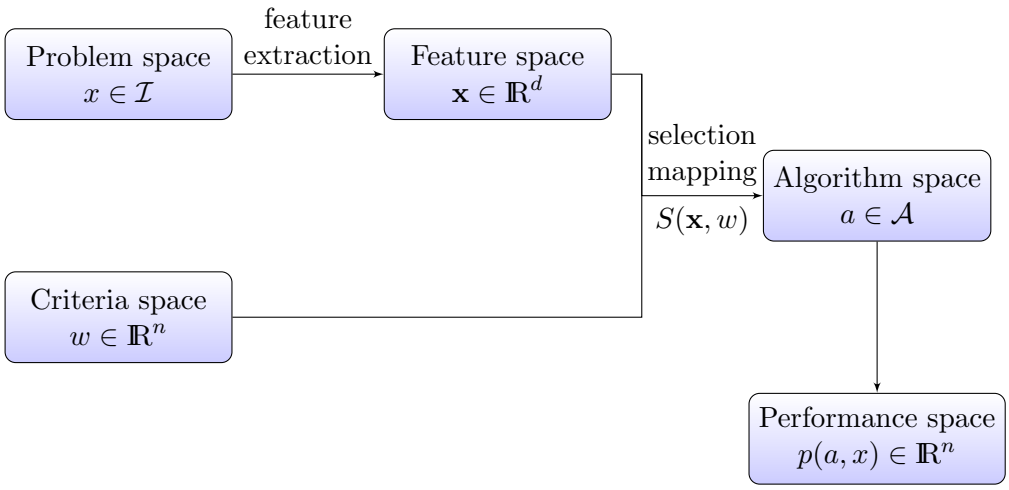

Figure 3.1: The original formulation of the Algorithm Selection problem in [Rice, 1976].

Definition 3.1 (Single best algorithm, single best selector)**.** Let $\mathcal{A}$ be a set of algorithms and $\mathcal{I}$ a set of problem instances. Given a performance measure $p : \mathcal{A} \times \mathcal{I} \mapsto \mathbb{R}$ (w.l.o.g., we assume p is to be maximized), the algorithm $a^* \in \mathcal{A}$ that performs the best on average on all $x \in \mathcal{I}$ is called the *single best algorithm* on $\mathcal{I}$.

$$\forall a \in \mathcal{A},\ \|p(a^*, x)\|_{\mathcal{I}} \geq \|p(a, x)\|_{\mathcal{I}}$$

The single best algorithm defines a constant selection algorithm, referred to as *single best selector*:

$$\begin{aligned} \mathcal{I} &\mapsto \mathcal{A} \\ x &\to a^* \end{aligned}$$

Typically, benchmarks on a (finite) set of instances are used to identify the single best algorithm (SB) (Def. 3.1) on that very set; let denote this set $\mathcal{I}_{SB}$, assuming for simplicity that SB is unique.[1] After SB has been identified from $\mathcal{I}_{SB}$, it is usually used to solve a wider set of instances $\mathcal{I}$. Naturally, there is no guarantee that SB determined from $\mathcal{I}_{SB}$ be the optimal choice for $\mathcal{I}$.

Using a single optimization algorithm to solve every problem instance is a most simple example of algorithm selector, though it is sub-optimal [Wolpert and Macready, 1997]. Research has thus focused on designing specialized algorithms, reflecting some decomposition of the optimization landscape according to general properties of the problem instances, or application domains. The selection mapping in that case thus boils down to determining the properties satisfied by x or the sub domain it belongs to, and run the corresponding SB. Such a rule-based approach however is coarse-grained: the SB of a subdomain might fail significantly on some problem instances in this sub-domain, as pointed out by Leyton-Brown et al. [2003]. Furthermore, the fragmentation of the optimization landscape and the design of algorithms specific to each niche reach their limit at some point: the value of an algorithm also depends on the breadth of its niche.

PIAS addresses a more flexible setting where some information is known *a priori* on instance x, and used to guide the selection – like optimization experts traditionally make their decision, relying on their experience of the algorithms. Particular attention must thus be paid to the characterization of problem instances, as the selection heavily relies on it (more in Sec. 3.2.2).

[1] Though there is no guarantee thereof.

The optimal decision in PIAS is referred to as *oracle selector* (Def. 3.2), also called *virtual best algorithm.*

Definition 3.2 (Oracle selector)**.** Let $\mathcal{A}$ be a set of algorithms and $\mathcal{I}$ a set of problem instances. Given a performance measure $p : \mathcal{A} \times \mathcal{I} \mapsto \mathbb{R}$ (w.l.o.g., we assume p is to be maximized), the *oracle selector*, denoted S^* is a per-instance selector that achieves optimal decision for all instances $x \in \mathcal{I}$.

$$\forall (x, a) \in \mathcal{I} \times \mathcal{A},\ p(S^*(x), x) \geq p(a, x)$$

On the algorithm side, an algorithm portfolio includes a usually small number of algorithms. Though it is often easy to identify good algorithms in fields where competitions are organized on a regular basis (more in Sec. 3.3.1), a number of questions remain for the practitioner:

- Are the algorithm implementations available?
- Are the algorithms applicable to the problem at hand?
- What are the algorithms intended for? Algorithms taking part in a competition may be designed for the distribution of problem instances in past editions of the challenge considered or for certain tracks in the challenge.
- Are the algorithms complementary? Making several algorithms cooperate in a portfolio is more profitable if they have negatively correlated performances [Huberman et al., 1997][2], *i.e.* if one algorithm excels at solving instances that are precisely hardly handled by the others.

The above questions must be addressed before designing a PIAS. As highlighted in Sec. 4.2.1, the diversity of algorithms within the portfolio is one of the key aspects to keep in mind.

Finally, Kotthoff [2016] notices that PIAS only answers the question "what" algorithm should be run but not "why". As in AI in general, explainable decisions would be very profitable to the community, to better understand the niche of existing algorithms and foster new algorithm design.

3.1.2 Parallel portfolios

Parallel portfolios allow several algorithms to run concurrently and independently, until one algorithm solves the problem instance or the computational budget is exceeded. Such an approach has first been studied with several instances of a single stochastic algorithm, in problems in the domain of combinatorial optimization.[3] Huberman et al. [1997] discuss the sharing of the computer time (illustrated with two instances of the same algorithm) from an economics perspective: maximizing the value (here, minimizing the expected runtime) while minimizing the risk (the variance of runtime over the set of optimization problems they consider). This works experimentally reveals the existence of an "efficient frontier" – *i.e.* a set of Pareto solutions of the value-risk optimization problem – for graph-coloring problem instances that improves both in terms of expected runtime and risk on the single algorithm setup.

Gomes and Selman [2001] present the first experiments with several *different* algorithms for solving a given instance of a specific domain of finite algebra (the quasigroup

[2]It is often sufficient that algorithms have uncorrelated performances.

[3]However, parallel portfolios can be used for any complex reasoning application.

problem). Three settings are explored: exploiting the parallel architecture of modern computers to run several algorithms (or several instances of the same stochastic algorithm) at the same time (each instance of algorithm runs on a dedicated processor); a setting close to the one used in [Huberman et al., 1997] where algorithms are run interleaved on a single processor (each algorithm gets the same computing resources); a single algorithm and single processor restart strategy[4]. For the first two settings, the number of dedicated processors (here equal to the number of component algorithms) governs the design of the optimal portfolio. For algorithm instances running on dedicated processors, running only instances of the algorithm that dominates the others when the time budget is low turns out to be the best portfolio design in terms of time-to-solution as well as risk when many processors are available. Quite the opposite, when only 2 processors are available, it is best to run only copies of the best algorithm for long running time. In between, a phase transition exists where it is best to use a mix of both algorithms. When several algorithms should share a single processor, it is beneficial to use an intermediate number of mixed copies, suggesting that the chances for one algorithm to solve the problem quickly no longer counterbalance the additional cost of running other algorithms interleaved when the number of copies grows.

It must be highlighted that these pioneering studies present experiments limited to one or a few optimization problem instances, where the expected runtime and risk are affected by the randomization induced by the algorithm more than on the variability of problem instances within the considered distribution of instances.

Parallel portfolios are formalized in [Sayag et al., 2006] as a *resource sharing* model and the authors draw the parallel with the *parallel unrelated machines model of job scheduling*, with the notable difference that the total cost of the schedule is not linear with respect to the *shares* variables. Sayag et al. [2006] propose an algorithm to find an optimal resource sharing schedule for a fixed number of algorithms.

3.1.3 Algorithm schedule

Algorithm schedules have been extensively investigated in the domain of constraint satisfaction [Sayag et al., 2006; Kadioglu et al., 2011; Streeter and Smith, 2012]. A main difference is that the focus is on the actual time required to find a solution, as opposed to the quality of the solution found on a computational budget in optimization: this is typical in decision problems[5]. algorithm schedules are defined as follows, after [Streeter and Smith, 2012].

Definition 3.3 (Algorithm schedule)**.** An *algorithm schedule* or *sequential portfolio* of algorithms consists of a sequence of $(a_k, \tau_k) \in \mathcal{A} \times \mathbb{R}^+$ pairs (called "steps") such that each instance is successively tackled by algorithm a_k with a computational budget τ_k, until being solved or the end of the schedule is reached. An algorithm schedule of finite size K is denoted $((a_1, \tau_1), \ldots, (a_K, \tau_K))$.

Suppose that instance i is solved by an algorithm a in $t_{ia} \in \mathbb{R}^+ \cup \{+\infty\}$. For practical reasons, a maximal runtime allowed is fixed to T^{max} (called timeout in the following). Let k_i^* denote the index of the first step that succeeds in solving instance $\mathbf{x}_i$:

$$k_i^* = \begin{cases} \min\{k | \tau_k \leq t_{ia_k}\} \text{ if } \{k | \tau_k \leq t_{ia_k}\} \neq \varnothing \\ K + 1 \text{ otherwise} \end{cases} \tag{3.1}$$

[4]This last setting is outside of the parallel portfolios scope and will be discussed in Sec. 3.1.3

[5]A decision problem can often be cast as an optimization problem and vice versa. For consistency, let us assume that we would consider the optimization variant of a decision problem.

The runtime needed for the schedule $S = \{(a_k, \tau_k)\}_k$ to solve an instance $\mathbf{x}_i$ is then given by:

$$r_t(\mathbf{x}_i, S) = \begin{cases} \sum_{k=1}^{k_i^*-1} \tau_{a_k} + t_{k_i^*} \text{ if } k_i^* \leq K \\ T^{max} \text{ otherwise} \end{cases} \tag{3.2}$$

The Penalized Average Runtime with penalization factor 10 ($PAR10$ score) is defined on a set of instances as the average of the runtimes to solve the instance, where the time associated with an unsolved instance is 10 times the timeout. The elementary evaluation on a single instance, denoted $PR10$ of a schedule S is computed as:

$$PR10(\mathbf{x}_i, S) = \begin{cases} \sum_{k=1}^{k_i^*-1} \tau_{a_k} + t_{k_i^*} \text{ if } k_i^* \leq K \\ 10 \cdot T^{max} \text{ otherwise} \end{cases} \tag{3.3}$$

The indicator function n_s reports whether instance $\mathbf{x}_i$ is solved by schedule S:

$$n_s(\mathbf{x}_i, S) = \begin{cases} 1 \text{ if } k_i^* \leq K \\ 0 \text{ otherwise} \end{cases} \tag{3.4}$$

The black-box algorithm context implies that if the same algorithm appears twice in the schedule, it will start again from scratch for the second run.[6]

The celebrated *restart strategy* is a particular case of algorithm schedule where all component algorithms are instances of a single algorithm with different initial states. Whereas it was originally designed for local search algorithms to escape local optima[7], it has been proven to be effective for solving quicker and a larger number of hard real-world problems when combined with specific algorithms [Gomes et al., 1998].

If the distribution of instances is known, the optimal schedule typically aims to maximize the number of instances solved in expectation [Hoos et al., 2015], where a schedule is defined by the algorithm ordering and the runtime allocated to each algorithm. Naturally, even the optimal schedule is dominated by the oracle selector, giving the entire available time budget to the best algorithm for every particular instance. In practice however, schedules seem to be more robust than (sub-optimal) algorithm selectors, making rare but often costly mistakes; additionally, schedules can advantageously be combined with algorithm selectors.

In practice, the distribution of instances is often unknown. It is commonly approximated through generating random instances, or gathering corpora of real-world instances, as done by challenge organizers (*e.g.* in SAT competitions[8], black-box optimization competition[9]) or by algorithm designers themselves. These corpora provide a basis for machine learning-based algorithm selection and for schedule optimization.

[6] Sayag et al. [2006] instead consider *task switching* schedules where algorithm runs can be stopped and resumed. Besides, runtimes are divided into units. Therefore they can define the runtime measure recursively in the time t.

[7] This argument comes from [Gomes and Selman, 2001]; the use of restart schemes can be found in much older publications but its origin remains unclear.

[8] http://www.satcompetition.org/

[9] https://bbcomp.ini.rub.de/

3.1.4 Evaluation measures

Evaluation is traditionally carried out on a test set denoted $\mathcal{I}_{test}$, disjoint from the set of instances used to construct a PIAS or a scheduler. Average measures are derived from r_t, $PR10$ and n_s on the test set, *e.g.* p_s the proportion of instances solved by a sequential schedule S or the $PAR10$ score:

$$p_s(S) = \frac{1}{|\mathcal{I}_{test}|} \sum_{\mathbf{x}_i \in \mathcal{I}_{test}} n_s(\mathbf{x}_i, S) \tag{3.5}$$

$$PAR10(S) = \frac{1}{|\mathcal{I}_{test}|} \sum_{\mathbf{x}_i \in \mathcal{I}_{test}} PR10(\mathbf{x}_i, S) \tag{3.6}$$

Normalized measures have been introduced to reflect the benefits of portfolios over the SB selector strategy and w.r.t. the oracle selector (best achievable portfolio given the set of algorithms). Following [Kotthoff, 2015] (Eq. 1), the general form of normalized measures is given by:

$$\overline{\mathcal{F}}(S) = \frac{\mathcal{F}(S) - \mathcal{F}(oracle)}{\mathcal{F}(SB) - \mathcal{F}(oracle)} \tag{3.7}$$

where $\mathcal{F}$ is a measure on $\mathcal{I}_{test}$, *e.g.* p_s or $PAR10$. The oracle scores 0 and the SB strategy scores 1 whatever performance indicator is considered. Besides, normalized measures allow to aggregate the measures of S over several datasets.

3.2 AS literature review

This section presents the basic concepts related to PIAS. The interested reader is referred to [Kotthoff, 2016] and [Smith-Miles, 2009] for a comprehensive survey of PIAS for combinatorial optimization, and AS. A comprehensive and regularly updated literature summary is maintained by Lars Kotthoff at `https://larskotthoff.github.io/assurvey/`.

3.2.1 Cold-start

PIAS can be cast as a recommendation problem, as popularized by the Netflix challenge Bennett et al. [2007]. In the Netflix problem, the goal is to recommend an item (movie) to a user, exploiting the movie ratings made by the user community. In the PIAS setting, the users are problem instances, the items are algorithms, and the goal is to recommend items (algorithms) to users (problem instances). The user (problem instance) "prefers" items (algorithms) that perform better on this particular instance.

A main difference between PIAS and recommendation systems lies in the fact that PIAS mostly tackles the cold-start problem [Schein et al., 2002], where the recommendation targets a brand new user (problem instance).[10] Indeed, there is no need to recommend an algorithm if the problem instance has already been solved in a practical context. There exist other cold-start settings (*e.g.* recommending brand new items to known users, Sec. 1.1.2) which are mostly irrelevant to the PIAS context (Sec. 1.2.1).

[10]Though some authors use a broader definition where items that received *few* recommendations fall into the cold-start setting, only the strict definition of cold start is considered in the following.

3.2.2 Feature extraction for PIAS

Per-instance AS strategies require informative features to describe the problem instances and determine which algorithm to run. As noted in [Rice, 1976], designing such features is itself a research question. On the one hand, features are domain-dependant; on the other hand, their design requires considerable expertise as relevant features are meant to capture the algorithm behaviors on problem instances. Additionally, these features must be computed or estimated cheaply [Mersmann et al., 2011]. As a consequence, computationally expensive features are sometimes discarded in order to save resources for the actual problem solving (*e.g.* feature cost prediction in `SATzilla` may cause the system to choose not to compute them [Xu et al., 2012b], see Sec. 3.3.1).

Feature sets include manually-specified features [Mersmann et al., 2010] and automatically computed ones; see, *e.g.* [Nudelman et al., 2004b; Mersmann et al., 2011]. Kotthoff [2016] proposes a classification of the descriptive features, as follows:

Static features They are obtained by *offline* computation, directly from the problem formulation or by using algorithms which only require light computation. Static features do not require to start solving the problem instance at hand but, of course, incur a computational burden beforehand.

Dynamic features They require to start solving the new instance at hand through the use of preconditioners, presolvers or actual algorithms. Dynamic features deliver information specific to the new instance in an online manner, allowing the PIAS to make better informed decisions. Since they are computed while an algorithm is running, the computational budget is shared between instance solving and feature computation, defining a trade-off between exploitation of the current algorithm and exploration.

Probing features An intermediate approach aims to estimate *offline* the performances of algorithms on the new instance, *e.g.* through running algorithms on a subset of the search space (the search space is "probed") that is thought to be representative of the whole space, or running simple algorithms [Pfahringer et al., 2000]. As the search space is partially explored, probing feature computation may actually solve the problem in hand. Exploratory landscape analysis is an area of research that focuses on the estimate of properties of an unseen instance using probing features. In continuous optimization for instance, running a hill-climbing algorithm from a set of initial points randomly sampled from the search space informs about the number of local minima the fitness landscape has. As these features require call to the fitness function for the sole goal of exploration, only those that can be computed or estimated cheaply are relevant to PIAS [Mersmann et al., 2011].

3.2.3 Algorithm performance prediction

Performance prediction lies at the crossroad of characterizing the hardness of a problem instance, and analyzing an algorithm empirical behavior. As said in Sec. 3.1, Rice [1976] originally based AS upon performance models. Algorithm performance prediction estimates *a priori* the potential outcomes of an algorithm on a given instance, depending on the application domain:

- Will the algorithm terminate within the allocated time?
- Will the algorithm return a solution of a given quality?

- How long will the algorithm run?

Algorithm prediction and its relation to (domain-dependent) features in optimization communities are very active research topics; the interested reader is referred to [Leyton-Brown et al., 2002] for combinatorial auctions, [Nudelman et al., 2004b] for SAT and [Muñoz et al., 2012] for black-box optimization, and references therein.

3.2.4 Classical algorithm selection techniques

Regression-based AS

AS can be achieved based on empirical performance models (EPMs). Many successful approaches ([Xu et al., 2008] and previous versions of `SATzilla`, [Nikolić et al., 2013], [Collautti et al., 2013] as a feature preprocessing step, [Gonard et al., 2016]) build such EPMs using Machine Learning algorithms, and more specifically regression algorithms Bishop [2006]. The training set associated with one portfolio algorithm is made of samples (x_i, y_i) where each x_i corresponds to the feature description of a problem instance and y_i the performance of the portfolio algorithm on this instance. From this training set is learned offline one regressor model, the EPM for this algorithm. For any new problem instance, its feature description $\mathbf{x}$ is computed and is passed to the performance models $\mathcal{G}_a$ (Fig. 3.2). If the performance models generalize well, the performance predictions are accurate and selection is obtained by simply choosing the algorithm with best predicted performances.

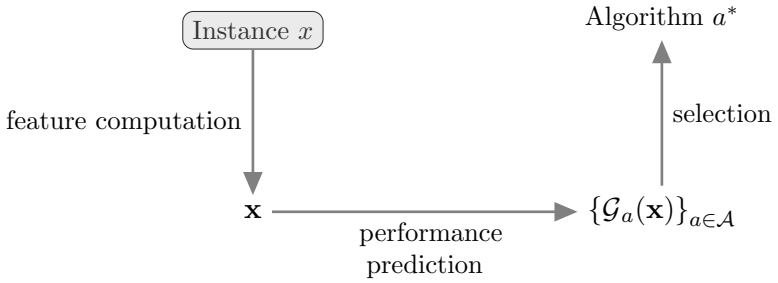

Figure 3.2: Regression-based AS for a new instance.

The simplest approach consists in predicting a global performance score, such as runtime. Other approaches consider a monotonous transform thereof, *e.g.* rescaling and centering *per instance* the performances (with zero mean and unit variance) [Collautti et al., 2013]. Such a preprocessing is expected to facilitate EPM learning and better *separating* solvers, thus supporting a more accurate selection decision.

This approach faces two difficulties. The former one regards the constitution of the training database, ideally running every algorithm on every problem instance until completion.[11] For computational tractability, however, any algorithm is run until the *cutoff* time is reached, causing "holes" in the database. Instances that can be solved by no algorithm on the computational budget are usually discarded. More delicate is the case when some algorithms could solve the instance and some could not, leading to missing information in the database; this information is hard to handle in classical regression algorithms and require imputation – which in turn is likely to add bias to the performance models.

[11] Note that the prediction of the best algorithm would only require to run all algorithms until one of them succeeds.

The second difficulty regards the accuracy and generality of the performance models, all the more so as the set of problem instances is restricted.

AS as a classification task

As the AS decision is to choose one algorithm among a few, AS can also be cast as a classification problem, where the class of an instance is associated with the algorithm that performs the best on it. A first advantage of classification compared to regression is the lesser computational cost needed to gather the training data, and the fact that no missing data is encountered. Another advantage is that the learned classifier will focus on what makes algorithms different (as opposed to explaining algorithm performance); this is desirable for AS as we are interested in selecting the best algorithm, be it for an instance easily handled by any algorithm or another one hard to solve.

On the other hand, if two algorithms a and b have equal performances, the instance should fall into both a and b classes. To comply with the standard classification setting, one out of both classes must be selected, biasing the training process. Along the same lines, the standard classification setting considers all errors as equal, although from an AS perspective, the cost of an error for a problem instance depends on the difference of performance between the desired and the predicted algorithm. This has drawn the research toward *cost-sensitive* classification (`SATzilla-2012` [Xu et al., 2012b], see below) or clustering (`CSHC` [Malitsky et al., 2013]).

Ranking-based AS

AS can also be cast as a *learning to rank* problem [Burges et al., 2005; Cao et al., 2007]. Such an approach takes into account a number of triplets $(x, a, b) \in \mathcal{I} \times \mathcal{A}^2$ and the relative order of a and b performances on problem instance x to build a preference model. The size of the problem (linear in the number of instances and quadratic in the number of algorithms) makes it usually tractable. However these approaches are not widely used, as standard learning-to-rank criteria do not take into account the cost of errors.

A notable exception is the recent work proposed in [Oentaryo et al., 2015], but the effectiveness of the approach is not assessed against runtime-related measures.

Pairwise regression based AS

Pairwise regression lies between regression and ranking. For each algorithm pair, this method aims to predict which one would perform the best on a new instance in a *cost-sensitive* way, *i.e.* the effect of a misclassification on the performances is integrated in the training loss. For instance, a set of regressors is used to predict the difference of performances for each pair of algorithms. Like classification, cost-sensitive pairwise regression does focus on what makes algorithm behaviors different but also takes into account to what extent their performances differ. On a new instance, trials between all pairs of algorithms are "predicted"; the actual selection comes with aggregating the results of the matches through vote, *e.g.* the algorithm that wins the highest number of duels is selected.

The fact that `SATzilla` switched from a performance regression-based to a pairwise regression based approach with `SATzilla-2012` [Xu et al., 2012b] is a strong clue in favor of this method. Other empirical evaluations (*e.g.* in [Kotthoff, 2015]) have acknowledged pairwise regression as a very effective technique, beating AS based on performance regression models using the very same ML algorithms. A limitation of pairwise regression approaches however is their quadratic complexity in the number of algorithms in the portfolio.

AS as a recommendation problem

Stern et al. [2010] cast algorithm selection as a *collaborative expert* system where experts are algorithms. The "fit" between all algorithms and the feature-described instance is expressed in a latent space learned by a *collaborative filtering* method called `Matchbox`, based on an approximate message passing algorithm. As noted in [Mısır and Sebag, 2017], collaborative approaches can handle sparse databases (where the performances of each algorithm might be known on a small subsample of problem instances), thus alleviating the computational effort needed to constitute and extend the database. Besides, Mısır and Sebag [2017] extend the `Matchbox` approach to a non-linear mapping from the initial feature space to the latent space.

AS as a reinforcement learning problem

Gagliolo and Schmidhuber [2006] cast the online version of PIAS as a bandit problem, learning *online* which arm to pull next. Picking an arm is meant as "run algorithm a for next instance" (pure AS) or "use time allocation TA for next instance" (when building a portfolio where the arm allocates time shares to the algorithms). Degroote et al. [2016] extend the pure selection approach with *contextual bandits*, exploiting the instance feature description to make their choice.

3.3 Review of state-of-the-art systems

This section reviews most prominent algorithm portfolio systems in the last decade. The review is structured by family of approaches, chronologically presenting the refined versions from the initial scientific breakthrough to almost all-inclusive frameworks.[12] The analysis of these state-of-the-art systems gives insights into the main characteristics a world-class algorithm portfolio should feature.

3.3.1 State-of-the-art optimization algorithm portfolios

SATzilla `SATzilla` [Nudelman et al., 2004a; Xu et al., 2007, 2008, 2012b,a] has a long history of successes in competition since the 2000s: it has first incorporated a portfolio strategy for the SAT 2003 competition [Le Berre and Simon, 2003]. `SATzilla` won various competition tracks (in majority in SAT competitions, but also in algorithm selection) and has had a considerable impact on the deployment of algorithm portfolio techniques in algorithm design. The latest version dedicated to SAT solving (`SATzilla-2012` [Xu et al., 2012b]) is described below. `SATzilla` includes a set of problem instances (used as training instances) $\mathcal{I}$, a set of candidate algorithms $\mathcal{A}$ and feature extractors, making it directly usable in a stand-alone setup.

`SATzilla` combines algorithm schedule techniques with algorithm selection. In a first phase, a sequence of algorithms – called presolvers – is run for a short amount of time to solve very easy instances. As SAT solvers usually have relatively uncorrelated performances, a combination of presolvers allows solving a large scope of problem instances. The order and budget given to each presolver is constant. In a second phase, the system then focuses on the remaining instances, called "harder instances", using per-instance

[12] Though related, algorithm configuration systems (see, *e.g.* [Belkhir, 2017] and references therein) and algorithm selection for machine learning (*e.g.* `AutoWEKA` [Thornton et al., 2013; Kotthoff et al., 2016]) are out of the scope of the present work and thus omitted.

algorithm selection. Before features are extracted from the instance at hand, a first classification model is run to predict whether the cost of computing features is affordable. If not, the SB algorithm (determined on the training "harder instances") is run; otherwise the solving process goes on with the feature computation. Selection is based on a collection of cost-sensitive binary classifiers between each pair of algorithms.[13] The classifiers are $|\mathcal{A}|\,(|\mathcal{A}|-1)/2$ random forests and their votes are aggregated to select the winner (algorithm that wins the highest number of binary trials).[14] An online mechanism finally switches the selected method to the second-best choice (resp. the SB) if the first (resp. the second) fails (*e.g.* crashes).

CP-Hydra [O'Mahony et al., 2008] is a solver developed for constraint programming. It uses a *case-based reasoning* approach, *i.e.* it retrieves relevant past *cases* most similar to the new instance and combine them into a schedule.[15] Each instance description comprises a set of static features and a set of probing features obtained by recording the modeling choices of a specific constraint solver; ratios are rescaled to [0,100] and quantified features are log-scaled. Retrieval is then achieved using a k-nearest neighbor (k-NN) approach in the instance description space. Finally, time shares are allocated to all algorithms, so that the resulting schedule solves the maximum number of retrieved instances, weighted by their similarity to the new instance.

3S and CSHC Kadioglu et al. [2011] propose `3S`, combining an optimized schedule of algorithms and an adaptative k-NN-based selector. The instance space is first partitioned and each training instance assigned to a cluster. Hyper-parameter k is learned for each cluster based on the training instances. At test time, the distance-weighted k-NN of the cluster relevant to the new instance is used for selection. The schedule is optimized as a linear programming problem, with the set of initial algorithms extended with 4 versions of the selector (such a schedule may thus depend on features if they incorporate one of the selectors; they are dubbed *semi-static schedules*). A last setting splits the available runtime: 90% is allocated to the selector while the remaining 10% are left for a static schedule for preventing the AS from overfitting the training instances.

`CSHC` [Malitsky et al., 2013] uses the same static schedule as `3S` for the first 10%. The selection component is based on hierarchical clustering where each cluster is assigned one algorithm: starting with a single one cluster, clusters are recursively splitted into two by axis-parallel (in the feature space) hyperplanes[16] such that when assigning a new algorithm to each subcluster, the sum of misclassification cost over all instances decreases. The final selector is built from bootstrap-aggregation of several such clusterings. `CSHC` has been shown to improve on `3S`, especially in terms of stability.

ArgoSmArT k-NN [Nikolić et al., 2013] is an instance of a dynamic portfolio. A new problem instance is solved by the algorithm that performs the best on its k nearest neighbors in the training instances. The selection is based on distance-weighted k-NN

[13] Misclassifying a pair of algorithms that perform almost equally has a low weight; misclassification if only one of the solvers actually succeeds in solving the instance has a high weight; see Sec. 3.2.4.

[14] This method is introduced in 3.2.4 as the regression pair model.

[15] As the performance measures the number of solved instances but not the time-to-solution, it only requires the sum of the runtimes to be less than the computational budget.

[16] Axis-parallel hyperplanes are preferred to maximum-margin hyperplanes as the second are considerably more expensive while they do not boost the performances. The splitting of clusters is achieved like in decision trees, but based on a different cost function.

runtime prediction models. Instead of the classical Euclidean distance the authors use:

$$d(x,y) = \sum_i \frac{|x_i - y_i|}{\sqrt{x_i y_i} + 1}$$

SNNAP (for Solver-based Nearest Neighbor for Algorithm Portfolios) [Collautti et al., 2013] relies on instance clustering with a modified set of features. The approach is inspired by `ISAC` (Instance-Specific Algorithm Configurator) [Kadioglu et al., 2010]: g-means clustering iteratively splits in two the current set of instances, such that the resulting two clusters better follow a Gaussian distribution than the former one, in feature space. The main drawbacks according to [Collautti et al., 2013] is that features are normalized – and thus all have the same weight for clustering, regardless of how important they are – and that the clustering objective does not necessarily translate into a performance improvement. To overcome this, `SNNAP` proceeds as follows: performance models (random forests) are trained to predict the centered runtimes (with 0 mean, unit variance for each training instance); for a new instance, the outcomes of these models replace the initial feature description for the selection phase. In a second step, k-NN is performed in the (centered performances) feature space to finally select the algorithm that behaves the best in the k-neighborhood of the new instance. It should be highlighted that predicting such scaled performances closely relates to predict an ordering on the performances of each algorithm on the new instance, and that the distance used for k-NN only takes into account the best performing (or predicted best performing) algorithms.

ME-ASP Developed for Answer Set Programming (ASP) by [Maratea et al., 2014], `ME-ASP` first computes incrementally a set of solvers until the last attempt to add a solver results in a portfolio not "sufficiently" distinguishable from the previous one. Selection is then achieved by a multi-label classification task (6 classifiers are evaluated: aggregation pheromone density based pattern classification, decision rules, decision trees, multinomial logistic regression, nearest-neighbor, support vector machine).

LLAMA [Kotthoff, 2013] is a flexible framework to compare AS strategies, parameterization and preprocessing steps, rather than a portfolio system. It is shipped as an `R` package.

aspeed [Hoos et al., 2015] defines an optimized static schedule. Optimization proceeds in two steps: first, the time budget is allocated to each algorithm so that the number of instances solved is maximized.[17] In a second step, the alignment of the algorithm is found to minimize the runtime. Both optimizations are performed using answer set programming. This sequential scheduling approach is then extended to multi-processing units working in parallel, where each algorithm is (additionally) assigned to a single unit. As it does not rely on domain-specific features, the `aspeed` approach is widely applicable to domains other than SAT.

claspfolio 2 and flexfolio. `claspfolio 2` [Hoos et al., 2014] integrates different feature generators, AS components, solving algorithms and presolvers (*i.e.* algorithms that are run at the beginning of the process to solve easy instances). It is based on the generation of a schedule of algorithms by `aspeed` and gives the possibility to produce an executable

[17] As a secondary criterion to decide between ties, the minimal L_2-norm of the vector of time budget is considered.

solver. It goes beyond the usual scope as it implements flexible software optimization [Hoos, 2012] and produces an executable, suitable for large audiences. Besides the core modules, it implements feature preprocessing and backup strategies. Another distinctive feature is the assessment of the selector component through cross-validation that can modulate the time budget given to the presolving schedule. The framework was further extended under the name `flexfolio`.[18]

SUNNY Originally developed for constraint programming [Amadini et al., 2014], `SUNNY` is an online approach that builds for the new instance the portfolio that solves the maximum number of instances in its k-neighborhood and with minimum size. All of these algorithms are used in the output schedule, where the time budget allocated to each component is proportional to the number of solved instances.

AutoFolio [Lindauer et al., 2015] takes a meta-AS perspective, using an algorithm configurator (SMAC [Hutter et al., 2011]) to configure an algorithm selector approach. Specifically, it involves two cross-validation schemes: the inner one assesses the selector within the current configuration, while the outer one assesses the configuration. The configuration space includes the type of selector (*e.g.* regression based approach, k-NN), the machine learning technique (*e.g.* ridge regression, support vector regression) and its hyperparameters (*e.g.* regularization coefficient for an SVM), the hyperparameters controlling the feature and performance preprocessing steps (*e.g.* feature selection, normalization parameters), and the presolving schedule parameters (*e.g.* number of solvers, budget to the presolver). Such a flexibility is achieved as a result of `AutoFolio` being based on the modular design of `claspfolio 2`. In a meta-AS perspective, `AutoFolio` is also a great tool to identify the relevant choices when designing an AS (*cf.* the analysis in [Lindauer et al., 2015]).

3.3.2 Lessons for portfolio design

Portfolio with "orthogonal" components

A main lesson regards the complementarity of the portfolio algorithms, regarded as a desirable property ever since [Huberman et al., 1997]: "the effect of cooperation, when manifested in negative correlations, is to increase the performance as well as reduce the risk". This claim has been validated later on, showing that: i) state-of-the-art algorithms are indeed often uncorrelated, and ii) their combination is worthwile in practice, as first demonstrated by [Leyton-Brown et al., 2003] and the first portfolio version of `SATzilla` [Nudelman et al., 2004a]. The diversity of a portfolio not only matters to algorithm selection; it is also crucial for schedules. Typically, if the two best algorithms have highly correlated performances, a successful schedule should retain one of them together with a not-so-good but more complementary algorithm.

Presolvers

The reader will have noticed that most systems described in Sec. 3.3.1 do not implement one of the straightforward approaches in Sec. 3.2 but rather appear as *hybrid* systems. The most successful approach consists in using a *global* portfolio (*i.e.* not instance-specific) as a so-called "presolver" together with an AS. This strategy is justified as a number of instances can be solved in no time by some algorithms; for these instances, a PIAS would

[18] `http://www.ml4aad.org/algorithm-selection/flexfolio/`

not bring substantial improvements. Presolvers are thus built as a schedule of a few algorithms designed to maximize the number of instances solved within a limited budget. The presolver does not rely on features, and can be run before feature extraction (thus saving the feature computation time in case the current instance is solved by the presolver). Acknowledging their importance, presolvers will be called **presolving schedule** and will be determined using a **pre-scheduler** in the remaining of this work.

Semi-static schedule

According to [Kadioglu et al., 2011], a *semi-static schedule* combines a static schedule (independent from the feature description of the instance) and an algorithm selected by an PIAS. Combining a pre-scheduler and a selector component is the most widespread example of semi-static schedule. An extension thereof includes the computation of features in the schedule and is especially relevant when feature computation is able to solve a couple of instances (*e.g.* feature computation that simplify the instance). Semi-static schedules thus define per-instance schedules that are more robust than simple PIAS and more efficient than static schedules. Their efficiency depends on the trade-off between the high-risk high-reward PIAS approach (all budget is given to the selected algorithm) and the more robust static schedules.

Feature preprocessing

Considerable effort has been expended by optimization communities to design features relevant to performance analysis (see, *e.g.* [Nudelman et al., 2004b] for SAT, [Maratea et al., 2014] for ASP, [Abell et al., 2013] for black-box optimization). Recent approaches also include various feature preprocessing techniques: feature selection, feature normalization (especially for k-NN based approaches), feature imputation or feature set extension, with two-fold motivations. Firstly, an extended feature set where the most influential features have been selected facilitate the AS task of detecting feature patterns that explain the algorithms performances. Secondly, feature selection also contributes to reducing the risk of overfitting. The most recent approaches consider feature preprocessing as modules or hyperparameters, to be configured along with the AS component. The most widespread technique for feature selection is feature filtering: statistics are computed to identify and eliminate redundant features or those that do not carry information relevant to, *e.g.* classification. At the opposite, forward feature selection considers an iteratively growing set of features (see *e.g.* [Hutter et al., 2013]).

Overfitting prevention

Due to the restricted size of the training datasets (typically a few hundred instances), overfitting is a main concern for AS designers. Overfitting is combatted by feature selection, as mentioned previously, and by using regularized learning criteria (regression for performance models, performance for schedule optimization) or ensemble methods (*e.g.* random forests). Another strategy relies on cross-validation; some systems jointly configure and train the selector component, with an embedded cross-validation mechanism to mitigate the risks of overfitting.

Hyperparameter tuning

As could have been expected, the efficiency of many options is domain-dependent; for instance a same feature filtering method – *chi squared* – leads from 10% improvement to

30% deterioration depending on the considered SAT dataset [Collautti et al., 2013]. For this reason, most recent systems are themselves subject to a careful manual tuning or come with automatic configuration mechanisms.

3.4 A library to assess them all: ASlib

The difficulty of comparing algorithm selection systems implies a high barrier of entry to the AS field. To alleviate this barrier a joint effort was undertaken to build the Algorithm Selection Library (ASlib), providing comprehensive publicly available resources to facilitate the design, sharing and comparison of AS systems [Bischl et al., 2016]. This section describes this library, referring the interested reader to [Bischl et al., 2016] for a more comprehensive presentation, including the context in which the ASlib resources were retrieved and the optimization domains it covers.

3.4.1 ASlib data format

ASlib (version 1.0.1) involves 13 datasets, also called scenarios (Tab. 3.1), gathered from recent challenges and surveys in the operations research, artificial intelligence and optimization fields. It includes all algorithm runs and instance feature values; however the actual algorithms and problem instances are not available. As such, it allows AS practitioners to *simulate* the algorithm runs (and feature computation) instantaneously, saving months of CPU-time computation. As a drawback, only the provided information can be used for an AS task, *i.e.* no other feature computed from the instances definitions can be used to extend the initial descriptions.

Dataset	# instances	# features (# groups)	# algorithms	Computation timeout (s)
ASP-POTASSCO	1294	138 (5)	11	600
CSP-2010	2024	86 (1)	2	5000
MAXSAT12-PMS	876	37 (1)	6	2100
PREMARSHALLING-ASTAR-2013	527	16 (1)	4	3600
PROTEUS-2014	4021	198 (4)	22	3600
QBF-2011	1368	46 (1)	5	3600
SAT11-HAND	296	115 (10)	15	5000
SAT11-INDU	300	115 (10)	18	5000
SAT11-RAND	600	115 (10)	9	5000
SAT12-ALL	1614	115 (10)	31	1200
SAT12-HAND	767	115 (10)	31	1200
SAT12-INDU	1167	115 (10)	31	1200
SAT12-RAND	1362	115 (10)	31	1200

Table 3.1: ASlib datasets (V. 1.0.1)

Each dataset includes:[19]

- the dataset computation time limit (called timeout).
- the performance and computational status of each algorithm on each problem instance. Performances are measured in seconds and are known for each (instance, algorithm) pair if the computation succeeded (tagged *ok*). Other (non-succeeding) computation status are: *timeout* (computation exceeded timeout), *memout* (computation exceeded permitted memory), *not_applicable* (the algorithm cannot be run on this instance), *crash* (the program failed to execute) and *other* (unexpected error).

[19]The full library format is described at: `https://github.com/coseal/aslib-spec/blob/master/format.md`

For all these cases, the performance of the (algorithm, instance) pair is assumed to be the dataset *timeout*.

- the arrangement of features into groups. It is mandatory to compute an entire group of features (also called a feature step) to access the value of any feature in the group.
- the description of each problem instance, as a vector of the expert-designed feature values (as said, this description considerably facilitates the comparison of the AS systems). Some values may be unknown (see below).
- the computational status of each group of features. Possible status are computation success (tagged *ok*), *timeout* (when it exceeds the timeout for this specific group of features), *memout*, *presolved* (instance was solved during feature computation), *crash*, *unknown* (feature computation was not run, probably because an earlier feature group computation presolved the instance) and *other* (unexpected error).
- (not mandatory) the computational cost (in seconds) of calculating each feature group.
- a 10 fold "official" cross validation to enforce the reproducibility of the 10 fold cross-validation assessment of every AS algorithm.

3.5 Summary

In the optimization domain, algorithm portfolios essentially rely on manually designed features describing the problem instances to deliver peak performances. Leveraging the peculiarities of the problem instance at hand is key to improving on the single best algorithm.

Most successful algorithm portfolios combine algorithm schedules and algorithm selection (AS) to deliver peak performances in a robust manner. algorithm schedules and AS are tackled as supervised machine learning problems, though some recent approaches take inspiration from recommender systems and reinforcement learning.

Feature and performance preprocessing are widely acknowledged to have a crucial impact on AS. Some recent attempts have been made to achieve automatic hyperparameter tuning, taking inspiration from the neighbor field of Algorithm Configuration. The extensive and multi-level use of machine learning approaches might however hinder the robustness of the results and increase the risk of overfitting.

Extensive data resources (instances and algorithm performance data) are now publicly available in standard libraries (ASlib for optimization and decision problems, OpenML[20] for machine learning), fostering the design and evaluation of new approaches and witnessing the maturity of the community.

[20] `https://www.openml.org/`

Chapter 4

ASAP: Algorithm Selector And Pre-scheduler

This chapter presents our contributions in the domain of algorithm portfolios, ASAP.V1 and ASAP.V2, building upon the lessons learned about algorithm portfolios (Chap. 3). The ASAP architectures are first presented and discussed. ASAP.V1 is then detailed together with its main behavioral design parameters. ASAP.V1 has been extensively and successfully assessed during the ICON challenge 2015 on algorithm selection [Kotthoff, 2015]. Its successor, ASAP.V2, follows the current trend in algorithm portfolio with a special fine-tuning of ASAP.V1 to enforce the cooperation between an algorithm sequential schedule and an algorithm selector. A detailed evaluation is provided to highlight the improvements of ASAP.V2 over ASAP.V1 and other state-of-the-art systems. ASAP.V2 won the Open Algorithm Selection Challenge 2017 [Kotthoff et al., 2017], demonstrating the efficiency of the approach.

This chapter is based on [Gonard et al., 2016]. It presents complementary experiments together with extended discussions and more comprehensive research perspectives.

4.1 Overview of ASAP

This section provides an overview of the ASAP system architecture and its motivations. Special attention is paid to the interactions between the different components during training.

4.1.1 Motivation

It is notorious that the hardness of a problem instance often depends on the considered algorithm. As shown on Fig. 4.1 for the SAT11-HAND dataset (Section 4.3), while several algorithms might solve 20% of the problem instances within seconds, the oracle (selecting the best one out of these algorithms for each problem instance) solves about 40% of the problem instances within seconds. Along this line, the pre-scheduler's task is to select a few algorithms, such that running each of these algorithms for a few seconds would solve a significant fraction of the problem instances; the resulting schedule is called a *presolving schedule* (Def. 4.1).

Definition 4.1 (Presolving schedule)**.** Let $\mathcal{A}$ be a set of algorithms. A K-component presolving schedule is defined as a sequence of K (algorithm a_k, time-out τ_k) pairs,

$$\left((a_k, \tau_k)_{k=1}^{K}\right) \text{ with } (a_k, \tau_k) \in \mathcal{A} \times \mathbb{R}^+, \ \forall k \in 1, \ldots, K$$

For each problem instance $\mathbf{x}$, it sequentially launches algorithm a_k on $\mathbf{x}$ until either a_k solves $\mathbf{x}$, or time τ_k is reached, or a_k stops without solving $\mathbf{x}$. If $\mathbf{x}$ has been solved, the execution stops. Otherwise, k is incremented while $k \leq K$.

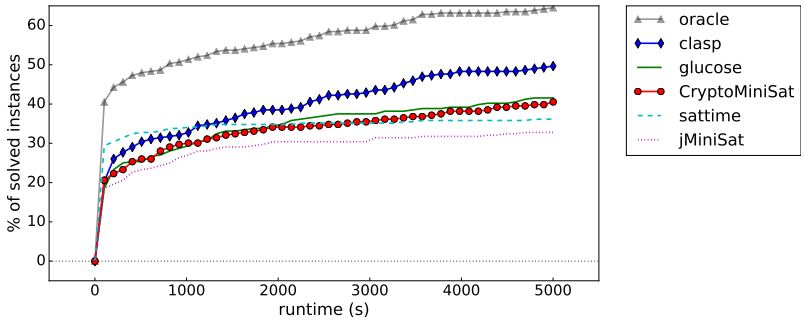

Figure 4.1: Percentage of solved instances vs. runtime on the SAT11-HAND dataset, for 5 algorithms and the oracle (selecting the best algorithm out of 5 for each problem instance).

Def. 4.1 is similar to that of an algorithm schedule (a.k.a. sequential portfolio) (Def. 3.3). The main difference in practice is that K is low (a typical value is 3) and the total length of a presolving schedule denoted T_{ps} is "small" in comparison with the dataset timeout denoted T^{max}:

$$T_{ps} = \sum_{k=1}^{K} \tau_k \ll T^{max}$$

A pre-scheduler can contribute to better peak performances [Lindauer et al., 2016]. It can also increase the overall robustness of the resolution process and mitigate the impact of PIAS failures (where the selected algorithm requires high computational resources to solve a problem instance or fails to solve it), as it increases the chance for each problem instance to be solved in no time, everything else being equal.

4.1.2 ASAP full picture

Accordingly, the ASAP system involves two components. A pre-scheduler aims at solving as many problem instances as possible in a first stage; a PIAS takes care of the presumably "harder" remaining instances. Formally, ASAP follows the overall schedule:

$$S_{ASAP} = ((a_1, \tau_1), \ldots, (a_K, \tau_K), (a_{AS}, \tau_{AS})) \tag{4.1}$$

where a_{AS} is the algorithm recommended by the PIAS and τ_{AS} the budget given to it.

Division of labor A first decision regards the division of labor between the two components, specifically how to split the available time budget between the two. It is clear that the number of problem instances solved by a module will increase with its computational budget, everything else being equal; however the pre-scheduler and the AS components are interdependent and the joint system performance should be considered as the main decision criterion. A second choice regards the number of algorithms in the presolving schedule (parameter K). For simplicity and tractability, the maximal runtime allocated to the pre-scheduler is fixed to T_{ps}^{max} (10% of T^{max} in the experiments, Section 4.4), and the number K of algorithms selected by the pre-scheduler is set to 3. `3S` [Kadioglu et al.,

2011] uses a closely related setup for the fixed-split selection schedules, except the presolving schedule phase in ASAP is *at most* T_{ps}^{max} and its actual duration is optimized on a per-dataset basis.

An alternating optimization process Given T_{ps}^{max} and K, ASAP tackles the optimization of the pre-scheduler and the AS components. As previously stated, both optimization problems are interdependent: the AS focuses on the problem instances which are not solved by the presolving schedule, while the pre-scheduler focuses on the problem instances which are most uncertain or badly identified by the AS. Formally, this interdependence is handled as follows:

1. An initial pre-scheduler PS_{init} builds the presolving schedule to optimize the number of instances solved over all training problem instances within a small computational budget.

2. A performance model $\mathcal{G}(\mathbf{x}, a)$ is built for each algorithm over all training problem instances, defining AS_{init} as a regression-based selector (Sec. 3.2.4). The successes and failures of the algorithms in PS_{init} are used as additional features (more in Sec. 4.2.3).

3. A fine-tuned presolving schedule is determined by pre-scheduler PS_{post} to optimize the joint performance (PS_{post}, AS_{init}) over all training problem instances; in comparison to PS_{init}, PS_{post} gets little reward for solving instances that AS_{init} solves quickly;

4. A second performance model $\mathcal{G}_2(\mathbf{x}, a)$ is built for each algorithm over all training problem instances, using additional features derived from PS_{post}; yielding AS_{post}.

The full architecture of ASAP is illustrated in Fig. 4.2, displaying the component interactions. Each component, including the crucial feature preprocessing phase for the PIAS component, is detailed in the following sections together with the rationale for the main design decisions. For simplicity, we use the same notation (PS_{init}, resp. PS_{post}) to denote both the pre-scheduler and the resulting presolving schedule in the remainder of this chapter.

4.2 ASAP.V1 components

This section details the pre-scheduler and AS modules forming the ASAP system, Version 1 (ASAP.V1). Experiments are conducted using the ASlib data and specifications (described in Sec. 3.4) to illustrate the effects of the main design parameters.

4.2.1 ASAP.V1 pre-scheduler

Description

Let $\text{PS}_{init} = (a_k, \tau_k)_{k=1}^K$ denote a presolving schedule, with overall computational budget $T_{ps} = \sum_{k=1}^K \tau_k$, and let $\mathcal{F}\left((a_k, \tau_k)_{k=1}^K\right)$ denote the associated domain-dependent performance.

ASAP.V1 considers for simplicity:

- equal timeouts:

$$\forall k \in \{1 \dots K\},\ \tau_k = \frac{T_{ps}}{K}$$

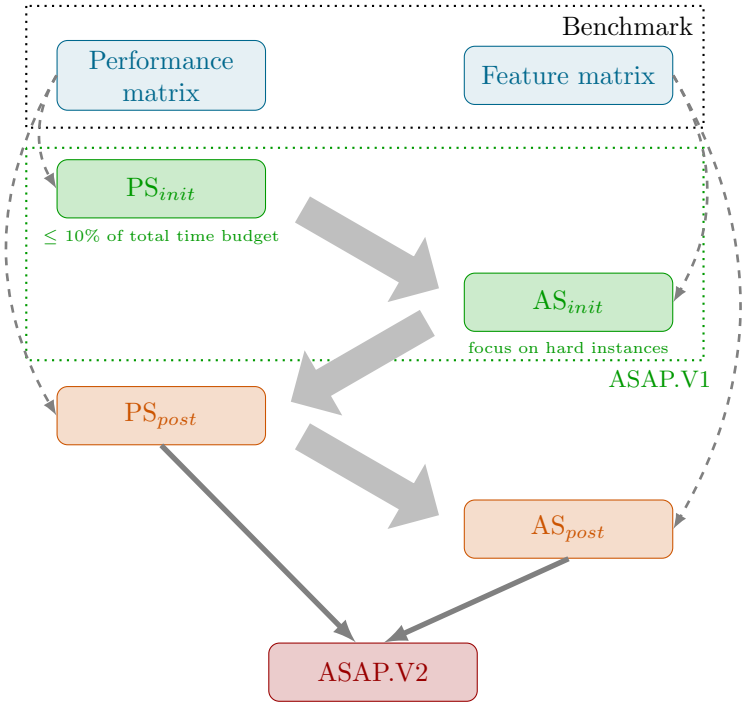

Figure 4.2: ASAP full picture.

- maximizing the proportion of instances solved:

$$\mathcal{F}(\text{PS}_{init}) = p_s(\text{PS}_{init}) = \sum_{\mathbf{x}_i \in \mathcal{I}} n_s(\mathbf{x}_i, \text{PS}_{init})$$

(with p_s as defined in Eq. (3.5)).

The ASAP.V1 pre-scheduler PS$_{init}$ solves the following optimization problem:

$$\text{find } T_{ps} \in \left[0, T_{ps}^{max}\right], (a_1, \ldots, a_K) \in \mathcal{A}^K, \ \max p_s\left((a_k, \frac{T_{ps}}{K})_{k=1}^K\right) \tag{4.2}$$

This mixed optimization problem is tackled in a hierarchical way, determining for each value of T_{ps} the optimal K-tuple of algorithms $(a_1, \ldots, a_K)$.[1] Thanks to both small K values ($K = 3$ in the experiments) and small number of algorithms (≤ 31 in the ASlib datasets, Sec. 3.4), the optimal K-uple is determined by exhaustive search for each T_{ps} value.

The ASAP.V1 pre-scheduler finally relies on the 1-dimensional optimization of the overall computational budget T_{ps} allocated to the presolving schedule. In all generality, the optimization of T_{ps} is a multi-objective optimization problem, *e.g.* balancing the proportion of problem instances solved by PS$_{init}$ and the overall computational budget spent during the pre-scheduler phase. In this regard, PS$_{init}$ should rather be optimized considering the following bi-objective optimization problem:

$$\text{find } T_{ps} \in \left[0, T_{ps}^{max}\right], (a_1, \ldots, a_K) \in \mathcal{A}^K, \ \left\{ \begin{array}{cc} \max & p_s\left((a_k, \frac{T_{ps}}{K})_{k=1}^K\right) \\ \min & T_{ps} \end{array} \right. \tag{4.3}$$

[1] Note that only a finite number of T_{ps} values need to be considered.

Multi-objective optimization commonly proceeds by determining the so-called Pareto front, consisting of non-dominated solutions. In our case, the Pareto front depicts how the proportion of solved instances varies with $T_{ps} \in [0, K \cdot T_{ps}^{max}]$, as illustrated on Fig. 4.3.

In multi-objective decision making [Branke et al., 2004], the choice of a solution on the Pareto front is tackled using post-optimal techniques [Deb, 2003], including: i) compromise programming, where one wants to find the closest point to an ideal target in the objective space, *e.g.* selecting the point closest to O in Fig. 4.3; ii) aggregating the objectives into a single one, *e.g.* using linear combination; or iii) marginal rate of return. The last heuristics consists of identifying the so-called "knees", that is, the solutions where any small improvement along a given criterion is obtained at the expense of a large decrease along another criterion, defining the so-called marginal rate of return. The vanilla marginal rate of return is however sensitive to strong local discontinuities; for instance, it would select point A in Fig. 4.3. Therefore, Deb and Gupta [2011] propose the *bend-angle approach*, a variant taking into account the global shape of the curve, and measuring the marginal rate of improvement w.r.t. the extreme solutions on the Pareto front (*e.g.* selecting point B instead of point A in Fig. 4.3).

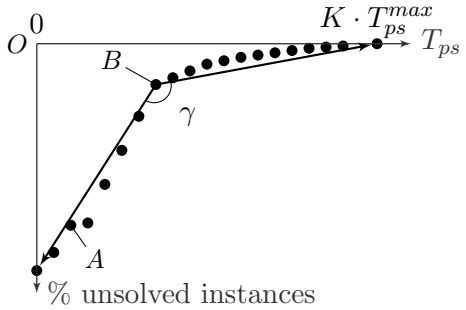

Figure 4.3: Knee detection: bend-angle approach. Among a set of Pareto-optimal solutions, solution A has the best marginal rate of return (local knee detection criteria); solution B, minimizing the bend-angle γ, is a knee as proposed in [Deb and Gupta, 2011].

The bend-angle approach however does not always find a knee: if the proportion of instances solved goes linearly with the runtime, bend-angle γ is constant, yielding an infinite number of possible knees.[2] Furthermore, the bend-angle is affected by objective re-scaling and would thus require to explicit a trade-off parameter to balance among both objectives.

Geometrical knee detection criterion. A new knee detection heuristics is proposed, invariant w.r.t. objective re-scaling and resulting in a finite number of possible knees. The *geometrical criterion* depicted in Fig. 4.4, minimizing the grey-shaded area, is considered in ASAP.

The geometrical criterion has the following properties:

- It is insensitive to objective re-scaling (as any rectangle area would be multiplied by the same factor). Let us consider the normalized curve and notations of Fig. 4.5 in the following.
- Provided that the performance curve is above the diagonal ($\forall t \in [0,1], f(t) \geq t$), knees are detected for $t \in \left[0, \frac{1}{2}\right]$. As a consequence, $T_{ps} \in \left[0, \frac{K}{2} T_{ps}^{max}\right]$.[3] Notice that

[2] Deb and Gupta [2011] suggest a fixed sharpness threshold on the bend-angle. With the notations of Fig. 4.3, γ should be at most $\pi - 1$ rad to be considered as a knee.

[3] If the knee x-coordinate exceeds T_{ps}^{max}, T_{ps} clipped to T_{ps}^{max} (in practice, this happens rarely).

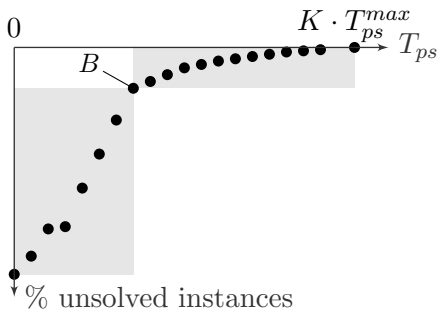

Figure 4.4: Knee detection: geometrical criterion. Solution B, which minimizes the grey-shaded areas is the knee selected in ASAP.

for $K \geq 2$, the entire range $\left[0, T_{ps}^{max}\right]$ is accessible.

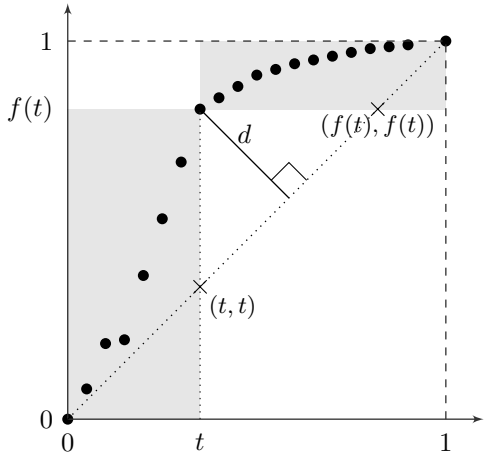

Figure 4.5: Notations for the geometrical criterion and the Normal-Boundary Intersection method. The boundary line (dotted line) joins the two extreme Pareto points, *i.e.* $(0, 0)$ and $(1, 1)$

A related knee detection method is the Normal-Boundary Intersection (NBI) technique [Das, 1999], where a knee is defined as the point of the curve farthest away from the "boundary line", *i.e.* maximizing distance d.[4] Noticing that $d^2 = \frac{1}{2}\left(f(t) - t\right)^2$, the NBI technique selects the knee as the point $(t^*_{NBI}, f(t^*_{NBI}))$ where:

$$t^*_{NBI} = \underset{t}{\operatorname{argmax}} \left(f(t) - t\right)^2 = \underset{t}{\operatorname{argmin}}\, \ell_{NBI}(t) \tag{4.4}$$

with $\ell_{NBI}(t) = -\left(f(t) - t\right)^2$

[4]The NBI technique is also insensitive to objective re-scaling [Das, 1999], enabling a fair comparison with the geometrical criterion on the normalized curve.

Using the notations of Fig. 4.5, the geometrical criterion reads:

$$\begin{aligned} t^*_{gm} &= \operatorname*{argmin}_t \{t \cdot f(t) + (1-t) \cdot (1-f(t))\} \\ &= \operatorname*{argmin}_t \{2t \cdot f(t) - t - f(t) + 1\} \\ &= \operatorname*{argmin}_t \left\{ -(f(t)-t)^2 + t^2 + f(t)^2 - t - f(t) + 1 \right\} \\ &= \operatorname*{argmin}_t \left\{ -(f(t)-t)^2 + \left(t - \frac{1}{2}\right)^2 + \left(f(t) - \frac{1}{2}\right)^2 + \frac{1}{2} \right\} \\ &= \operatorname*{argmin}_t \left\{ \ell_{NBI}(t) + \left\| (t, f(t)) - \left(\frac{1}{2}, \frac{1}{2}\right) \right\|_2^2 + \frac{1}{2} \right\} \end{aligned} \tag{4.5}$$

The geometrical criterion differs from the NBI criterion in that it includes an additional L_2 penalty term, the distance to point $(1/2, 1/2)$ in the normalized objective space. This term aims at trade-offs that do not pass over one objective or the other. Finally, $T_{ps} = t^*_{gm} \cdot K \cdot T^{max}_{ps}$.

Finally, the selected algorithms are ranked by decreasing number of instances solved in the training set.[5] This decision is motivated as: i) the criterion considered so far (number of instances solved) is invariant w.r.t. the ordering of algorithms within PS_{init}; ii) solving the most instances as early as possible is likely to save more time.

Performance analysis

Presolving-schedule duration The performances of PS_{init} with 3 algorithms ($K = 3$) are evaluated using the ASlib cross-validation, as reported in Tab. 4.1. As PS_{init} is blind to the actual time required to solve an instance, only the proportion of instances solved is considered. Notation $S^{\leq T}$ indicates that schedule S is cut at T. PS_{init} is naturally run with budget T_{ps}. As a comparison, the *oracle* selector performances with budget T_{ps} and budget T^{max} are reported.

Second column in Tab. 4.1 illustrates that a small fraction of the total time budget is allocated to PS_{init}. For 7 out of 13 datasets T_{ps} is strictly smaller than 5% of T^{max}. Comparing $\text{PS}^{\leq T_{ps}}_{init}$ and $oracle^{\leq T^{max}}$, more than half of the solvable instances (*i.e.* instances solved by $oracle^{\leq T^{max}}$, rightmost column of Tab. 4.1) are solved by PS_{init}. For the CSP-2010, PROTEUS-2014 and SAT11-HAND datasets, PS_{init} is nearly as efficient as the oracle when budgets are equal (3rd and 4th columns). In contrast, PS_{init} yields much lower performances for, *e.g.* the SAT-12-INDU dataset.

In brief, from the PIAS perspective the selector component budget is only slightly reduced by the addition of a PS, while in all cases its task is at least reduced by half.

Presolving-schedule algorithms The selection of the presolving schedule algorithms is investigated from the algorithm performance correlation perspective. A $|\mathcal{I}_{train}| \times |\mathcal{A}|$ performance matrix is built, recording for each algorithm whether (value 1) or not (value 0) it solved each instance within T_{ps} (*i.e.* , which instances each algorithm would be able

[5] Due to the ICON challenge setting (Sec. 4.3), forbidding a same algorithm to appear twice for any particular problem instance, when the PIAS selects an algorithm in PS_{init}, this algorithm has to be removed from the schedule phase. More specifically, the schedule $((a_1, \tau_1), \ldots, (a_{AS}, \tau_j), \ldots, (a_K, \tau_K), (a_{AS}, \tau_{AS}))$ is turned into $((a_1, \tau_1), \ldots, (a_{j-1}, \tau_{j-1}), \ldots, (a_{j+1}, \tau_{j+1}), \ldots, (a_K, \tau_K), (a_{AS}, \tau_{AS} + \tau j))$. This adjustment is made during the evaluation.

Dataset	T_{ps}/T^{max}	$p_s(\text{PS}_{init}^{\leq T_{ps}})$	$p_s(oracle^{\leq T_{ps}})$	$p_s(oracle^{\leq T^{max}})$
ASP-POTASSCO	5.0%	68.9%	78.4%	93.7%
CSP-2010	0.6%	72.9%	72.9%	87.5%
MAXSAT12-PMS	3.6%	67.8%	75.0%	85.3%
PREMARSHALLING	2.8%	64.7%	73.4%	100.0%
PROTEUS-2014	1.7%	75.2%	79.5%	88.7%
QBF-2011	2.1%	59.6%	64.1%	77.0%
SAT11-HAND	2.7%	41.9%	45.9%	74.0%
SAT11-INDU	5.8%	45.7%	53.7%	84.3%
SAT11-RAND	3.1%	56.8%	63.5%	82.0%
SAT12-ALL	6.2%	50.7%	61.6%	98.8%
SAT12-HAND	6.4%	36.0%	44.1%	70.1%
SAT12-INDU	7.5%	42.6%	57.1%	82.1%
SAT12-RAND	5.0%	58.0%	64.1%	76.4%

Table 4.1: Proportion of the total time budget allocated to PS_{init} (T_{ps}/T^{max}) and proportion of instances solved by $\text{PS}_{init}^{\leq T_{ps}}$, $oracle^{\leq T_{ps}}$ and $oracle^{\leq T^{max}}$. "PREMARSHALLING" stands for "PREMARSHALLING-ASTAR-2013"

to solve if it were incorporated in PS_{init}). A correlation matrix is then derived. PS_{init} must select $K = 3$ algorithms that are complementary, *i.e.* with low pairwise correlation.

The correlation matrices for the SAT11-HAND and SAT12-INDU datasets are reported in Fig. 4.6, with the 3 algorithms selected in PS_{init} indicated with "**". For simplicity, only the first fold of the ASlib cross-validation is considered. These two datasets are chosen for the reason that they widely differ in Tab. 4.1: PS_{init} performs nearly as well as the oracle on SAT11-HAND (41.9% vs. 45.9%) while it is largely dominated on SAT12-INDU (42.6% vs. 57.1%).

On SAT11-HAND, the performances of all algorithms are largely correlated, except for two of them (`sattine+` and `sattime` algorithms). As expected, the pre-scheduler has selected 3 algorithms that have relatively low correlations: resp. 0.57, 0.28 and 0.31 for pairs (`Sol`, `clasp2`), (`Sol`, `sattime`) and (`clasp2`, `sattime`). Note that all algorithms solve comparable numbers of instances: 69.9 on average (±9.9). With resp. 89, 85, 73 instances solved, `sattime`, `Sol` and `clasp2` are resp. the 1st-, 3rd- and 5th-best performing algorithms. These algorithms have different behavior; as expected, PS_{init} involves very different algorithms to maximize its coverage.

SAT12-INDU presents a different picture. 8 algorithms are un-correlated with all others and among themselves (with the exception of the (`sattime`, `sattimep`) pair). A more in-depth investigation reveals that they are uncorrelated, as they solve substantially fewer instances (less than 62, while the least performing among the 24 other algorithms solves 203 instances and 18 algorithms solve more than 350 instances). Despite this lack of correlation, these 8 algorithms thus are poor choices for PS_{init}. Eventually, PS_{init} includes the 2nd-, 7th- and 9th-best performing algorithms.

As a partial conclusion, using the correlation performances to compute the best combination of algorithms in PS_{init} is hardly sound, as illustrated of the SAT12-INDU dataset. Therefore, using a dedicated solver (as is done in, *e.g.* [Hoos et al., 2015]) appears to be a better and more robust way to build a presolving schedule from a large algorithm portfolio.

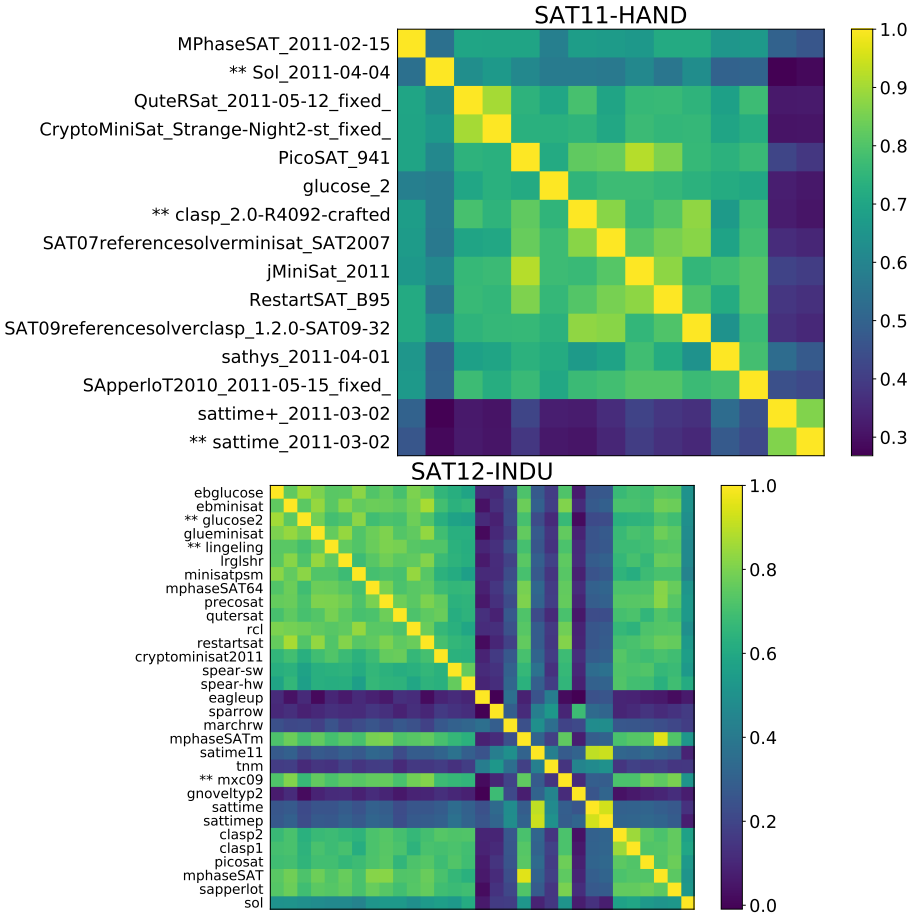

Figure 4.6: Algorithm performance correlation on the SAT11-HAND (top) and SAT12-INDU (bottom) datasets. The 3 algorithms that constitute the presolving schedule are indicated with double stars.

Discussion and limitations

A main way of preventing PS_{init} from overfitting is to enforce an equal time-sharing mechanism; the price to pay is to reduce the performances. The main other restrictive design choices are discussed thereafter.

The choice of T_{ps} is very sensitive to T_{ps}^{max}. The optimum of the geometrical criterion (Fig. 4.4) is greatly affected if T_{ps}^{max} is fixed to, *e.g.* 20% of T^{max} instead of 10%. On the other hand, setting T_{ps}^{max}as a fraction of the dataset timeout is desirable as "quick" or "affordable" are relative to the context – noting that the dataset timeout in ASlib varies from 600 to 5000s, Sec. 3.4. Finally, T_{ps} would also be affected by re-scaling the runtime, *e.g.* using the logarithm of the runtime as done by Leyton-Brown et al. [2002] when learning the performance models.

The ordering of algorithms relies on an informal argument. Other approaches, *e.g.* `aspeed` [Hoos et al., 2015] formally optimize the algorithm ordering.

The equally shared runtimes setting is sometimes used as a baseline by sequential schedule approaches (referred to as *uniform share allocator* in [Gagliolo and Schmidhuber, 2006] or *split schedule* in [O'Mahony et al., 2008]), and it is known to be suboptimal. ASAP.V2 (Sec. 4.5) will reconsider and improve the time-sharing issue.

4.2.2 ASAP.V1 algorithm selector

As mentioned in Sec. 4.1.1, the AS relies on performance models, regressors learned from the training problem instances and mapping the instance description space $\mathcal{X}$ onto the algorithm performance space. The performance predicted is the penalized runtime ($PR10$, see Sec. 3.1.3). Standard performance models include k-nearest neighbor approaches [Nikolić et al., 2013], ridge-regression [Xu et al., 2008], support vector regression [Gebser et al., 2011], random forest models (included, *e.g.* in `LLAMA` [Kotthoff, 2013]). After preliminary experiments the choice was made to use two learning algorithms: random forests and k-nearest neighbors, both well-suited to non-linear modeling. Two alternatives of ASAP.V1 are built – respectively named `ASAP_RF.V1` and `ASAP_kNN.V1` – illustrating that the ASAP.V1 approach can accommodate any PIAS.

For each ML algorithm, one hyper-parameter was tuned to each setting,[6] based on a few preliminary experiments: 35 trees are used for the random forest algorithm and the number of neighbors is set to 3 for the k-nearest neighbor approach. In the latter case, the predicted value associated with problem instance $\mathbf{x}$ is set to the weighted sum of the performance of its nearest neighbors, weighted by their relative distance to $\mathbf{x}$ in feature space:

$$\widehat{\mathcal{G}}(\mathbf{x}, a) = \frac{\sum_i ||\mathbf{x} - \mathbf{x}_i|| \mathcal{G}(\mathbf{x}_i, a)}{\sum_i ||\mathbf{x} - \mathbf{x}_i||} \tag{4.6}$$

where $\mathbf{x}_i$ ranges over the 3 nearest neighbors of $\mathbf{x}$. Features are centered (zero mean, unit variance) before selecting the neighbors.

4.2.3 Feature set improvement

As explained in Chap. 3, feature pre-processing is crucial for PIAS. While the ASlib data provide a rich set of manually designed features – expected to well capture the optimization instances specifics – improvements are sought along two directions: the addition of new features and feature selection.

[6] All other hyper-parameters are set to their default value, using the Python `scikit-learn` library [Pedregosa et al., 2011].

Missing data features

A major difficulty arises from the missing data in the representation of problem instances. Typically, feature values are missing for some groups of features, for quite a few problem instances, due to various causes provided as the computational status of each group of features (Sec. 3.4.1). The best practice, after the missing data community, is to use all available information and input the missing data rather than discarding examples [Schafer and Graham, 2002]. Accordingly, missing values are handled by:

- replacing the missing value by the average over the known values of this feature;
- adding to the set of descriptive features 7 additional Boolean features per group of initial features, indicating whether the feature group values are available or the reason why they are missing otherwise (using the feature computational status as provided in ASlib, Sec. 3.4).

Presolving schedule features

Taking inspiration from dynamic features (Sec. 3.2.2), Boolean features are introduced to indicate whether each algorithm in the presolving schedule succeeded in solving the instance in the allocated time. The intuition is that the presolving schedule algorithms not only help solving the instance but also inform about the difficulty of solving the instance if they fail. For new instances, the inclusion of such information is made possible, because features are not needed before the end of the presolving schedule is reached.

Another option considered in the literature (*e.g.* in `SATzilla`) consists in discarding all "easy" instances (*i.e.* instances that the presolvers solved) during the training of performance models. This choice ensures that performance models are most relevant to "hard" instances. We did not adopt this approach, however, due to the limited size of datasets, as further reduction of their size would increase the risk of overfitting.

Feature selection

The increase in the overall number of features is handled by an embedded feature selection mechanism, removing all features with negligible importance as determined by a random forest. Specifically, a 10-trees random forest regression model is separately trained to predict the algorithms $PR10$. The performance prediction for each training sample is obtained by aggregating all trees. The individual tree prediction is computed following a path from the root down to a leaf; each non-leaf node denotes a test along a specific feature and the prediction follows the path depending on the outcome of the test for the instance at hand. Depending on the path, a number of features thus contribute to the final performance prediction. An approximate feature importance criterion is based on the fraction of training sample predictions each feature contributes to; features below a manually specified threshold are then discarded.

Fig. 4.7 reports the effect of the selection **on a k-NN based algorithm selector component alone** (*i.e.* with no pre-scheduler) with $k = 3$; threshold values equal or greater than 10^{-1} are omitted since they tend to discard all features in some cases. A first observation is that the optimal threshold varies across datasets, and thus should be adapted per dataset. This is consistent with the recent attempts toward hyperparameter automatic configuration when building algorithm portfolios [Hoos et al., 2014; Lindauer et al., 2015]. The average optimal values retained across all datasets are displayed in Tab. 4.2. The results suggest that the improvements that can be achieved with feature selection

alone are minor and not significant (after a Wilcoxon signed-rank test for each pair of systems: p-values are all above 0.1 – except for the $5 \cdot 10^{-2}$ threshold with significantly worse overall performance). Given the very low sensitivity of the threshold parameter in the range $[10^{-5}, 10^{-2}]$, the selection threshold was fixed to 10^{-5}; this mostly excludes only quasi-constant features. We note that this threshold value is optimal for the random forest selector.

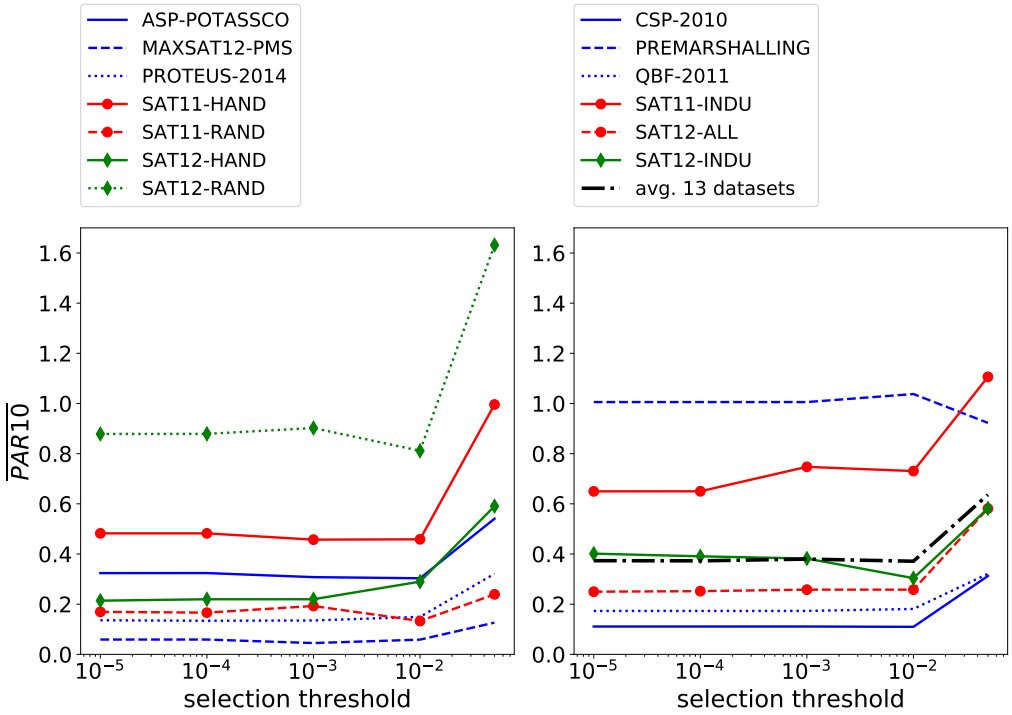

Figure 4.7: Per-dataset and average normalized PAR10 (defined in Sec. 3.1.4; the lower, the better) on the ASlib cross-validation with respect to the importance threshold (in fraction of the number of training samples, log-scale).

Threshold	0	10^{-5}	10^{-4}	10^{-3}	10^{-2}	$5 \cdot 10^{-2}$
$\overline{PAR10}$ (3-NN)	0.372	0.373	0.373	0.380	0.371	0.636

Table 4.2: Normalized PAR10 (averaged across all datasets) for different feature selection thresholds. Differences are not significant (according to the Wilcoxon signed-rank test), except for threshold $5 \cdot 10^{-2}$ which is significantly worse than all other thresholds (p-value is 10^{-14}).

As previously stated, the design of the descriptive features is a cornerstone of AS. In particular, it is often the case that the most informative features in order to predict $\mathcal{G}(\mathbf{x}, a)$ are computationally expensive themselves. For this reason, the sophisticated `SATzilla` algorithm involves prediction of the computational cost required to for feature computation, and only launch these computations if the prediction is below a given a threshold [Xu et al., 2012b].

Experiments on feature preprocessing

The previously mentioned feature preprocessing mechanisms are individually assessed, switching them on and off to investigate their impact on the performance. Naturally the presolving schedule features can only be assessed within a pre-scheduler scheme; all results (Fig. 4.8) are thus obtained with a $K = 3$ pre-scheduler for a fair comparison. The 3-NN selector is used to enable the comparison with the feature selection threshold sensitivity analysis (Tab. 4.2).

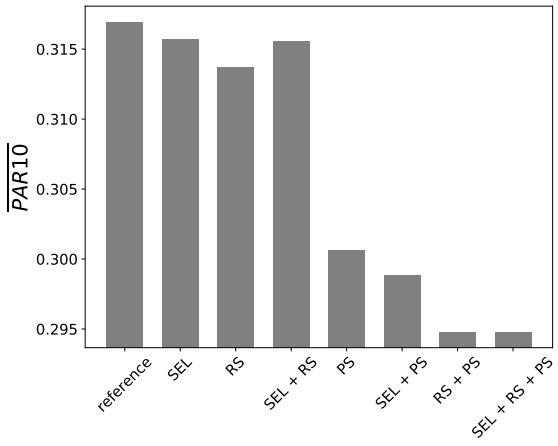

Figure 4.8: Comparison of different feature set improvements in ASAP.V1 (with a 3-step presolving schedule). SEL: feature selection activated with a 10^{-5} threshold. RS: feature run statuses are encoded as additional Boolean features. PS: the 3 presolving schedule features are added. By definition of the normalized measure, the SB strategy scores 1 and the oracle scores 0.

The inclusion of feature run status (RS) and the feature selection (SEL) have a moderate impact on ASAP.V1 performances: their effect is comparable with a careful tuning of the selection threshold in terms of intensity (of the order of 10^{-3}; note that the results presented in Fig. 4.8 are far better than those presented in Tab. 4.2 because of the use of a 3-step presolving schedule). However, a large benefit comes from including the presolving schedule features (PS) in the instance description. The significance of the improvement is assessed using a Wilcoxon signed-rank test between the matched sets of all datasets and all splits (130 measures) of each pair of options; p-values are reported in Tab. 4.3. The remainder of the experiments was performed with the most promising combination "SEL+RS+PS" including all preprocessing mechanisms.

	SEL	RS	SEL + RS	PS	SEL + PS	RS + PS	SEL + RS + PS
reference	$1.0 \cdot 10^{0}$	$2.9 \cdot 10^{-1}$	$6.7 \cdot 10^{-1}$	$4.0 \cdot 10^{-2}$	$3.9 \cdot 10^{-2}$	$\mathbf{7.6 \cdot 10^{-3}}$	$\mathbf{7.9 \cdot 10^{-3}}$
SEL	-	$3.4 \cdot 10^{-1}$	$6.1 \cdot 10^{-1}$	$5.8 \cdot 10^{-2}$	$5.7 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
RS	-	-	$3.6 \cdot 10^{-1}$	$7.7 \cdot 10^{-2}$	$7.5 \cdot 10^{-2}$	$2.7 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$
SEL + RS	-	-	-	$6.0 \cdot 10^{-2}$	$6.0 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$
PS	-	-	-	-	$3.1 \cdot 10^{-1}$	$5.7 \cdot 10^{-2}$	$2.3 \cdot 10^{-1}$
SEL + PS	-	-	-	-	-	$1.8 \cdot 10^{-1}$	$5.0 \cdot 10^{-1}$
RS + PS	-	-	-	-	-	-	$1.4 \cdot 10^{-1}$

Table 4.3: Improvement significance of the feature improvements setups: p-value of the Wilcoxon signed-rank test (values below the significance threshold 0.01 are in bold).

4.2.4 Sensitivity to pre-scheduler hyperparameters

As already said, the pre-scheduler is controlled by two main hyperparameters:

- K: the number of algorithms in the presolving schedule. When $|\mathcal{A}| \leq K$, all algorithms but one are actually considered in the pre-scheduler while the algorithm chosen by the AS component intervenes last.
- T_{ps}^{max}: the maximal budget available to the pre-scheduler. This choice affects how T_{ps} will be selected.

The global effect of K on ASAP.V1 performances is investigated through a sensitivity analysis. The results are graphically depicted in Fig. 4.9 on a per-dataset basis (top) and on average (bottom). Note that the exhaustive search becomes prohibitively expensive for $K \geq 6$,[7] although some datasets (*e.g.* SAT11-HAND and PROTEUS-2014 datasets) would probably benefit from higher K values. Another goal of this experiment is to make the link between feature selection without any pre-scheduler (Tab. 4.2) and the effect of feature pre-processing (Fig. 4.8), which requires a pre-scheduler.

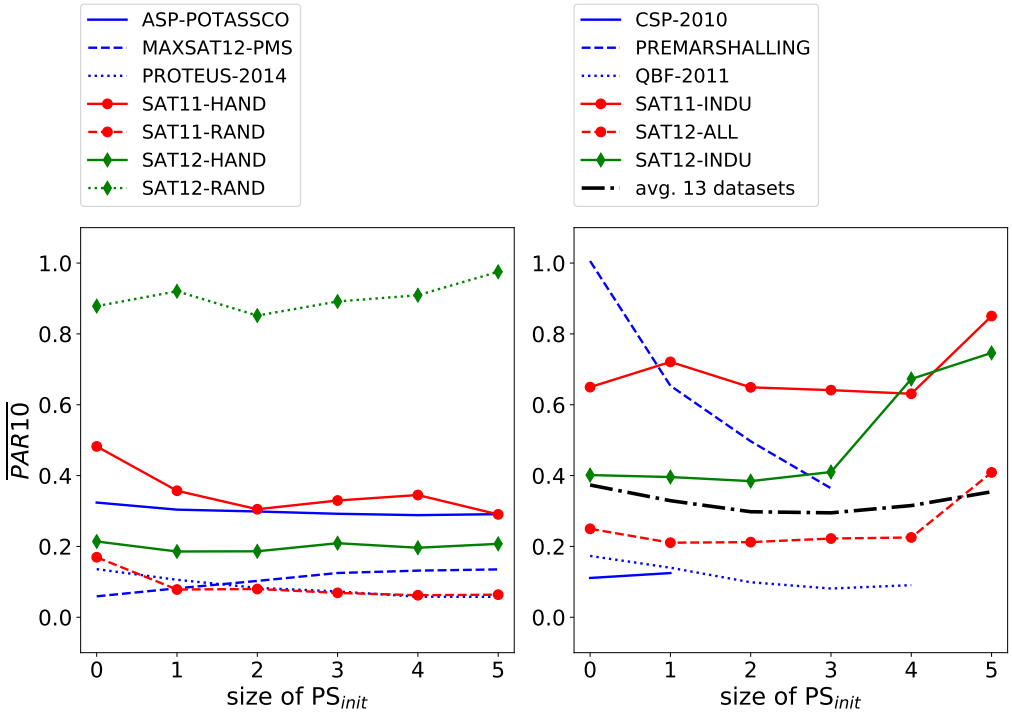

Figure 4.9: Per-dataset and average effect of the presolving schedule size K on `ASAP_kNN.V1` performances (normalized PAR10 measure). Size 0 means that no pre-scheduler is used (ASAP.V1 is thus a pure selector). Since the CSP-2010, PREMARSHALLING-ASTAR-2013 and QBF datasets have respectively 2, 4 and 5 algorithms, the PS$_{init}$ size is upper-bounded by respectively 1, 3 and 4 for those datasets. Not all datasets benefit for the use of a pre-scheduler.

The main conclusion is that the pre-scheduler has a tremendous effect on some datasets. In particular on the PREMARSHALLING-ASTAR-2013 dataset, the 3-NN selector component fails to improve on the SB algorithm (score 1 for a 0-length presolving schedule). A

[7] For $K = 4$, PS$_{init}$ optimization takes up to 3h with a naive implementation; for $K = 5$, it takes 24h for the largest datasets.

3-step presolving schedule, however, does halve the gap with the oracle selector. Note that in this particular case all 4 algorithms are used to solve some instances (3 in the presolving schedule when the 4th is selected; otherwise when one of the 3 "usual" presolving schedule algorithms is selected, the presolving schedule has only 2 steps and runs for a total duration of 2/3rd of T_{ps}). However, increasing the presolving schedule size deteriorates the result for the MAXSAT12-PMS and CSP-2010 datasets. For both, the pure selection approach already shos excellent performance. Therefore, the additional robustness of using a pre-scheduler is not worth the extra cost.[8]

Fig. 4.9 hides that the optimization in T_{ps} is done *after* the choice of K, meaning that T_{ps} is hardly comparable for different values of K. Another point to keep in mind is that when the presolving schedule size increases, the time shares allocated to each algorithm tend to decrease, which is why increasing the size of the pre-scheduler does not necessarily lead to global performance improvement. In the average perspective over all datasets (Fig. 4.9, bottom), a good trade-off is obtained for $K = 3$. However, this average is heavily affected by the huge benefit of the pre-scheduler on the PREMARSHALLING-ASTAR-2013 dataset. In conclusion, this parameter should be tuned in a dataset-dependent manner.

4.3 Evaluation framework: The ICON Challenge on Algorithm Selection

This section details the ICON Challenge on Algorithm Selection, used to experimentally validate the ASAP systems (Secs. 4.4, 4.6).

4.3.1 Specific rules of the challenge

The ICON Challenge on Algorithm Selection was carried on between February and July 2015 to evaluate AS systems in a fair, comprehensive and reproducible manner. Submissions were required to be free for academic use and were run by the organizers for the final evaluation. The codes of all submitted systems and the results were made publicly available after the end of the challenge.[9]

Task of the challenge

Each submitted system provides a dataset-dependent, per-instance schedule of algorithms to be trained and tested on the ASlib datasets.

This sequential algorithm portfolio can optionally be preceded with a per-dataset presolver (single algorithm running on all instances during a given runtime before the per-instance schedule runs).

Each system can also, per-dataset, specify the groups of features to be used in order to save the time needed to compute features. As some features are computed in several steps, some precedence constraints have to be respected.[10]

[8] A careful comparison of the full system with 0- and 1- step presolving schedule reveals that for MAXSAT12-PMS and CSP-2010 datasets, roughly the same number of instances are solved by both options.

[9] Accessible at: `http://challenge.icon-fet.eu/challengeas`

[10] Note that the feature computation order is fixed in the ASlib format.

Challenge evaluation setup

Each submitted system is assessed on the 13 ASlib V.1.0.1 datasets [Bischl et al., 2016] (Sec. 3.4) with respect to three classical performance measures in combinatorial optimization challenges:

1. p_s: proportion of problem instances solved;
2. r_t^+: average extra runtime compared with the oracle solver (a.k.a. virtual best solver). Given a schedule S it is computed as:

$$r_t^+(S) = \sum_{\mathbf{x} \in \mathcal{I}} r_t(\mathbf{x}, S) - r_t(\mathbf{x}, oracle)$$

3. $PAR10$: average Penalized Average Runtime-10 (PAR10) which is the cumulative runtime needed to solve all problem instances (using 10 times the dataset timeout for unsolved instances).

As all datasets were available to the community from the start, the evaluation was based on hidden splits between training and test set. Table 4.4 summarizes the differences between the ASlib provided cross-validation splits and the one used for the challenge evaluation. The main difference regards the size of the splits: training sets are larger in ASlib, resulting in a reduced risk of overfitting, but the test sets are more heterogeneous. For a fair comparison, ASAP systems are tuned using the ASlib cross-validation and evaluated with the challenge splits.

Dataset	**Train set size (avg.)**		**Test set size (avg.)**	
	ASlib	**challenge**	**ASlib**	**challenge**
ASP-POTASSCO	1164	813	129	480
CSP-2010	1821	1275	202	748
MAXSAT12-PMS	788	552	87	323
PREMARSHALLING-ASTAR-2013	474	334	52	192
PROTEUS-2014	3618	2535	402	1485
QBF-2011	1231	865	136	502
SAT11-HAND	266	190	29	105
SAT11-INDU	270	189	30	110
SAT11-RAND	540	376	60	223
SAT12-ALL	1452	1021	161	592
SAT12-HAND	690	482	76	284
SAT12-INDU	1050	734	116	432
SAT12-RAND	1225	853	136	508

Table 4.4: ASlib and ICON challenge splits details. While ASlib considers a 90/10 split ratio between training and test set, the ratio is roughly 1.7 in the challenge setting.

Two baselines are considered: the oracle, selecting the best algorithm for each problem instance; and the single best (SB) algorithm, with best average performances over all problem instances.[11] Note that the feature computation cost is not included in the

[11] After the challenge setting, the SB is determined considering all instances in the dataset – including training and test instances. Due to the limited size of some datasets, the SB based on the training set only might differ from the overall SB.

baselines performances. The baselines are used to normalize every system performance over all datasets, associating performance 0 to the oracle (respectively performance 1 to the SB algorithm). The normalized performance (Sec. 3.1.4) is recalled here:

$$\overline{\mathcal{F}}(S) = \frac{\mathcal{F}(S) - \mathcal{F}(oracle)}{\mathcal{F}(SB) - \mathcal{F}(oracle)} \tag{4.7}$$

with $\mathcal{F}$ from in $\left\{p_s, r_t^+, PAR10\right\}$.[12] This normalization facilitates the aggregation of the system results over all datasets. A value larger than 1 means the submitted system did not improve on the SB algorithm; negative values are impossible by construction. All reasonable systems are thus expected to have performance between 0 and 1.

The per-dataset, per-split score of each system is obtained by simulating the solving schedule S on all instances in the split test set. Provided an instance is solved with times $t_1, \ldots, t_k$ by respectively algorithms $a_1, \ldots, a_k$, its simulated runtime is computed as:

1. Simulate presolver a_{pre} of S during presolving time τ_{pre}. If $t_{a_{pre}} \leq \tau_{pre}$, the runtime is t_{pre} and the instance is marked as solved.

2. Simulate the computation of groups of features required by S. This computation is completed in t_{feat}. If the feature computation solved the instance, the runtime is $\tau_{pre} + t_{feat}$ and the instance is marked as solved.

3. Simulate the run of each step of the schedule $(a_{S(1)}, \tau_{S(1)}), \ldots (a_{S(k)}, \tau_{S(k)})$ provided by S.[13] If the instance is solved by step k^*, the runtime is $\tau_{pre} + t_{feat} + \tau_{S(1)} + \ldots + \tau_{S(k^*-1)} + t_{S(k^*)}$ and the instance is marked as solved. If the end of the schedule or the dataset timeout T^{max} is reached, the runtime is set to T^{max} and the instance is mark as unsolved.

Note that this evaluation is equivalent to considering a pseudo-schedule of the form (presolver, feature computation steps, schedule S) where the steps of features computation do not have any time limit; in practice such time limits exist and are already included in the feature values and costs provided in ASlib.

A total of 130 ballots – one for each (dataset, split) pair – are obtained and averaged to obtain 3 separate rankings (one per performance measure). The official final ranking is computed as the average of the three measures.

4.3.2 Competitors

The ICON challenge was taken on by 4 teams for a total of 8 submitted systems. ASAP.V1[14] entered the competition with two variants: `ASAP_kNN` and `ASAP_RF`, respectively using a k-Nearest Neighbor (k-NN) and a Random Forest (RF) algorithm at the core of the selector component. The 6 other competing systems were: `autofolio`, `flexfolio`, `SUNNY`, `SUNNY-presolv`, `zilla`, `zillafolio`.

As said, each system had the option to use a presolver and to choose (on a per-dataset basis) not to compute a subset of the features. These options were inspired by `SATzilla` (Sec. 3.3.1), as its ability to use presolvers and per-instance feature computation undoubtedly contributed to its success in numerous competitions. For simplicity, ASAP.V1 did not implement such dataset-dependent feature computation; the use of PS_{init} as presolver was also excluded, as only a $K = 1$ presolving schedule was allowed.

[12] Note that $\overline{\mathcal{F}}$ is summarized as a scalar value, although better performances means higher p_s but lower r_t^+ and $PAR10$.

[13] Note that all k algorithms are not necessarily run and that a given algorithm is not allowed to be run in two different steps.

[14] ASAP.V2 (Sec. 4.5) was designed after the end of the competition.

4.4 ASAP.V1 experimental validation in the ICON challenge

ASAP.V1 was assessed during the ICON Challenge on Algorithm Selection[15] after the challenge setting described in Sec. 4.3. As already mentioned, two versions of ASAP.V1 were competing, using the same global architecture but with different selector components:

- `ASAP_kNN.V1`: selector based on performance models learned using a distance-weighted k-NN approach (Eq. 4.6) and $k = 3$;
- `ASAP_RF.V1`: selector based on performance models learned using a random forest approach with 35 trees (all other parameters set to their default values in `scikit-learn` [Pedregosa et al., 2011] framework).

The `ASAP_RF.V1` and `ASAP_kNN.V1` versions ranked 4th and 5th, respectively, on a total of 8 participants. The official ranking is based on the mean average across all datasets, all splits and 3 performance measures, with `zilla` as the winner. As noted by the organizers [Kotthoff, 2015], retaining the median average instead of the mean would make `ASAP_RF.V1` to beat `zilla`. `ASAP_RF.V1` was **awarded an honourable mention**.

4.4.1 Comparative results

Table 4.5 reports the results of all submitted systems on all datasets. The general trend is that `zilla` systems dominate all other algorithm selector approaches on the SAT datasets, as expected from their experience in the SAT domain, since they have consistently dominated the SAT contests in the last decade. On most of non-SAT problems however, `zilla` systems are dominated by `ASAP_RF.V1`.

On the negative side, the ASAP approaches perform slightly worse than the single best on some datasets though they never rank last. The rescaled performances of `ASAP_RF.V1` is compared to `zilla` and `autofolio` (Fig. 4.10, on the left). On the positive side, `ASAP_RF.V1` delivers balanced performances, significantly lower than for `zilla` and `autofolio` on the SAT problems, but significantly higher on the other datasets; it defines an overall robust portfolio approach.

4.4.2 Improvement perspectives on the ASAP.V1 selector component

The ICON challenge results [Kotthoff, 2015] suggest that ASAP.V1 requires less computation that most other competitors; note, however, that the overall computational budget was limited to 12 CPU hours on every dataset. Further experiments explored how ASAP.V1 with random forest can be straightforwardly improved with the use of a larger random forest, as reported in Fig. 4.11 for the ASlib evaluation setting.

The average gain of using 200 trees (instead of 35 in `ASAP_RF.V1`) is 3.7%, though the gain widely varies from one dataset to another. In short, a larger random forest is beneficial on most datasets, while it significantly reduces the performances for some datasets (up to 45% loss for SAT11-HAND). This suggests that a per-dataset approach (*e.g.* through automatic configuration) is needed to achieve the best selector performance.

Possible directions of improvement for the selector include the use of a pairwise regression approach to replace the current performance prediction based AS. A similar change has been introduced in the 2012 version of `SATzilla` [Xu et al., 2012b]. The extended evaluation on the ICON challenge setup [Kotthoff, 2015] (including additional systems)

[15] All data, code and results are available at: `http://challenge.icon-fet.eu/challengeas`.

	ASAP_RF.V1	ASAP_kNN.V1	autofolio	flexfolio	sunny	sunny-presolv	zilla	zillafolio
ASP-POTASSCO	0.294 (2)	0.359	0.299	0.314	0.37	0.336	0.319 (5)	**0.283**
CSP-2010	**0.146** (1)	0.247	0.288	0.223	0.263	0.406	0.2 (3)	0.157
MAXSAT12-PMS	0.168 (4)	0.159	0.45	**0.149**	0.166	0.224	0.201 (5)	0.233
PREMARSHALLING	0.349 (4)	0.369	0.359	0.307	0.325	**0.296**	0.374 (7)	0.385
PROTEUS-2014	0.16 (4)	0.177	0.222	**0.056**	0.134	0.103	0.245 (8)	0.223
QBF-2011	0.097 (2)	**0.091**	0.169	0.096	0.142	0.162	0.191 (7)	0.194
SAT11-HAND	0.341 (4)	0.318	0.342	0.342	0.466	0.464	0.328 (3)	**0.302**
SAT11-INDU	1.036 (5)	0.957	**0.875**	1.144	1.13	1.236	0.905 (2)	0.966
SAT11-RAND	0.104 (6)	0.09	**0.046**	0.226	0.116	0.088	0.053 (2)	0.067
SAT12-ALL	0.392 (5)	0.383	0.306	0.502	0.509	0.532	**0.273** (1)	0.322
SAT12-HAND	0.334 (5)	0.31	**0.256**	0.434	0.45	0.467	0.272 (2)	0.296
SAT12-INDU	0.955 (6)	0.919	0.604	0.884	1.074	1.018	0.618 (3)	**0.594**
SAT12-RAND	1.032 (5)	1.122	0.862	1.073	1.126	0.97	**0.779** (1)	0.79

Table 4.5: ICON challenge results: normalized performances of submitted systems, aggregated across all splits and all measures (the lower, the better). Ranks of `zilla` (challenge winner) and `ASAP_RF.V1` (honourable mention) are given in parenthesis. Numbers are computed from the challenge outputs. Note: "PREMARSHALLING" stands for "PREMARSHALLING-ASTAR-2013".

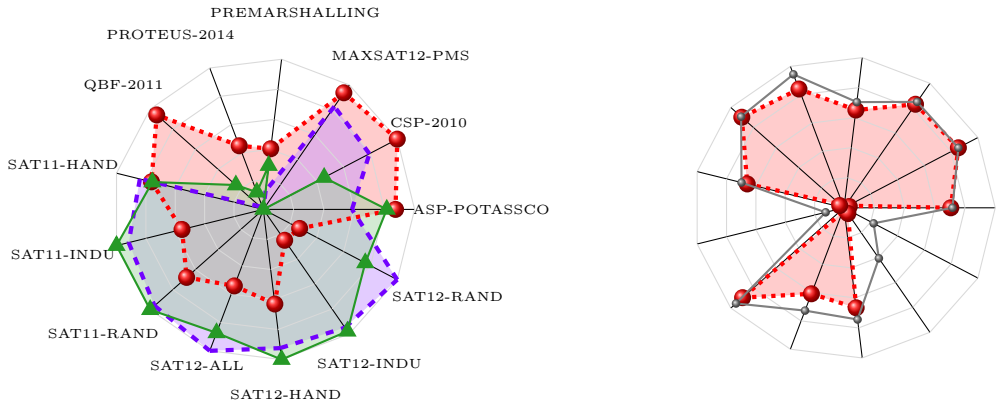

Figure 4.10: Comparative performances. Left: per-dataset performances of `ASAP_RF.V1` (balls, dotted line), `zilla` (no marker, dashed line) and `autofolio` (triangles, solid line); the scale is such that 0 corresponds to the worst submission and 1 to the best one. Right: comparison of `ASAP_RF.V1` and the best submitted AS per dataset in normalized performance (small balls, solid line); the scale is such that normalized performance values 1 (SB) and 0 (oracle) are represented as the center and at the outer end of a radius, respectively. On all datasets except 3, `ASAP_RF.V1` reaches similar performance to the best submitted AS on this dataset.

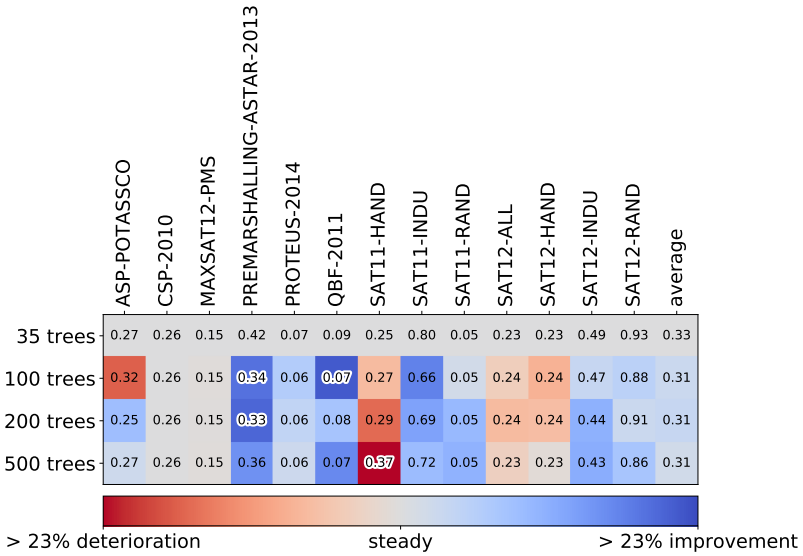

Figure 4.11: Effect of the number of trees on `ASAP_RF.V1` selector (normalized PAR10 score). Color depicts the relative improvement w.r.t. the initial 35 trees setting (blue denotes improvement, red denotes deterioration).

shows that the vanilla pairwise regression based-AS `llama-regrPairs` by the challenge organizers would have beaten both versions of ASAP.V1. In contrast, the vanilla regression based-AS `llama-regr` is behind, stressing again that ASAP architecture – using a pre-scheduler in addition to a standard regression based AS – is an efficient solution.

Another direction of improvement is per-dataset autoconfiguration as done in `autofolio` [Lindauer et al., 2015]. A pairwise regression based approach would most likely benefit of such overhead, as it comprises a wider range of hyperparameters (controlling the regression and the final aggregation through, *e.g.* vote).

4.4.3 Summary and Discussion

ASAP.V1 combines a algorithm schedule with a performance model based algorithm selector. This approach allows ASAP.V1 to achieve peak performances on a per-instance basis thanks to the AS component while the pre-scheduler helps keeping the effect of a critical AS failure low. This division of task is addressed as a bi-objective problem (peak performances *vs.* robustness), with a single parameter controlling the trade-off between the budget shares allocated to each objective.

The merits of this dual architecture were demonstrated during and after the ICON challenge on AS, as ASAP.V1 outperforms both a pure single performance model based AS (`llama-regr` of the organizers, with a similar random forest algorithm as a predictor) and a pure scheduler (`flexfolio`), and to a larger extent the SB algorithm. These conclusions are in line with a number of comparable experimental validations in similar settings, *e.g.* the successes of `SATzilla` and `3S` in different competitions.

The analysis of the sensitivity to the main hyperparameters suggests that the algorithm portfolio itself should be configured on a per-dataset basis. Such observation is at the root of the `autofolio` design, where the prominent algorithm configurator `SMAC` [Hutter et al., 2011] is used to select and parameterize the portfolio approach. However, specific mechanisms need to be implemented to be able to deal with this extended parameter space and preserve the generalization property. The most natural option is to consider

more data. However, data campaigns are expensive; moreover, they would require an instance generator. Data augmentation could also be used; however, the perturbation of known instances also requires some specific knowledge about the underlying instance distribution. A second option is to use a cross-validation scheme to learn the AS and the hyperparameters using separate sets of data to prevent overfitting; this solution is used in `autofolio`. A last option is based on regularization, a widely used technique in machine learning: the complexity of the system is penalized through an additional term in the training loss function.

Another most promising option is to revisit the equal time-sharing scheme of the pre-scheduler. Although a pre-scheduler mainly aims to improve the PIAS robustness, it is a sequential algorithm portfolio of itself. Previous work [Gagliolo and Schmidhuber, 2006] has demonstrated that the equal time sharing (called *uniform time allocator*) option is suboptimal.

4.5 ASAP.V2

The lessons learned from the ICON challenge inspired several extensions of ASAP.V1, aimed at exploring a richer pre-scheduler-AS search space while preventing the risk of overfitting induced by this larger search space. Two directions of improvement along this line are identified. Firstly, the presolving schedule is viewed as a portfolio of its own, and it makes sense to optimize it as well, particularly so regarding the equal time-sharing scheme. The second improvement regards the division of labor between the pre-scheduler and the AS components. Noting that AS_{init} was trained to handle the instances that were not solved by the PS_{init}, it appears relevant to optimize PS_{init} so that it may compensate for the mistakes of AS_{init}, and to eventually jointly optimize both pre-scheduler and AS components. In ASAP.V2, the joint optimization is tackled using an alternating optimization scheme.

With same notations as in Sec. 4.1.1, ASAP.V2 outputs a schedule $S_{ASAP.V2}$ composed of PS_{post} and AS_{post}:

$$S_{ASAP.V2} = \big((a_1, \tau_1), \ldots, (a_K, \tau_K), (a_{AS_{post}}, \tau_{AS_{post}})\big) \tag{4.8}$$

Fig. 4.12 depicts the proposed alternating optimization procedure.

4.5.1 ASAP.V2 pre-scheduler

PS_{post} differs from PS_{init} as it considers the use of different timeouts τ_k for each algorithm in the presolving schedule; the set of algorithms $(a_1, \ldots, a_K)$ and the overall computational budget T_{ps} are as in PS_{init}. The hierarchical optimization strategy (Sec. 4.2.1), is thus extended along two directions:

- timeouts $(\tau_1, \ldots \tau_{K-1})$ are optimized conditionally to $\sum_{k=1}^{K-1} \tau_i \leq T_{ps}$ (the remaining budget $T_{ps} - \sum_{k=1}^{K-1} \tau_k$ devoted to the pre-scheduler is assigned to τ_K). Whereas PS_{init} maximizes the proportion of training instances solved, PS_{post} directly optimizes the $PAR10$ score.

- optimization is done conditionally on AS_{init}, *i.e.* the performance of the full system (PS_{post}, AS_{init}) is considered as the optimization objective. This encourages PS_{post} to focus on solving instances badly handled by AS_{init} (which is kept fixed in this first phase).

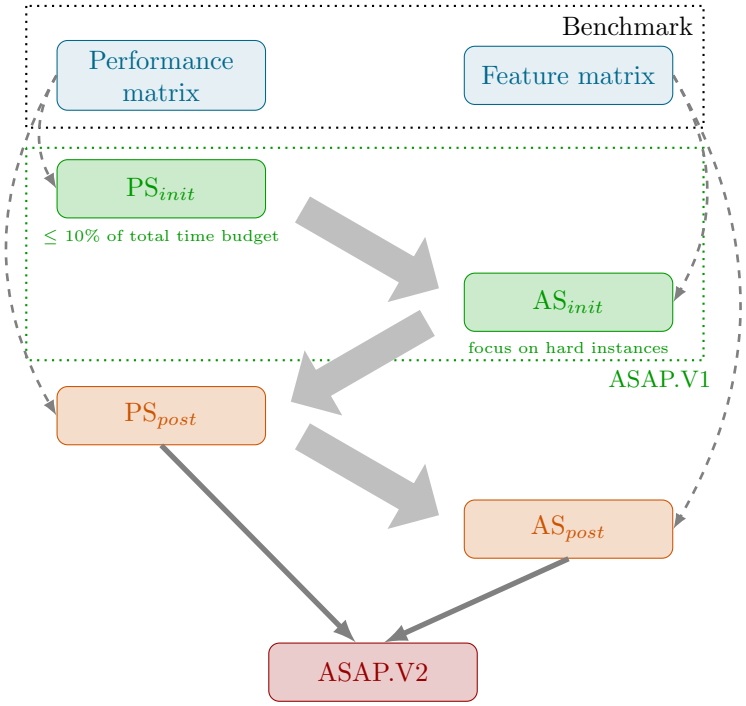

Figure 4.12: ASAP.V2 alternating optimization procedure. ASAP.V2 sequentially trains: PS$_{init}$; AS$_{init}$ (resulting in ASAP.V1); PS$_{post}$; AS$_{post}$. ASAP.V2 comprises these two last components.

In a second phase, PS$_{post}$ is determined as the solution of the following optimization problem:

$$\begin{aligned} &\underset{\tau_1,\ldots,\tau_{K-1}}{\text{optimize}}\ \mathcal{F}\left((a_1,\tau_1),\ldots,(a_K,\tau_K),(a_{\text{AS}_{init}},\tau_{\text{AS}_{init}})\right) \\ &\text{subject to}\ \begin{cases} \sum_{k=1}^{K}\tau_k = T_{ps} \\ \forall k, \tau_k \in \mathbb{R}^+ \end{cases} \end{aligned} \tag{4.9}$$

Optimization details Optimization variables are encoded in $[0,1]$. The equality constraint is handled by dropping τ_K from the set of optimization variables (see below). The actual optimization variables $z_1,\ldots,z_{K-1}$ are defined as:

$$\forall k \in \{1,\cdots,K-1\},\ z_k = \frac{1}{T_{ps}}\min\left(\tau_k, T_{ps} - \sum_{l=1}^{k}\tau_l\right)$$

The decoding procedure, required to provide the objective function with a valid set of τ_k values, implements the boundary constraints and the equality constraint. Note that the resulting optimization problem considers the absolute values of the z_ks.

$$\forall k \in \{1,\cdots,K-1\},\ \tau_k = T_{ps}\cdot\min\left(|z_k|, 1 - \sum_{l=1}^{k}|z_l|\right)$$

$$\tau_K = T_{ps}\cdot\max\left(0, 1 - \sum_{l=1}^{K-1}|z_l|\right)$$

For multi-objective optimization[16] (*i.e.* for $K \geq 3$) a prominent continuous black-box optimizer, specifically, the Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) [Hansen et al., 2003] is used to carry out the search. The initial standard deviation (parameter σ) is set to 0.25 and the number of CMA-ES inner restart procedures (increasing each time the population size) is set to 3 after preliminary experiments. The optimization starts from the initial solution where all time shares are equal.

Raw optimization criterion

This extended search space is first investigated by considering the *raw optimization criterion* $\mathcal{F}_{raw}$ that measures the PAR10 of schedule $S = (\text{PS}_{post}, \text{AS}_{init})$ over all training problem instances:

$$\mathcal{F}_{raw}(S) = PAR10(S) = \frac{1}{|\mathcal{I}|} \sum_{\mathbf{x}_i \in \mathcal{I}} PR10(\mathbf{x}_i, S) \tag{4.10}$$

As previously stated, the risk of overfitting increases as a richer search space is considered and the size of the training set remains the same (a few hundred to a few thousand problem instances, as detailed in Tab. 4.4).

Penalized optimization criterion

To prevent overfitting, a *penalized optimization criterion* is thus introduced and augments $\mathcal{F}_{raw}$ with a L_2-norm regularization term:

$$\mathcal{F}_{L_2}(S) = \mathcal{F}_{raw}(S) + w \cdot \frac{T^{max}}{T_{ps}} \cdot \|\boldsymbol{\tau}\|_2$$

which penalizes an uneven allocation of times within the presolving schedule.[17] The rationale for this penalization is to prevent brittle improvements on the training set due to opportunistic adjustments of the τ_k values, at the expense of stable performances on further instances. The penalization weight w is adjusted using a separate cross-validation process.

Randomized optimization criterion

A *randomized optimization criterion* is also considered. By construction, the ideal fitness function to be minimized is the expected performance over the problem domain. Only the empirical average performance over the problem instances is available, defining a noisy optimization problem. Sophisticated approaches have been proposed to address this noisy optimization issue in non-convex optimization-based machine learning settings (see, *e.g.* [Cauwet et al., 2016; Heidrich-Meisner and Igel, 2009]). Another approach is proposed here, based on the bootstrap principle: in each CMA-ES generation, the set of $|\mathcal{I}|$ problem instances used to compute the performance is uniformly drawn with replacement from the $|\mathcal{I}|$-size training set. In this manner, each optimization generation considers a slightly different optimization objective noted $\mathcal{F}_{rand}$, thereby discouraging hazardous improvements

[16] If the dataset considers exactly 3 algorithms, K is reduced to 2 and the problem is a 1-dimensional optimization, which is not handled by the CMA-ES implementation used. If there are only 2 algorithms – such as for the CSP-2010 dataset – K is set to 1 and the pre-scheduler optimization trivially retains the SB for a budget of T_{ps}.

[17] Since $\|\boldsymbol{\tau}\|_1 = T_{ps}$ is fixed, notice that the penalization term is minimal when $\boldsymbol{\tau} = \left[\frac{T_{ps}}{K}\right]_k$.

and contributing to a more robust search.

$$\text{draw } \mathcal{J} = \left(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_{|\mathcal{I}|}}\right) \sim \mathcal{U}(\mathcal{I}^{|\mathcal{I}|})$$
$$\mathcal{F}_{rand}(S) = \frac{1}{|\mathcal{I}|} \sum_{\mathbf{x}_j \in \mathcal{J}} PR10(\mathbf{x}_j, S)$$

Probabilistic optimization criterion

Finally, a *probabilistic optimization criterion* is considered, modeling ASAP performance on a single problem instance as a random variable with a triangle-shape distribution (Fig. 4.13) centered on the actual runtime t_{ia} required for algorithm a to solve the i-th instance, with support in $[t_{ia} - \theta, t_{ia} + \theta]$,[18] and taking the expectation thereof. The merit of this triangular probability distribution function is to allow for an analytical computation of the overall fitness expectation, noted $\mathcal{F}_{dfp}$.

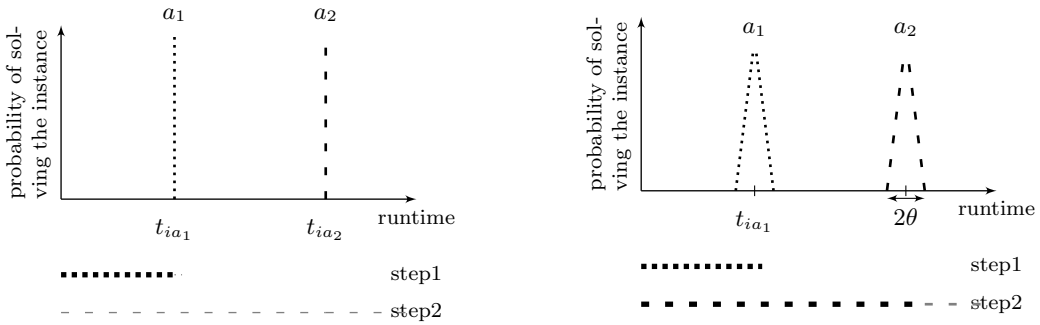

Figure 4.13: Impact of a probabilistic optimization criterion: Difference between deterministic and probabilistic execution time. Left: the schedule deterministically stops as a_1 solves the instance. Right: with some probability, a_1 does not solve the instance and the execution proceeds.

4.5.2 ASAP.V2 selector

Similarly to AS_{init} in ASAP.V1, AS_{post} is based on algorithm performance models. Only random forest models are considered in ASAP.V2 given that they outperform other ML algorithms (ASAP.V1 considered k-NN, linear SVM and Gaussian kernel SVM). However, any other PIAS approach (*e.g.* classification or pairwise regression, Sec. 3.2.4) could be used instead without requiring any change in the ASAP.V2 alternating optimization procedure.

Following ASAP.V1 post-challenge improvements on the AS component (Sec. 4.4.2), several sizes of the random forest models are considered to jointly adjust AS_{post} (with a varying number of trees) and PS_{post} (where the optimization criterion varies).

Like in ASAP.V1, the AS is learned conditionally to the presolving schedule: AS_{post} is learned conditionally to PS_{post} using pre-schedule features (Sec. 4.2.3).

4.6 ASAP.V2 experimental validation

The experimental validation of ASAP.V2 aims at answering two main questions:

[18] θ value is determined using a cross-validation; see Sec. 4.6

1. To what extent is the pre-scheduler optimization beneficial ? Along this line, the proposed loss functions introduced in 4.5.1 are compared in terms of raw performances; their impact is inspected to guide the automatic configuration of hyperparameters.

2. How efficient is the alternating optimization procedure? Do the ASAP.V2 pre-scheduler and selector actually depend on each other? The analysis of each component is undertaken to state how much each component can compensate for the errors of the other. Interestingly, it is easier to inspect the division of labor when both component are far from optimal.

This section considers two experimental settings to conduct a sensitivity analysis:

- ASlib, where the cross-validation methodology is used to validate the approach and propose robust hyperparameter values.

- the ICON challenge, that enables a fair comparison of ASAP.V2 with ASAP.V1 and other state-of-the-art systems.

4.6.1 Experimental comparison of ASAP.V2 pre-scheduler optimization loss functions

The regularized optimization criteria (Sec. 4.5.1) involve hyper-parameters (the weight w of the L_2 penalization and the width θ of the distribution for the probabilistic criterion) that must also be calibrated. Fig. 4.14 shows the sensitivity of the ASAP.V2 normalized $PAR10$ depending on the hyperparameter values, with ASAP.V1 performance as a baseline (only differing from ASAP.V2 as its pre-scheduler is not optimized). The raw criterion is also represented (with a red dot) for the extreme cases $w \to 0$ or $\theta \to 0$.

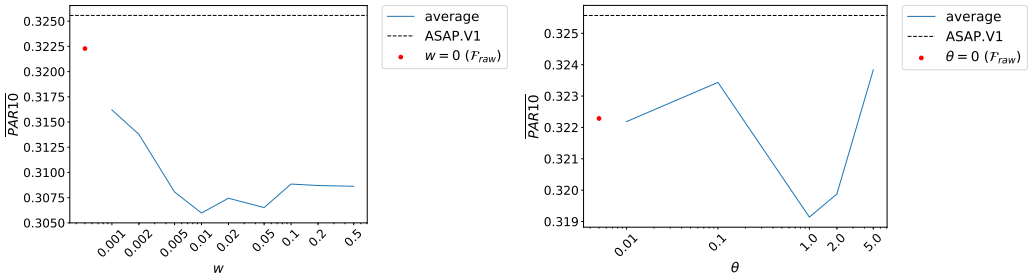

Figure 4.14: ASAP.V2: Sensitivity of the penalized ($\mathcal{F}_{L_2}$, left) and probabilistic ($\mathcal{F}_{dfp}$, right) optimization criterion with respect to their hyperparameter settings (x-axis, respectively w and θ, log-scale). For well calibrated values ($w = 0.01$ or $\theta = 1$), both options outperform the raw optimization criterion and ASAP.V1 with a non-optimized pre-scheduler.

The advantage of the proposed criteria over ASAP.V1 is summarized in Tab. 4.6. Factually, the pre-scheduler optimization with penalized criterion (AS_{init} remains unchanged) improves ASAP_RF.V1 results by about 10% (from 0.325 to 0.3059). In comparison, fine-tuning the selector alone (Sec. 4.4.2) improves ASAP_RF.V1 performance by a lesser amount (from 0.325 to 0.3073; Fig. 4.11, 200 trees). In summary, optimizing the pre-scheduler is more effective than optimizing the AS; the joint optimization of both components is considered in Sec. 4.6.3.

As stressed in Tab. 4.6, the penalized optimization criterion significantly outperforms the raw optimization criterion; this suggests that the latter is subject to overfitting. The

Optimization strategy	None	$\mathcal{F}_{raw}$	$\mathcal{F}_{L_2}$ $(w = 0.01)$	$\mathcal{F}_{rand}$	$\mathcal{F}_{dfp}$ $(\theta = 1)$
$\overline{PAR10}$	0.325	0.322	**0.306**	0.321	0.319

Table 4.6: Comparison of PS_{init} optimization strategies on the ASlib cross-validation in terms of normalized PAR10 measure (lower is better). "None" refers to no optimization, *i.e.* ASAP_RF.V1 (PS_{init} being unchanged). Standard deviation is not shown for the sake of readability; note that the folds of a same dataset are heterogeneous.

other two considered options (randomized and probabilistic optimization criterion) outperform the raw optimization criterion to a much smaller extent. According to this first experiment, they should be discarded, all the more since the probabilistic criterion involves one extra hyper-parameter.

4.6.2 Experimental validation of the alternating optimization process

Experimental settings

This section examines the impact of the joint optimization of the pre-scheduler and the AS. The gains obtained from the presolving schedule optimization conditionally to AS_{init} are inspected by comparing three variants of ASAP.V2:[19]

(O.1) : the presolving schedule is optimized independently from the selector, *i.e.* all instances that the presolving schedule does not solve are considered as unsolved. Due to the high penalty on unsolved instances with the $PAR10$ score, the presolving schedule is required to solve as many instances as possible.

(O.2) : the performance of the pair (pre-scheduler, oracle selector) is maximized (where instances not solved by the presolving schedule are handled by an oracle selector). The presolving schedule then is expected to solve the instances as quickly as possible while the others are passed on to the oracle.

(O.3) : the performance of the true system (pre-scheduler, AS_{init}) is maximized, where all instances in the training set are considered when training AS_{init} and PS_{post}. By construction, this performance is an upper bound on ASAP.V2 selector performance as it is trained and simulated on the same data. ASAP.V2 (if nothing else is mentioned) implements ((O.3)).

The difference between the first two options can be sketched as: the former attempts to solve all problem instances; the latter is in a "best effort" mode. They actually lead to different presolving schedule solutions, as shown in Ex. 4.2:

Example 4.2. *Consider 3 algorithms solving 5 instances with the integer costs reported in Tab. 4.7 (left of the double bars). Assume a 2-steps presolving schedule* $PS = ((a_1, \tau_1), (a_2, \tau_2))$ *with overall budget* $T_{ps} = 2$ *and assume that the oracle would select algorithm* a_3 *for any instance. Time spent by the presolving schedule for all combinations of* (τ_1, τ_2) *(assuming the presolving schedule gets only integer costs) are given in Tab. 4.7 (right of the double bars), with the number of stars denoting the number of instances PS failed to solve.*

Optimizing the presolving schedule alone (option (O.1)), the best solution is $(\tau_1, \tau_2) = (0, 2)$ *as the two other solutions are endorsed with a strong penalty for not solving instance*

[19] Recall that ASAP.V1 validation showed how beneficial it is to train the selector conditionally to the pre-scheduler, through the use of extra features (Sec. 4.2.3).

Algorithm	Cost a_1	Cost a_2	Cost a_3	Cost of PS for (τ_1, τ_2) $(2,0)$	$(1,1)$	$(0,2)$
i_1	1	1	1	1	1	1
i_2	1	2	1	1	1	2
i_3	1	2	1	1	1	2
i_4	1	1	1	1	1	1
i_5	3	2	1	2+*	2+*	2
Total				6+*	6+*	8

Table 4.7: Cost of all presolving schedule in Ex. 4.2, *e.g.* instance i_2 is solved by PS in 1 if in the first step a_1 is given budget 1 or 2, and it is solved in 2 if a_1 is given 0 and a_2 is given budget 2.

i_5. *On the other hand, optimizing the presolving schedule as if it were followed by the oracle (option (O.2)) would result in total cost* $6 + 1$ *for solutions* $(1, 1)$ *and* $(2, 0)$ *that would be preferred to* $(0, 2)$ *with cost* 8.

Experiments below are based on a 35-tree RF selector. This selector is preferred to better ones (Sec. 4.4.2) to better illustrate the differences between option (O.2) (selector is the oracle) and option (O.3) (selector is AS_{init}). For the same reason, the k-NN based selector is not used, as it performs perfectly on the training data due to the distance weighting in Eq. 4.6.

Analysis of the results

Most surprisingly, as shown on Fig. 4.15 most of the gains are due to the optimization of the presolving schedule runtimes. In the meanwhile, the division of labor between the pre-scheduler and the selector is not enforced by the optimization process. The optimization of the presolving schedule alone (option (O.1)) is very sensitive to hyperparameter values when considering the average across all datasets; a more detailed analysis is required.

To facilitate the comparison between the three options, performance ratios are defined to quantify the improvement on one option over another:

$$\mathcal{F}_{ratio}(X, Y) = \mathcal{F}(X)/\mathcal{F}(Y) \tag{4.11}$$

Fig. 4.16 reports $\mathcal{F}_{ratio}(ASAP.V2, ASAP.V1)$, thus comparing ASAP.V2 (option (O.3)) with ASAP.V1 as a reference. Note that some datasets (ASP-POTASSCO, SAT11-HAND, SAT12-ALL) actually do *not* benefit from the presolving schedule optimization.

As illustrated on Fig. 4.17, options (O.3) and (O.1) are equivalent for most datasets except for MAXSAT12-PMS, PREMARSHALLING and PROTEUS-2014. For those, a per-dataset pre-scheduler configuration would be relevant.

Options (O.3) and (O.2) are mostly indistinguishable, and their comparison is omitted, indicating that the presolving schedule optimization in option (O.3) is optimistic (the AS is trained and evaluated on the same data,). A more realistic setting would simulate the selector along a leave-one-out process (training the selector on all the training instances but one and then make the prediction for it). This is however hardly tractable, as it would require to train as many selectors as there are training instances. An alternative is to consider a nested cross-validation, introducing a new trade-off between the computational cost (the more folds, the more selectors to train) and the representativity of the selectors (the fewer folds, the smaller the selectors training sets are and as a consequence the poorer the selectors will be compared to AS_{init} – that is trained on the whole training set data).

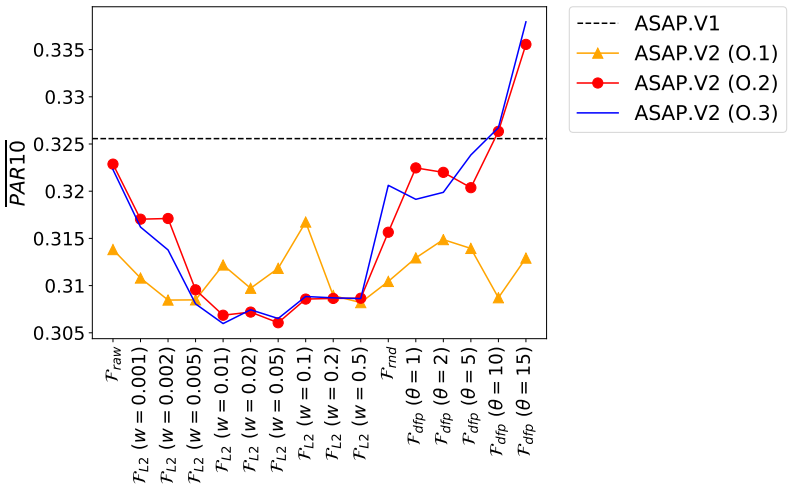

Figure 4.15: Comparison of ASAP.V2 variants (O.1), (O.2) and (O.3) in terms of $\overline{PAR10}$ (avg. across all datasets, ASlib cross-validation). The sensitivity w.r.t. the pre-scheduler optimization loss function is similar for (O.2) and (O.3). Variant (O.1) contrasts with the two others, confirming that simulating a selector component does affect the pre-scheduler optimization.

4.6.3 Evaluation in the ICON challenge framework

The sensitivity analysis conducted after the closing of the challenge compares ASAP.V2 (with different time-outs in the presolving schedule) and ASAP.V1, and examines the impact of the different optimization criteria, aimed at avoiding overfitting: the raw fitness, the L_2-penalized fitness, the randomized fitness and the probabilistic fitness (Sec. 4.5.1).

The impact of the hyper-parameters used in the AS (number of trees set to 35, 100, 200, 300 and 500 trees in the Random Forest) is also investigated.

Table 4.8 summarizes the experimental results for each ASAP.V2 configuration along the ICON challenge setting, together with the actual submissions results, including systems that were not competing in the challenge: `llama-regr` and `llama-regrPairs` from the organizers, and `autofolio-48` which is identical to `autofolio` but with 48h time for training (12h was the time limit authorized in the challenge) [Kotthoff, 2015].

The significance analysis, using a Wilcoxon signed-rank test, is reported in Fig. 4.18. A first result is that all ASAP.V2 variants improve on ASAP.V1 with significance level 1%. A second result is that ASAP.V2 with the probabilistic optimization criterion is not statistically significantly different from the challenge-winner `zilla`, `autofolio` and `zillafolio`.

Fig. 4.19 details per dataset the performance improvement between ASAP.V2 (500 trees, $\mathcal{F}_{L_2}$ version) and ASAP.V2 (500 trees, $\mathcal{F}_{dfp}$ version) and on the other hand ASAP_RF.V1 (35 trees). Note that ASAP.V2 outperforms the per-dataset best submission to the challenge for 3 datasets: MAXSAT12-PMS, QBF-2011 and SAT11-HAND.

[20] `ASAP_RF.V2` with 35 trees and no optimization in the pre-scheduler is identical to `ASAP_RF.V1`; due to small changes in the implementation however, scores slightly differ.

[21] For the CSP-2010 dataset, only two algorithms are available: the presolving schedule thus consists of a single algorithm, and all `ASAP_RF.V2` variants with the same selector hyperparameter are identical.

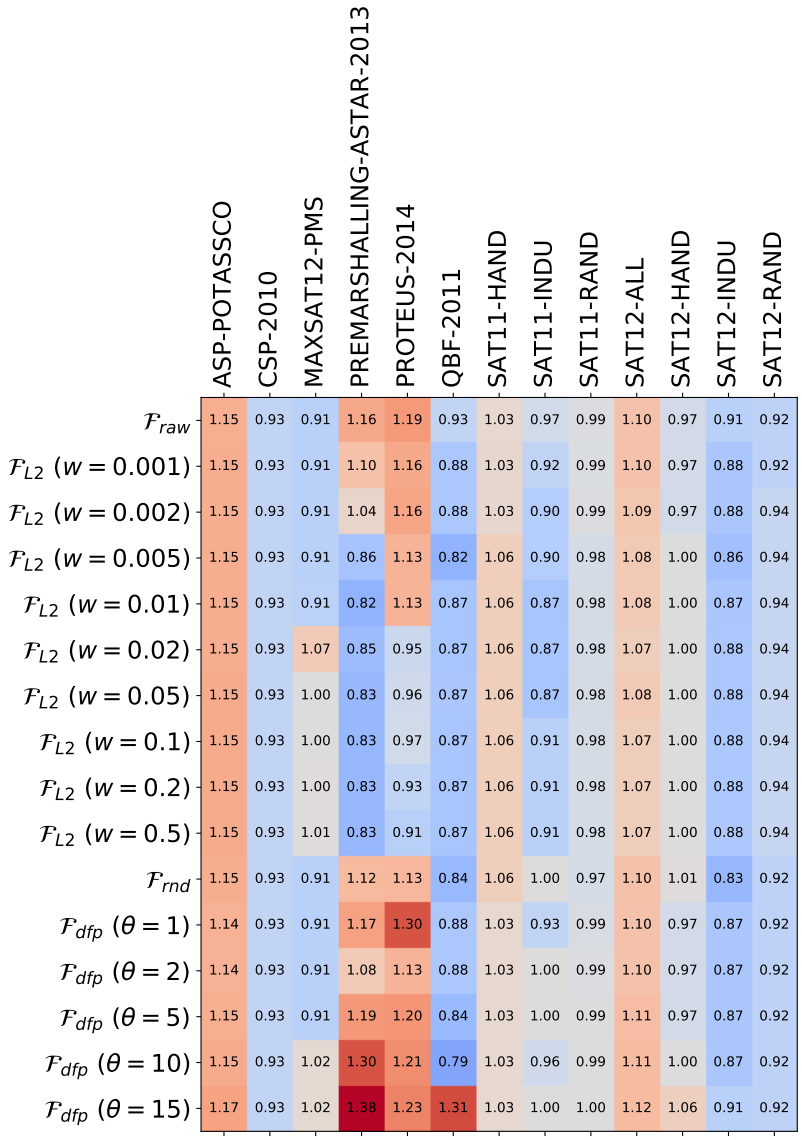

Figure 4.16: Performance ratio $\mathcal{F}_{ratio}(ASAP.V2, ASAP.V1)$ between ASAP.V1 and ASAP.V2 for each dataset. Values below 1 – blue color – indicates that ASAP.V2 outperforms ASAP.V1. Values above 1 – red color – indicate the opposite.

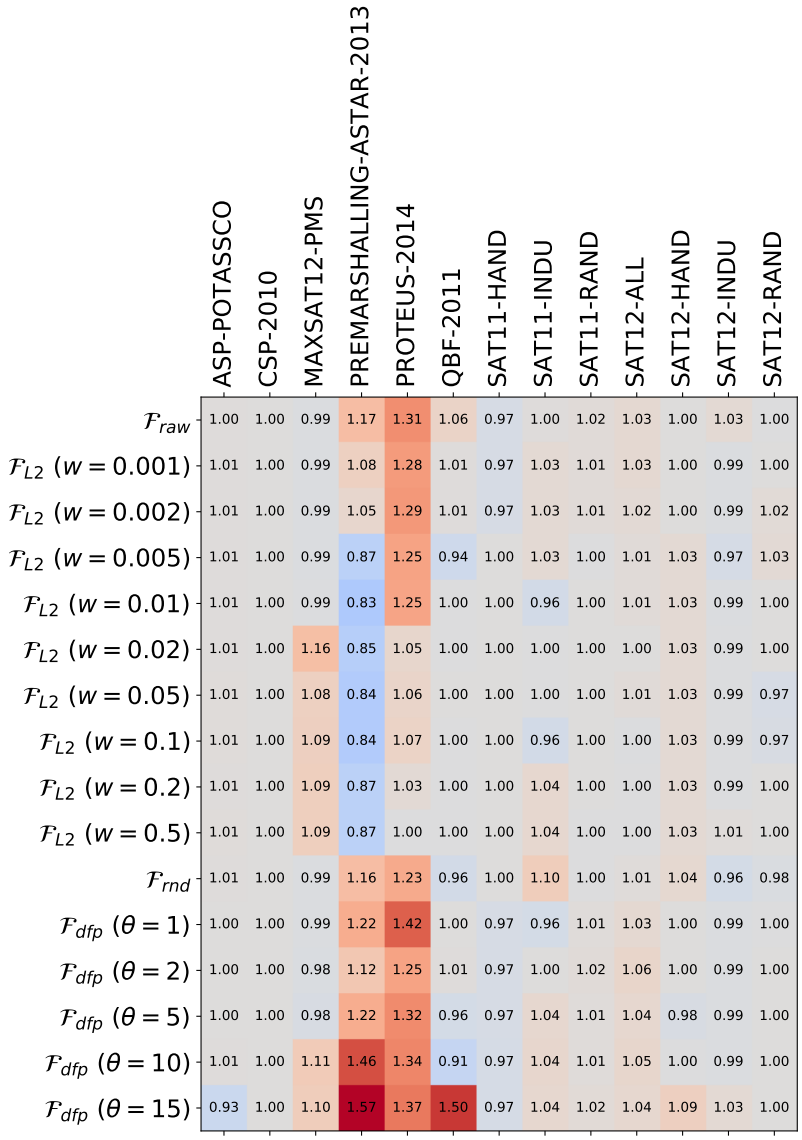

Figure 4.17: Performance ratio $\mathcal{F}_{ratio}((O.3),(O.1))$ between (O.3) and (O.1) for each dataset. Values below 1 – blue color – indicates that (O.3) outperforms (O.1). Values above 1 – red color – indicate the opposite.

4.6.4 Analysis of the behaviour of the ASAP.V2 optimized pre-scheduler

ASAP.V1 was designed based on the observation that its pre-scheduler and AS components should be complementary. ASAP.V2 strengthens the division of labor between both components through extra tuning of the pre-scheduler. Specifically, the optimized presolving schedule must: i) solve instances badly-handled by the selector and ii) solve "easy" instances as efficiently as possible. The comparison of ASAP.V2 variants and the non-optimized variant (rightmost column in Tab. 4.8, with equal time-outs) demonstrates that both goals are met to some extent.

Firstly, pre-scheduler fine-tuning does improve the presolving schedule performance; the overlap between instances that each component can solve by itself[22] (within T_{ps} for the presolving schedule, within the remaining time for the algorithm selected by the AS) tends

[22]Remind that these instances are not actually passed to the AS in the challenge evaluation setup.

fitness function (if relevant)		$\mathcal{F}_{L_2}$	$\mathcal{F}_{dfp}$	$\mathcal{F}_{rand}$	$\mathcal{F}_{raw}$	none
ASAP_RF.V2	35	0.416	0.414	0.412	**0.410**	0.414[20]
ASAP_RF.V2	100	0.404	**0.398**	0.405	0.402	0.414
ASAP_RF.V2	200	0.404	0.402	0.402	**0.399**	0.405
ASAP_RF.V2	300	0.399	0.399	0.402	**0.393**	0.405
ASAP_RF.V2	500	0.398	**0.394**	0.398	0.398	0.401
ASAP_RF.V1		0.416[20]				
ASAP_kNN.V1		0.423				
autofolio		0.391				
flexfolio		0.442				
sunny		0.482				
sunny-presolv		0.485				
zilla		0.366				
zillafolio		0.37				
autofolio-48				0.375		
llama-regrPairs				0.395		
llama-regr				0.425		

Table 4.8: Optimized presolving schedule performance[21] aggregated across all datasets, all splits and all measures (the lower, the better). The hyperparameters for $\mathcal{F}_{L_2}$ and $\mathcal{F}_{dfp}$ were chosen after preliminary experiments using the cross validation provided with ASlib. For each configuration of the selector, the best-evaluated fitness function appears in bold.

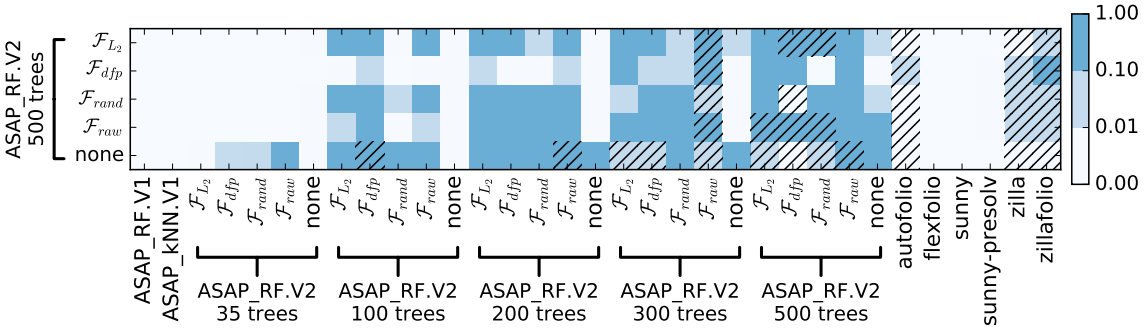

Figure 4.18: Significance analysis according to Wilcoxon signed-rank test p-value: ASAP_RF.V2 (varying objectives, 500 trees) against all other systems. Color indicates significance; hatching indicates that the row algorithm is outperformed by the column algorithm.

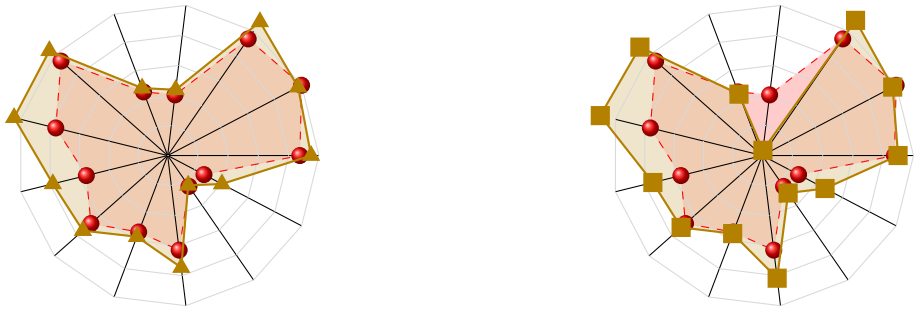

Figure 4.19: Left: Comparison of ASAP_RF.V2 ($\mathcal{F}_{L_2}$, 500 trees, triangles) with ASAP_RF.V1. Right: Comparison of ASAP_RF.V2 ($\mathcal{F}_{dfp}$, 500 trees, squares) with ASAP_RF.V1.

to decrease when optimizing the presolving schedule for most datasets, as depicted on Fig. 4.20. On the other hand, the full ASAP.V2 systems (optimized pre-scheduler + AS) solve roughly as many instances as the non-optimized setup (difference $< 1\%$). It follows that the pre-scheduler fine-tuning leads to a better specialization of both components, though it does not translate into a global improvement.

The inclusion of T_{ps} in the set of optimized variables is expected to further strengthen this specialization.

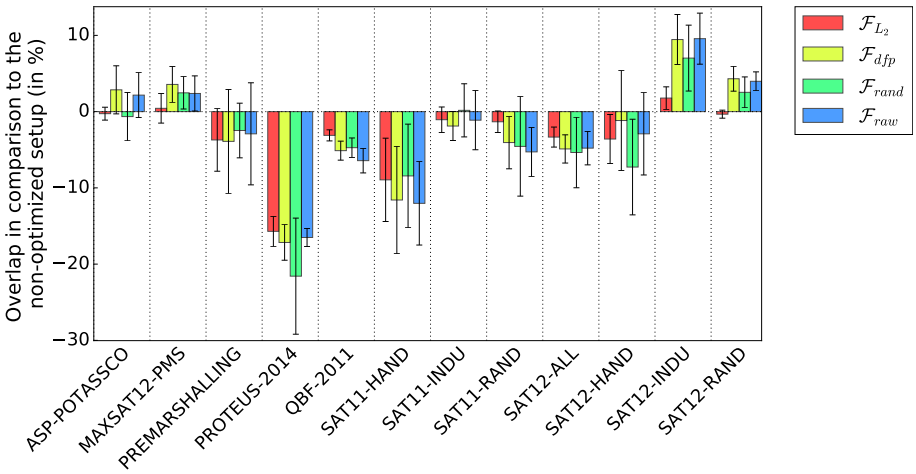

Figure 4.20: Overlap between the pre-scheduler and the selector components: per-dataset number of instances that *can* be solved by both the pre-scheduler and the selector component in comparison to the non-optimized pre-scheduler variant.

The time spent in the pre-scheduling phase is reduced (up to 29%) by the pre-scheduler fine-tuning, as illustrated in Fig. 4.21. As one could have expected, the use of the L_2 regularization mitigates this effect (we note that this setup prevents the optimized τ_k values to be far apart from the equal time-outs of the non-optimized presolving schedule): it is a low-risk, low-reward strategy. No clear winner emerges from the other 3 variants.

4.7 Open Algorithm Selection Challenge

After the 2015 ICON challenge on algorithm selection, the COSEAL group organized in 2017 the Open Algorithm Selection Challenge (OASC), which was won by ASAP.V2.

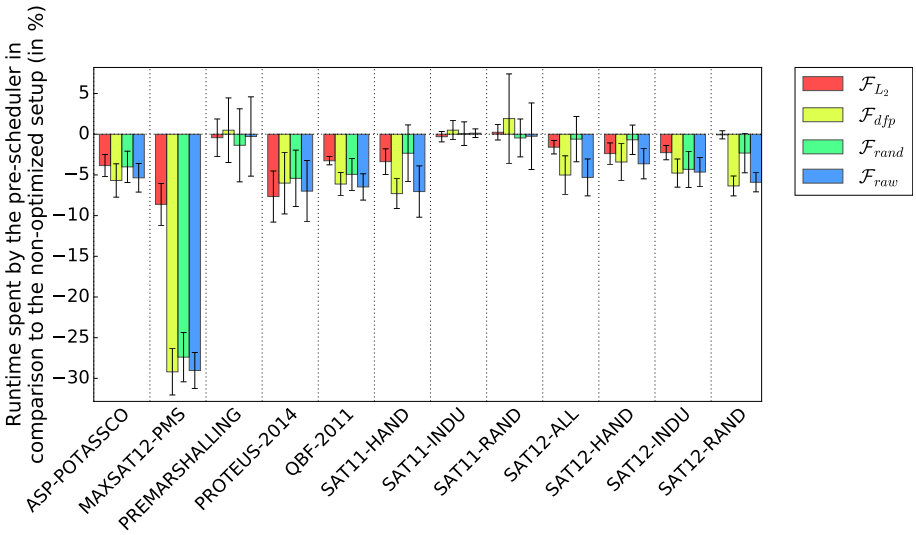

Figure 4.21: Per-dataset runtime spent in the pre-scheduler phase in comparison to the non-optimized pre-scheduler variant.

Some crucial changes have been made to the challenge compared to ICON, in order to attract a broader audience and overcome some limitations of the previous challenge. The OASC involved two phases, respectively ranging from May 20th to June 16th and from June 16th to August 27th 2017. The first phase allowed the participants to have their say in the evaluation setup, while the second phase was a usual machine learning prediction step. In this section, we firstly present the challenge settings and evaluation. A preliminary analysis of the results [Kotthoff et al., 2017] is proposed, and the section concludes with a discussion of perspectives for further portfolios approaches.

4.7.1 Challenge setting

Scenarios

The first phase aimed at building a corpus of datasets (a.k.a. scenarios – the two terms are used interchangeably) for the evaluation. Each participant had the possibility to submit at most two new datasets to be released to the community after the challenge, with two intended benefits. The first benefit is to provide the challenge organizers and the whole AS community with additional public datasets, thus broadening the diversity and representativity of the scenarios. The benefit for the participant is to provide datasets that they know well, expectedly giving them an edge over the other competitors. For this reason, only two datasets were allowed for each participant team; the final evaluation setting was meant to include 10 datasets, possibly completed with fresh new datasets provided by the organizers, or scrambled versions of already known ASlib scenarios.[23]

At the end of this first phase, 8 new scenarios had been submitted by the competing teams or by the organizers; overall, 11 datasets were selected for the evaluation. Tab. 4.9 describes these datasets and their identity – disclosed after the end of the challenge. In contrast with the ICON challenge, the train/test split were known to the participants while the performances of the algorithms on the test set were hidden. It is important to note that the participants were then aware of the amount of data used for training and evaluating.

[23]Participants were asked for a list of their preference among existing scenarios.

Dataset (anonymized name)	# instances	# features (# groups)	# algorithms	Objective
BNSL-2016 (Bado)	1179	87 (7)	8	Runtime
CSP-Minizinc-Obj-2016 (Camilla)	100	95 (1)	8	Solution quality
CSP-Minizinc-Obj-2016 (Caren)	100	95 (1)	8	Runtime
MAXSAT-PMS-2016 (Magnus)	601	37 (1)	19	Runtime
MAXSAT-WPMS-2016 (Monty)	630	37 (1)	18	Runtime
MIP-2016 (Mira)	218	143 (1)	5	Runtime
OPENML-WEKA-2017 (Oberon)	105	103 (1)	19	Accuracy (+)
QBF-2016 (Quill)	825	46 (1)	24	Runtime
SAT12-ALL (Svea)	1614	115 (10)	31	Runtime
SAT03-16_INDU (Sora)	2000	483 (16)	10	Runtime
TTP-2016 (Titus)	9720	50 (1)	22	Solution quality (+)

Table 4.9: Datasets of the Open Algorithm Selection Challenge. (+) in the objective column indicates that the objective is to be maximized (default is minimized).

Datasets Camilla, Oberon and Titus are called "quality" datasets in the following. Specific rules apply to the "quality" datasets: the schedule must consist of a single algorithm (*i.e.* it is a pure algorithm selection problem) and the feature computation cost is free.

Submission rules

As in the ICON challenge, the goal is to solve test instances with a sequential schedule of algorithms. A difference is that only the schedule is mandatory for submission. Any amount of feature computation, and of global[24] or per-instance algorithms is allowed. The training effort to produce the submissions is left to the participant without any computational constraint. The only restrictions regard the logical order of the operations: any feature-dependent system must be run *after* the required features are computed; similarly, the schedule must satisfy all precedence constraints in the feature computation steps. Note that feature computations are not provided with a cutoff time (and thus are run until completion).

A total of 8 submissions from 4 teams entered the OASC[25], all competing in the open source submission track.[26] 6 of them were based on systems that already competed for the ICON challenge, namely `*zilla`, `SUNNY` and `ASAP`. An extension of ASAP.V2 named ASAP.V3 was submitted to the challenge[Gonard et al., 2017]: the number of algorithms K in the pre-scheduler is automatically configured in $\{1, ..., 4\}$ as the best PS_{init} option over the whole training set. For "quality" datasets, the pre-scheduler was disabled; in this case ASAP.V3 and ASAP.V2 both implement the same and only AS.

4.7.2 Hyper-parameter tuning

Limited tuning effort has been carried out to adapt ASAP.V2 and V3 to the OASC datasets. Based on previous experiments (Fig. 4.11), the number of trees in the random forest models is set to 200; using more trees does not result in higher performances. To select among the different pre-scheduler optimization options, experiments with a 10-fold cross-validation on the challenge training data is performed, as depicted in Fig. 4.22.

[24] This contrasts with the ICON Challenge setting, where one global – *i.e.* fixed for the dataset – algorithm was allowed as a presolver (Sec. 4.3). It is reminded that such algorithms, sometimes termed as dataset-oblivious algorithms, can be run before computing the features.

[25] An additional rule limited to 2 the number of submissions per team.

[26] Another track was offered to commercial and non-public systems, to open the challenge and attract as many participants as possible. It turned out that all submissions were open-source.

According to the challenge setup, only runtime datasets are used.

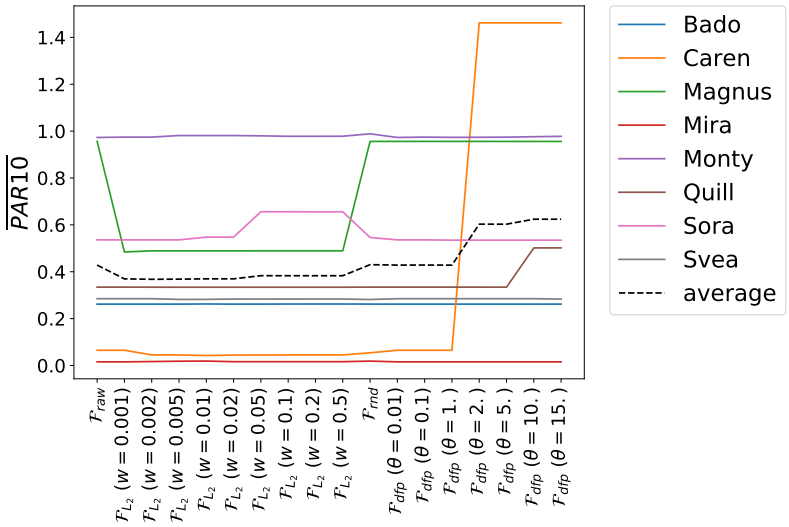

Figure 4.22: Comparison of PS_{post} optimization strategies on the OASC runtime datasets (normalized PAR10 score).

Like in ASlib V1.0.1 (Secs. 4.4 and 4.6), datasets get rise to different behaviour. Averaging over all datasets, the best option is the $\mathcal{F}_{L_2}$ loss; the average normalized PAR10 score respectively is 0.3677 for penalty weight $w = 0.002$ and 0.3684 for $w = 0.005$. Finally, $w = 0.005$ is selected, as it was a better option in the previous experiments (Sec. 4.6.1). Tab. 4.10 reports the best optimization criteria for each runtime dataset and the difference compared to the result of the best on average option ($\mathcal{F}_{L_2}$ with $w = 0.005$). As the difference is low (always under 1%), the version that performs best on average is retained for all datasets.

Dataset	**Best option**	**Improvement w.r.t.** $\mathcal{F}_{L_2}$ $(w = 0.005)$
Bado	$\mathcal{F}_{L_2}$ $(w = 0.02)$	$4.87 \cdot 10^{-5}$
Caren	$\mathcal{F}_{L_2}$ $(w = 0.01)$	$2.57 \cdot 10^{-3}$
Magnus	$\mathcal{F}_{L_2}$ $(w = 0.001)$	$4.71 \cdot 10^{-3}$
Mira	$\mathcal{F}_{raw}$	$2.76 \cdot 10^{-3}$
Monty	$\mathcal{F}_{raw}$	$8.19 \cdot 10^{-3}$
Quill	$\mathcal{F}_{L_2}$ $(w = 0.005)$	NA
Sora	$\mathcal{F}_{dfp}$ $(\theta = 1.)$	$1.12 \cdot 10^{-3}$
Svea	$\mathcal{F}_{rnd}$	$3.97 \cdot 10^{-4}$

Table 4.10: Best optimization strategy for each OASC runtime dataset (10-fold cross-validation). Last column reports the performance loss between the per-dataset best options and the best on average option.

4.7.3 Results

For each dataset, the score measures the proportion of *gap closed* in terms of PAR10, defined as:

- $1 - \overline{PAR10}$ (with $\overline{PAR10}$ defined as in Eq. 4.7) for all datasets whose performance measure is to be minimized;
- $\overline{PAR10}$ for the CSP-Minizinc-Obj-2016 and OPENML-WEKA-2017 datasets, whose performance measures (resp. solution quality and accuracy) are to be maximized.

The *gap closed* measure is 1 if the system reaches the oracle performances, 0 if it reaches the SB performances, and negative if is outperformed by the VBS. Tab. 4.11[27] reports the per-dataset *gap closed* for all systems (the winner announcement, with a less detailed version of the results can be found in [Kotthoff et al., 2017]).

	AS-ASL	AS-RF	**ASAP.V2**	**ASAP.V3**	Sunny-autok	Sunny-fkvar	*zilla	*zilla_dyn_sched
Bado	0.675	0.833	0.757	0.805	0.744	**0.844**	0.702	0.475
Camilla *	-1.289	-0.974	**0.975**	**0.975**	-0.475	0.106	-2.218	-2.218
Caren	-1.732	-1.062	0.324	0.328	0.644	**0.910**	-0.641	0.635
Magnus	-1.053	-1.052	0.498	0.496	0.492	0.572	0.575	**0.582**
Mira	-0.406	0.495	0.505	**0.509**	-0.014	0.432	0.033	-1.337
Monty	-6.389	-6.899	0.833	0.763	0.632	**0.910**	0.173	0.487
Oberon *	-9.688	-4.613	-0.404	-0.404	-0.297	**-0.164**	-0.478	-0.478
Quill	-0.299	-0.328	0.698	0.580	**0.850**	0.569	0.308	0.459
Sora	-0.669	-0.370	**0.215**	0.064	0.002	0.008	0.171	0.171
Svea	0.439	0.585	0.676	**0.688**	0.579	0.658	0.172	0.172
Titus *	-0.113	-0.535	**0.846**	**0.84**6	0.805	0.799	0.665	0.665
Average	-1.8660	-1.2655	**0.5385**	0.5136	0.3602	0.5133	-0.0489	-0.0352
Rank	8	7	**1**	**2**	4	3	6	5

Table 4.11: OASC final results. Per-dataset *gap closed* of every submission to the OASC. The best performing system appears in bold font. All datasets with a star are "quality" datasets.

ASAP.V2 is declared winner of the challenge while ASAP.V3 ranked second. Notably, ASAP versions neither rank worse than 5 (once for each system) and together establish the state-of-the art for 5 of the 11 datasets. ASAP.V2 performs strictly better than V3 on 5 datasets; ASAP.V3 strictly outperforms ASAP.V2 on 3 datasets.

Preliminary analysis

Overall, the winner of the OASC closes 54% of the gap between the SB algorithm and the oracle, while it closed 60% on the ICON challenge setup (Tab. 4.8). The tentative explanation is based on the wider diversity of the OASC datasets: SAT or CSP problems were underlying 10 ICON datasets out of 13, compared to 6 out of 11 in OASC. Furthermore, the so-called "quality" datasets appear for the first time in an algorithm portfolio challenge, to the best of our knowledge .

[27]The OASC organizers are warmly thanked for providing access to this material, unpublished at the date of writing.

On both SAT datasets Sora and Svea, `*zilla` is beaten by ASAP.V2. A revised version of `*zilla`, released after the end of the challenge closing, is still outperformed by ASAP.V2. Both ASAP.V2 and `*zilla` are based on similar principles (pre-scheduler/ presolvers and a PIAS for solving "hard" instances) and further investigations are needed to determine whether the performance differences come from the PIAS (regression-based random forests in ASAP.V2 and pairwise regression based random forests in `*zilla`) or from the additional effort to make the pre-scheduler and the PIAS complementary in ASAP.V2.

The results on "quality" datasets are especially interesting as all submitted system were primarily designed for runtime cost functions only, to the best of our knowledge. A first observation regards the small size of these datasets: Camilla and Oberon both only contain 100 instances, with 2/3rd of them accessible for training. On Oberon, all systems perform worse than the SB algorithm: all PIAS fail. It is not yet clear whether the failure is due to the limited size of the dataset, or to the very small edge of the oracle over the SB. The difference between the oracle and the SB is circa 2% in terms of normalized PAR10 on Oberon, 4% on Titus, 75% on Camilla. For comparison, the difference is one order of magnitude or more on all runtime datasets. Titus is much better handled by most systems than Oberon, though the SB edge is only slightly larger (4% instead of 2%), but it is by far the largest dataset with 9,720 instances, compared to 105 for Oberon.

The last "quality" dataset is Camilla; it is analyzed with the Caren dataset as both datasets only differ by the performance measure (with same instances and features). For the quality measure (Camilla), `ASAP` (V2/V3) dominates all other approaches with a performance of 0.975. For the runtime measure (Caren) ASAP.V2 and V3 score 0.324 and 0.328 respectively, while `Sunny-fkvar` outperforms by far all other approaches. The main lesson here is how much the performances and ranks depend on the performance measures, all else being equal. This echoes earlier conclusions (Sec. 4.2, 4.4, 4.6), stating that a *per-dataset* configuration of the portfolios is needed to deliver peak performances.

A last observation regards the lower performances of ASAP.V3 in comparison to ASAP.V2. The simple scheme implemented in ASAP.V3 to adapt K on a per-dataset basis, improving the system flexibility, do not translate into better performances (except for minor improvement on Caren and Svea, and more substantial improvement on Bado). This fact is blamed on overfitting: the number of presolving schedule algorithms should be optimized with adequate regularization, or using a cross-validation scheme.

Chapter 5

Conclusions on algorithm portfolios

Optimization, primarily a mathematical concept, is nowadays tackled as a computational problem with the use of human-designed strategies, and more recently with an increasing number of automatically designed strategies. Chap. 2 through 4 investigate algorithm portfolios, making profit of multiple different strategies in a machine learning perspective. Our main contributions are summarized and discussed in this chapter before discussing some most challenging follow-ups of this work.

5.1 Summary of contributions and discussion

Algorithm portfolios bring an answer to the long-observed fact that, often, no universal algorithm has been designed yet, even when we restrict to real-world problems. Two main directions have been considered to leverage a set of algorithms: firstly, by statistically exploiting the uncorrelated performances of multiple algorithms; secondly, by exploiting the specifics of each instance to infer which strategy is the most appropriate one to this instance.

A first remark regards the current achievements of algorithm portfolios. After the main international challenges, building upon extensive domain-knowledge and pre-existing algorithms, portfolios are increasingly able to make the best of it and to deliver peak performance. In a machine learning perspective, this implies that for some categories of problem, the mapping from problem instances onto state-of-the-art search strategies is effective. Further improvements involves the design of new search components, giving back the hand to domain-expert algorithm designers.

For some other categories of problems however, it is not the case. As shown by the Open Algorithm Selection Challenge, the current algorithm portfolios do not yet close the gap with the peak performance, even when considering a restricted set of applications. Typically, on an excerpt of the ML scenario (OpenML) no portfolio system could beat the SB strategy. Indeed, algorithm portfolios have had considerable and positive impact in the combinatorial optimization community. However, their merits in other AI contexts are less well acknowledged. The huge amount of data and algorithms already put together in platforms like OpenML naturally is a considerable asset; the design of appropriate descriptive features, however, might require more work [Muñoz et al., 2017].

On a finer grain scale, one of the contributions of this thesis is to experimentally show how heterogeneous datasets are. Like for most algorithms, the performance of an algorithm portfolio depends on its hyper-parameters; in the presented approach, the hyper-

parameter setting has been done manually for a few selected hyper-parameters. More sophisticated approaches – *e.g.*, AutoFolio [Lindauer et al., 2015] – use automatic algorithm configuration to configure their algorithm portfolios, exemplifying the use of per-instance reasoning (where an instance here is a scenario) to achieve peak performances.

The diversity of the datasets also explains why, as expected, there exists no single set of rules to design an efficient algorithm portfolio. The ASAP systems experimentally shed some light on the key design choices to obtain an efficient system for the range of datasets at hand. Some in-depth analysis with the help of domain experts is required to better understand why algorithms behave so differently within a domain; this might also inspire new algorithm design aimed at solving the problems that are poorly handled so far. This long hope (already mentioned as early as 2003 [Leyton-Brown et al., 2003]) has not been not fulfilled so far, due perhaps to the few interactions between the AS and the algorithms design communities.[1]

Another consequence of this diversity is that portfolio design itself is highly prone to overfitting. The performances of `zilla` in the ICON challenge, dominating all algorithms on SAT datasets and outperformed (by ASAP.V1) on all other datasets, indicate how biased it is toward the SAT domain.[2]

Overall, our main contributions in this first part of the thesis are the ASAP systems (Chap. 4), achieving some division of labor between a sequential schedule and a per-instance algorithm selector. ASAP.V1 delivers decent performances compared to other state-of-the-art portfolio systems and received an honourable mention in the ICON Challenge on Algorithm Selection. ASAP.V2 improves on ASAP.V1, using alternating optimization to enforce the division of labor in a dataset-dependent manner. The main efforts here regard the prevention of overfitting, through the design of specific loss functions for the sequential schedule, together with an ensemble method for the AS component. ASAP.V2 was awarded a gold medal in the Open Algorithm Selection Challenge 2017. The generality of the approach is shown as the AS also performed well on the so-called quality scenarios, demonstrating the effectiveness of the proposed learned-model based AS.

5.2 Future directions

In the short term, a first research perspective is to strengthen the division of labor between the pre-scheduler and the AS, by considering a better AS. As of now, the AS is optimistic, as its model has been learned on the entire training data. A nested cross-validation process will provide more robust estimates and enable the pre-scheduler optimization to cover up for the AS mistakes in a more reliable manner.

A second direction of research concerns the learned models themselves. As said, ASAP focuses on the single performance model-based algorithm selection technique and only consider the performance ordering among the algorithms. The lessons learned from the challenges and from complementary experiments suggest that a cost-sensitive learning-to-rank setting might be more appropriate to achieve per-instance algorithm selection; this is supported for instance by `SATzilla`'s performance in the ICON challenge (Sec. 4.4). Such an extension can be implemented in ASAP in a straightforward manner, due to the modularity of the AS and the sequential schedule. In a longer-term perspective, the

[1] Portfolios have been marginalized to only compete in the "No-Limit" (previously "Open") track of SAT competitions since 2013, although much research in the AS area is coming out of the SAT community.

[2] The domination of the SAT competitions by `SATzilla` during nearly a decade (as mentioned in Sec. 3.3.1) is arguably a strong legacy for `zilla` algorithms.

choice of the underlying AS model will be achieved in a dataset-dependent way, along an automatic configuration strategy. Indeed, automatic configuration appears a research priority for algorithm portfolio design (Chap. 3), and its impact on, *e.g.* the configuration of the overfitting mechanisms implemented in ASAP.V2 pre-scheduler (Sec. 4.5) is yet to be determined.

A third perspective regards the exploitation of the performance models to improve PIAS. Insofar, they are only used as black-box predictors. However, characterizing their accuracy and the variance of the error w.r.t. the feature space opens the way toward a per-instance value-risk approach. Indeed, one might prefer selecting an algorithm which is very likely to solve the instance for a moderately high runtime, rather than another algorithm with lower predicted runtime but with a high error variance in the (feature space) neighborhood of this instance. Note that same ML-based approaches can be used to train the performance models and the risk models [Papadopoulos et al., 2002]. A major constraint on such value at risk strategies regards the amount of data needed to train both models, limiting this approach to the largest datasets.

Finally, an utmost challenging direction of research regards the assessment of portfolio systems with respect to the required data resources. Experiments have extensively shown the risk of overfitting due to the reduced size of certain datasets. A most needed direction of research for AI (though perhaps not the most popular one at the moment) regards how to make best use of as little data as possible. Most algorithm portfolio systems, including ASAP, are trained from the complete performance archive (a # instances $\times$ # algorithms matrix). Getting such extensive and computationally demanding data is not even imaginable in some AI domains, such as machine learning. Two ways of addressing this limitation have been considered to our best knowledge. The former one is based on Bayesian optimization – used in [Kotthoff et al., 2016; Feurer et al., 2015] for solving the *combined algorithm selection and hyperparameter optimization problem* in the domain of machine learning –, where the algorithm portfolio interleaves the recommendation of promising algorithms for the current instance and the update of the per-instance performance model. The latter one is based on collaborative filtering [Stern et al., 2009; Mısır and Sebag, 2017]. On the one hand, this latter approach only requires a fraction of the overall instance $\times$ algorithms matrix to be available. On the other hand, it requires to fully reconsider the division of labor and the design of both pre-scheduler and AS components.

Part II

Recommender systems for employment

Chapter 6

Introduction to job-applicant matching

As many economic sectors have been impacted by data science, ranging from commerce to tourism and from travel to education, a question is whether and how the field of employment can also benefit from the conjunction of extensive data resources and machine learning algorithms. All these applications face the same challenge: the profusion of items prevents any user from making fully informed choice, as the amount of data she would need to review is overwhelming.

Automatic job-applicant matching (Jam in the following) aims to select for each user a personalized list of recommended jobs, from the whole set of job ads an online recruitment platform may gather.

Jam is tackled here from the *item cold-start* perspective, *i.e.* the objective is to recommend brand-new job ads (the items) to users that have already interacted with the system.

This chapter is organized as follows. The formal background of recommendation systems is presented first. Secondly, in order to tackle the cold-start recommendation setting, it is mandatory to exploit the natural language descriptions of the job ads; the common natural language processing (NLP) approaches used to process job ad documents are thus presented. Work related to the Jam context is finally presented and discussed, distinguishing two user modeling approaches. The first of these approaches considers a single representation of the user, and recommends items that are *on average* relevant for this representation. The second approach builds multiple representations of the user and recommend items with a good match to at least one of the user's representations.

6.1 Context and scope of the work

This work focuses on the *frictional unemployment* phenomenon, manifested as significant numbers of job positions are unfilled, although there exists significant numbers of unemployed people who are qualified for these jobs [Mortensen and Pissarides, 1994]. Frictional unemployment is tentatively explained by the cost and asymmetry of information for recruiters and applicants. The information issue at the root of frictional unemployment is tackled in this work using Natural Language Processing (NLP) based techniques, aimed to automatically match job ads and resumes.

Several approaches have been proposed to facilitate the recruitment task, using diverse types of data to describe applicants: curriculum vitae [Malinowski et al., 2006], track record of past jobs [Paparrizos et al., 2011] and/or profiles in social networks [Ma

et al., 2015]. However, such resources might be insufficiently informative for those who need them the most. Unemployment is known to mainly affect both categories of unskilled people, and young people. In the category of young people with no degree, the unemployment rate exceeds 50% (2014 in France[1]). Yet, these people have a resume with moderately informative content, neither including any information about academic degrees or diplomas, nor on past jobs and track records. A JAM system thus finds little signal in these resumes. When considering young, highly skilled people, a JAM faces another kind of difficulty: applicants might describe their expertise using rare words (*e.g.* the title of their PhD), whereas a job advertisement might describe the sought skills using an entirely different vocabulary.[2]. Domain resources such as field-based ontologies can be used to enrich the documents and facilitate their matching, though they face some limitations due to the rapid pace of change in science and technology, and the emergence of new jobs.

The goal of this work is to learn a JAM, automatically recommending job ads to applicants. Naturally, user choices heavily depend upon individual preferences and skills, making JAM a *personalized* recommendation problem. Unlike most recommendation problems, JAM comes rarely as a *warm-start* problem: after an open position has received sufficient attention and attracted some relevant applications, it is most likely filled and closed for recommendation. Using the terminology defined in 3.2.1, JAM is called an item cold-start recommendation problem (respectively a full cold-start recommendation problem): brand new jobs are recommended to known applicants (resp. new labour market entrants). While the recommendation of job ads to applicants and the recommendation of resumes to recruiters are formally equivalent problems, both problems are different in practice and only the former recommendation problem is considered in the remainder.

Our study has benefitted from two large-size proprietary databases. The first database concerns low-paid jobs and unskilled people. Indeed, the low-paid job sector is less profitable than the high-paid job sector, and has been largely neglected in the literature [Malinowski et al., 2006; Malherbe et al., 2014; Carpi et al., 2016]; it also presents specific difficulties due to the low signal-to-noise ratio in the resumes. Note that the item cold-start recommendation mode is relevant to the JAM context in the case of the temporary work sector, where applicants interact with the hiring agency on a regular basis and the current recommendations exploit the past interactions. The second database concerns PhD-level resumes and industrial job ads targeted at PhDs. Each database includes the resume and job ad corpora, plus user applications (recorded clicks) on job positions. The interaction data is referred to as collaborative filtering matrix.

6.2 Related Work

The field of automatic job/applicant matching is hindered by the lack of extensive public resources, able to foster algorithm design and assessment on a grand scale, as was done for the field of computer vision by the ImageNet dataset [Krizhevsky et al., 2012]. This lack of publicly available data resources is explained from the privacy requirements, and the difficulty of resume anonymization. To the best of our knowledge, the only large scale public JAM databases were proposed in the frame of the RecSys 2016 and RecSys 2017 challenges [Abel et al., 2016, 2017]. The RecSys 2016 (respectively 2017) database

[1] `http://cache.media.education.gouv.fr/file/etat25/39/3/depp-etat-ecole-2015-diplome-exposition-chomage_484393.pdf`

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

includes over 1 million job ads (resp. 1.3 million), 780,000 resumes (resp. 1.4 million), and their interactions available for circa 0.01% of the resume/job pairs (resp. 0.001%) [Carpi et al., 2016]. The anonymization was enforced through encoding CVs and job ads using 90,000 binary features (replacing raw text by numerical identifiers), and by additionally perturbing the data by adding and removing clicks.

Early related work was focused on NLP, considering manually designed features [Malinowski et al., 2006] or bag-of-words representation [Malherbe et al., 2014] (see Sec. 6.4), and enriched using domain resources such as job ontologies. On top of these representations, a similarity measure was designed or learned to estimate the relevance of a resume w.r.t. a job ad, possibly taking into account the resume structure through weighted similarities on the resume subparts [Malherbe et al., 2014].

The JAM problem features specific characteristics compared to "pure" NLP problems such as Information Retrieval (IR) [Büttcher et al., 2010] and Recognizing Textual Entailment (RTE) [Dagan et al., 2006]. Indeed, a JAM could learn whether a resume is "relevant" to a job query, or whether the skills required by a job can be "entailed" by a resume; however, the resume and the job ad are of similar size, whereas a document is usually much more detailed than a query in IR (resp., than a statement in RTE). Another difference is that the notion of relevance or entailment could in principle be logically inferred. Quite the contrary, the match between a user and a recruiter depends on hidden information (*e.g.* the applicant's and recruiter's positive or negative biases). In other words, what people do (clicking on job ads or resumes) can hardly be logically inferred from what they say (the contents of the resumes and job ads).

Cold-start recommendation (resp. item cold-start recommendation) is only made possible by the so-called "side information" describing users and items (resp. items). We emphasize that the most strict definition of cold-start recommendation (resp. item cold-start recommendation) is considered in the following, that is, recommendation of brand new items to brand new users (resp. known users) [Schein et al., 2002]. In the JAM context, a job position (respectively an applicant) is associated with a job ad (resp., a resume) usually augmented with a job posting date and geolocation. The JAM problem thus defines a *collaborative retrieval* task [Weston et al., 2012].

As far as JAM tackles cold-start recommendation, approaches relying exclusively on the collaborative filtering matrix, *e.g.* based on matrix factorization [Koren and Bell, 2015; Weston et al., 2012], on restricted Boltzman machines [Larochelle and Murray, 2011], or learning a continuous embedding of the items [Grbovic et al., 2015] are not applicable. Some representation capturing both the collaborative filtering information and the user/item description must thus be defined or learned. In early approaches [Färber et al., 2003; Malinowski et al., 2006], manually designed text features have been defined and used to recode the collaborative filtering matrix, defining the probability of a resume to be clicked upon by a recruiter, conditionally to the presence of such features in respectively the resume and the job ad. This more general representation of the collaborative filtering matrix thus allows one to both take into account the textual content-based features, and address the cold-start recommendation problem.

Overall, JAM systems involve two issues: finding representation(s) for resumes and/or job ads, and an affinity measure on the top of these representations, amenable to predict whether a (resume, job ad) pair is well-suited to each other. Related work along these directions builds upon the continuous language modeling, mapping words [Mikolov et al., 2013a; Pennington et al., 2014], sentences or documents [Le and Mikolov, 2014] onto vectors. A distance between documents is unduced by the NLP embedding into a Euclidean space, or can be defined using Word Movers Distance [Kusner et al., 2015]. Such continuous

embeddings can thus be used for semi cold-start recommendation. Another possibility builds upon supervised learning and ranking [Weinberger and Saul, 2009; Burges, 2010], where the metric on the Euclidean space is optimized to maximize a classification or ranking criterion. Notably, Siamese neural networks [Bromley et al., 1993; Chopra et al., 2005] optimize the input embedding in a continuous space w.r.t. the associated metric [Wang et al., 2014; Hoffer and Ailon, 2015]. Such approaches have been used to achieve job title normalization and classification [Neculoiu et al., 2016] or learning sentence similarity [Mueller and Thyagarajan, 2016]. In Information Retrieval, Deep Structured Semantic Mode (DSSM) optimizes the similarity in the latent space according to clickthrough data [Huang et al., 2013] (see also [Zhu et al., 2016]). Multi-view DNN [Elkahky et al., 2015] further extends DSSM to non-Siamese architectures, where different types of information involved in the query/documents are associated with different embeddings.

6.3 Background on recommender systems

A recommender system constructs a bridge between user individual tastes and features, and sets of potentially interesting items. Notations and formal background are introduced in this section; basic concepts more specifically relevant to the JAM problem are presented thereafter.

6.3.1 Problem statement and notations

As previously stated, a recommender system (RS) aims at making new connections between large and heterogeneous sets of users and items, specifically selecting a subset of items to be presented to each user on an individual basis. This problem can be formalized as a bipartite graph, where the two sets of nodes represent users and items, and the edges represent their connections. An alternative representation is in matrix form:

- $\mathcal{U}$ is the set of users. $\mathbf{x}_u$ is a representation of a user u, *i.e.* the associated resume in a JAM problem.
- $\mathcal{I}$ is the set of items. $\mathbf{y}_i$ is a representation of an item i, *i.e.* the job description in a JAM problem.
- $\mathcal{M}$ is the $|\mathcal{U}| \times |\mathcal{I}|$ interaction matrix (a.k.a. collaborative filtering matrix a.k.a. matrix of connections): $\mathcal{M}_{u,i} = 1$ if user u made a connection with item i and 0 otherwise.

Classically, $\mathcal{M}$ is only partially observed (most elements $\mathcal{M}_{u,i}$ are unknown), and some new connections $\mathcal{M}_{u,i}$ would take place if user u were (made) aware of item i. The RS goal is to bring relevant items to user attention, providing every user u with a personalized recommendation list $\{i_1, ..., i_k\}$. The recommendation list is deemed relevant if u were to draw many connections with items among the recommendation list (more on the evaluation of recommender systems in Sec. 6.3.4).

Naturally, the JAM problem considered here constitutes only one of the phases of the recruitment process. The symmetrical problem (recommending relevant resumes to the stakeholder) is not considered; the bilateral selection problem [Pizzato and Bhasin, 2013] is not considered either. The JAM system only aims at bringing relevant job ads to the job seeker's attention. The overall recruitment process is out of the scope of this work.

6.3.2 Terminology

Recommendation systems are categorized depending on the information and data used to make personalized recommendations.

Collaborative filtering

Pure collaborative filtering (CF) methods [Herlocker et al., 1999; Sarwar et al., 2001; Deshpande and Karypis, 2004; Rendle et al., 2009] use the interaction data of the community of users to suggest new connections. Such methods, however, ignore any side data (*e.g.* user profile or item description). They may come in two modes: *user-oriented* or *item-oriented*.

In *user-oriented* collaborative filtering, recommendations for a user are built from connections similar users made (see Fig. 6.1, left). The similarity measure between two users is derived from the interaction matrix (specifically, from the rows of $\mathcal{M}$). Depending on the approach, all similar users or only the most similar q users are considered to build the recommendation. Note that popular items (*e.g.* i_5) are likely to be recommended to anyone, since it is likely that at least one user similar to the target user u_6 has interacted with it.

Item-oriented collaborative filtering follows a similar reasoning and recommends items similar to those the user already interacted with (Fig. 6.1, right). There again, the item-item similarity is derived from past interactions, *i.e.* from the columns of $\mathcal{M}$.

Note that interactions data are used twice: Firstly, they serve to build a similarity among users or items. Secondly, they are used as a filter to build the recommendation list.

Pure CF approaches are limited to the warm-start setting. A new item cannot be recommended as it has not been selected by any user. In user-oriented CF, it is always rejected in the filtering phase because no user has already selected it. In item-oriented CF, it is deemed to be dissimilar to all other items. For similar reasons, a new user cannot be proposed relevant personalized recommendations.

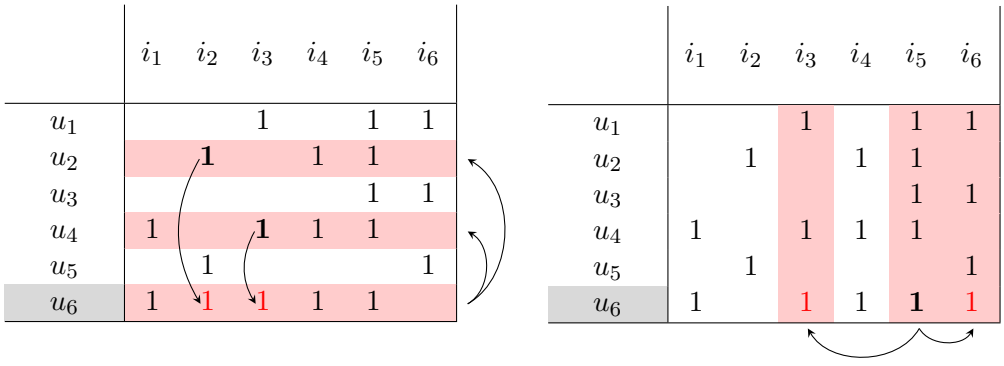

Figure 6.1: How *user-oriented* (left) and *item-oriented* (right) collaborative filtering methods make recommendations (red "1") for user u_6. Red (shaded) areas indicate which users (resp. items) the system found similar based on interaction data. Arrows indicate the past interactions (bold "1") used to select recommended items.

Content-based recommendation

In contrast, content-based recommendation (a.k.a. content-based filtering or information filtering) [Pazzani and Billsus, 2007; Lops et al., 2011] only considers the users and items side information (user profile, item description) to make recommendations, *e.g.* through

matching attributes of users and items as illustrated on Fig. 6.2. The interaction data are not used to form the recommendation lists, although they may be used as part of the user profile construction. In the considered context, where the resumes happen to be written by non-experienced people, these resumes might be insufficient to retrieve relevant job ads. It is commonly acknowledged that the same job experience may be described in many different ways and that the use of inadequate vocabulary and style (not even mentioning grammatical and spelling correctness) may be crucial for the recruiter. The same goes for a content-based system: unusual vocabulary in a user may significantly hinder the recommendation process.

However, content-based recommendation can handle the cold-start setting, and it can advantageously combine several types of data: textual, categorical (*e.g.* in large domains such as human resources or aeronautics industry), discrete ordered (*e.g.* levels of experience) and continuous (*e.g.* geolocation).

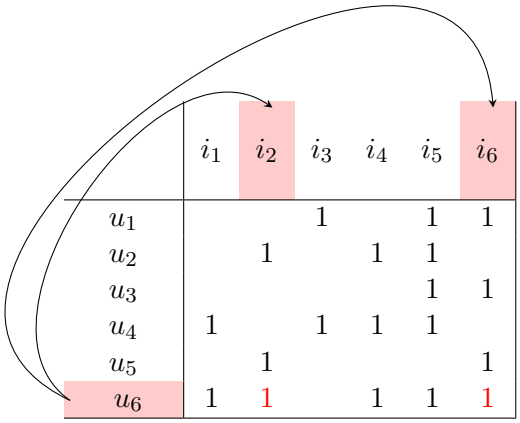

Figure 6.2: Pure content-based recommendation methods: recommendations (red / shaded "1") for user u_6 are only based on the item-user content matching.

Hybrid systems

Intermediate approaches have naturally been developed to leverage both interaction and content information [Balabanović and Shoham, 1997], and to tackle the cold-start setting. An example thereof is illustrated on Fig. 6.3. Hybrid systems expectedly provide more accurate recommendations since they exploit more and diverse data, representing different aspects of the same, *e.g.* user.

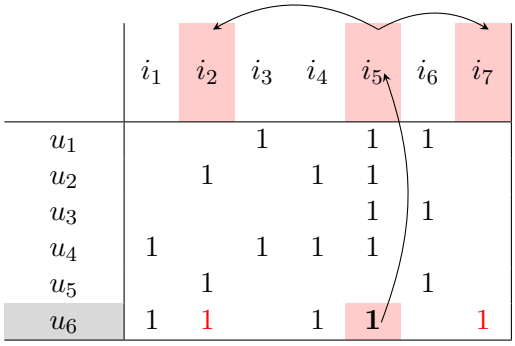

Figure 6.3: An item-oriented hybrid system uses interaction data to filter the list of items depending on the target user (u_6); items similar to the filtered items (i_5) are thereafter retrieved based on their content information, and are recommended.

6.3.3 Framework: one-class collaborative filtering

The JAM problem features a specific characteristic: The connection between a user and an item is binary, with different semantics on the "yes" (user's application on the job position) and the "no" (irrelevant or unobserved interaction). The JAM problem must therefore be cast as *one-class collaborative filtering* [Pan et al., 2008] (abridged OCCF in the following), a recommendation problem framework where only the positive interactions are known. Specifically, the absence of interaction is interpreted as the user being truly not interested ("negative interaction") *or* the user being not aware of the item (missing data).

The lack of explicit negative feedback forbids the use of methods that leverage the difference between positive and negative feedback (*e.g.* like/dislike, high/low rated items). In principle, additional data, *e.g.* whether the item was displayed to the user, could be exploited to infer negative feedback; however, such data are usually not available. Moreover, as noted by [Joachims et al., 2005], users usually consider but the top-ranked recommended items, thus making even more fuzzy the distinction between true "zeroes" and missing data. We shall come back to the interpretation of the feedback of user in Sec. 8.2.5.

6.3.4 Evaluation

Recommendation systems are most generally assessed in an *offline* setting, akin to a crossfold validation procedure. The usual experimental setting in the scientific community does not include the *online* assessment of the recommendation algorithms by a significant number of users. A notable exception is that of the RecSys challenge 2017: After a first phase, twenty recommender systems (RS) were selected and the actual recommendations of these selected systems were displayed to real users of the Xing platform (a professional network partner of the challenge) to assess the competitors.

In the remainder of the manuscript, unless mentioned otherwise, RS are evaluated in an offline setting, where training and test data are disjoint subsets of the same database. In particular, despite the previously mentioned ambiguity about the interpretation of a "no" interaction, recommending a "no" interaction will be considered an error.

Recommendations are rarely assessed in terms of accuracy, as the positive interactions outnumber the negative ones by several orders of magnitude. In [Schein et al., 2002], several RS performance indicators are discussed with a special focus on the cold-start setting. These performance indicators measure the number and position of positive interactions within the recommendation list. Note that these performance indicators implicitly take $\mathcal{M}$ as "ground truth" data. However, by construction, $\mathcal{M}$ is usually acquired as the result of the users querying the platform through a job position search – which involves an information retrieval engine – or being presented a specific fraction of the job ads – sorted by date, by popularity, or involving a first personalized RS –, inducing a considerable bias[3] on the evaluation process. Formally, the performance indicators actually measure how well a RS i) retrieves the items that were actually presented to the user; ii) selects the relevant items in the presented ones.

The main performance indicators naturally measure the RS *usefulness* w.r.t. the recommendation task, *i.e.* the selection of relevant items. Still, some other RS properties are worth evaluating depending on the context. [Herlocker et al., 2004] identify some of those, such as covering all good items, producing consistent sequence of recommendations or helping other users. Other measures may also be of interest depending on the context:

[3]No real user could see a decent *random* fraction of the items given the number of items.

diversity, trust in the system, serendipity... Only standard, relevance-driven performance indicators are considered in the following. These so-called *classification accuracy metrics* [Herlocker et al., 2004] measure the quality of the top-T elements in the recommendation list, relevant to both OCCF setting and the close field of information retrieval.

Precision Precision@T (denoted $P@T$) measures the proportion of relevant recommendations for user u within the top-T recommendations for that very user. Precision is averaged over the set of users. Letting $\mathcal{R}_T(u)$ denote the top-T recommendations for user u, it is defined as:

$$P@T = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\{\mathcal{M}_{u,i} = 1 | i \in \mathcal{R}_T(u)\}|}{T}$$

Precision is typically used in information retrieval. Its weakness is i) to be poorly informative when $\mathcal{M}$ is very sparse; ii) to be very sensitive to the way interactions are counted.

Recall Recall@T (denoted $R@T$) measures the proportion of relevant items for user u recovered within the top-T recommended items for that very user.[4] Recall is averaged over the set of users. With same notations as above, it is defined as:

$$R@T = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\{\mathcal{M}_{u,i} = 1 | i \in \mathcal{R}_T(u)\}|}{\sum_{i=1}^{|\mathcal{I}|} \mathcal{M}_{u,i}}$$

Recall is not affected by the sparsity of $\mathcal{M}$. Note that the Recall@T for a single T value does not allow for distinguishing high-rank and low-rank relevant recommendations. Still, the order in the recommendation list is known to be very important for the user [Joachims et al., 2005]). A finer-grained performance measure characterizes the so-called *recall curve*, displaying the Recall@T performance for varying values of T.

Another performance measure is based on the precision-recall curve, reporting the set of (R@T, P@T) points as T varies. The precision-recall curve of an oracle RS jumps from (0,0) to (0,1) and then to (1,1) as all relevant items and only those are top-ranked in the recommendation list; then, recall stays to 1 while precision drops towards 0.

Average precision summarizes the precision-recall curve by its integral (in $[0, 1]$ since the oracle precision-recall has integral 1). **Mean Average Precision** averages the average precision over all users.

F1-score The F-score aggregates the precision and recall measures. The F-score parameterized by T is defined as:

$$F_\beta@T = \frac{(1 + \beta^2) \cdot P@T \cdot R@T}{\beta^2 P@T + R@T}$$

where β is a positive parameter. F1-score is obtained with $\beta = 1$; it is maximal if both precision and recall are themselves maximal.

[4] [Herlocker et al., 2004] also define a Receiver Operating Characteristic (ROC) very similar to recall: the proportion of relevant documents among the top-T items that are recommended and received an evaluation (positive or negative) – note how non-rated items are excluded from this ROC measure. This kind of metrics is irrelevant in OCCF as any kind of negative feedback is missing.

Discounted cumulative gain Discounted cumulative gain (DCG) takes into account the rank of the relevant recommendations in recommendation list $\mathcal{R}_T(u)$: the higher the rank, the better. Letting $r_T(u)$ be the T-th recommendation for user u, we define the $DCG@T$ as in [Weimer et al., 2008]:

$$DCG@T(u) = \sum_{l=1}^{T} \frac{2^{\mathcal{M}_{u,r_l(u)}} - 1}{\log(l+2)}$$

Since we are considering $\mathcal{M}$ with binary values, an equivalent and simpler definition is:

$$DCG@T(u) = \sum_{l=1}^{T} \frac{\mathcal{M}_{u,r_l(u)}}{\log(l+2)}$$

Let denote $DCG^*@T(u)$ the DCG of a perfect recommendation list (all $\|\mathcal{M}_{u,.}\|_1$ relevant items are positioned as the best recommendations). A normalized version, called normalized DCG and denoted NDCG, takes value in $[0, 1]$:

$$DCG^*@T(u) = \sum_{l=1}^{\|\mathcal{M}_{u,.}\|_1} \frac{1}{\log(l+2)}$$

$$NDCG@T(u) = \frac{DCG@T(u)}{DCG^*@T(u)}$$

Mean Reciprocal Rank Mean Reciprocal Rank (MRR) takes into account the position of the first relevant recommendation within the recommendation list

$$MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{\min\left\{l | \mathcal{M}_{u,r_l(u)} = 1\right\}}$$

Note that MRR requires to compute the recommendation list until all users have at least one relevant recommendation.

Mean Average Precision Average precision summarizes the precision-recall curve by its integral. As said earlier, the oracle precision-recall delimits the unit square and corresponds to integral 1.

6.4 Natural Language Processing for document representation

This section is devoted to the main steps involved in the representation of the documents (job ads and resumes) from the textual data.

The initial step, not detailed here, consists of extracting "words" from the documents.[5] There exists an extensive literature on the *words* representation in NLP; the interested

[5] In practice, this extraction most often relies on empirical rules, *e.g.* a word is a character string delimited by specific characters like punctuation or whitespaces. An alternative is based on the n-gram extraction (see, *e.g.* [Mcnamee and Mayfield, 2004]) at the character level: a "word" consists of a sequence of n characters including whitespaces. For instance string "my name is" would be associated the 10 "3"-grams "_my", "my_", "y_n, "_na, "nam", "ame", "me_", "e_i", "_is" and "is_" (where the underscore character symbolizes a whitespace). Note that whitespaces are added at the beginning and end of the string.

reader is referred to the Probabilistic Feed-forward Neural Network [Bengio et al., 2003], Deep Neural Networks with Multitask Learning [Collobert and Weston, 2008], Word2Vec [Mikolov et al., 2013b], GloVe [Pennington et al., 2014] and fastText [Bojanowski et al., 2017] for more detail. The rest of the section focuses on *unsupervised* textual representations, and their use within a recommender system.

6.4.1 Bag-of-words

Bag-of-words (BOW) is the simplest way to map a document onto a vector with same dimension as the vocabulary size. A BOW transformation involves two steps:

1. A dictionary of all D words contained in (at least one) a document is built, with D in the dozen thousands;
2. Each document is encoded as a vector of size D where the d-th coordinate is non-zero iff the d-th word appears in the document.

Most usually, the vectorial representation of documents is sparse, as any particular document contains a very small fraction of all dictionary words. Technically, documents are coded as a sparse matrix (only non-zero elements being kept in memory) for efficiency. A difficulty resides in the distribution of word occurrences, known as the Zipf's law [Powers, 1998]: the frequency of a word decreases as a power law of its rank. As most words appear extremely rarely, it is hard to extract meaningful information from them from a machine learning perspective. In contrast, a few words are extremely frequent and blur the signal.

Another drawback of BOW and representations built thereupon is that they do not reflect the order of the words in the document, making them inappropriate for complex NLP tasks such as translation of understanding (typically, sentences such as "The cat ate the mouse" and "The mouse ate the cat" only differ by the word order and carry very different meaning).

6.4.2 tf-idf

tf-idf (*"term frequency - inverse document frequency"*) representations differ from the BOW, as the presence of a word in a document is associated with its number of occurrences in the document (tf) divided by the number of documents including this word (idf). The rationale is that, the more a word appears in a document, the more important it is in that document; in the meanwhile, the more documents it appears in, the higher the chance, it is trivial or uninformative. We use a slightly edited tf version, where the number of occurrences is taken logarithmically:

$$tf(i,w) = \log\left(\frac{\text{number of occurrences of } w \text{ in } i}{\text{number of words in } i}\right)$$

where i is a document and w a word. Likewise, the idf version used in the following is:

$$idf(w) = \log\frac{1+n_d}{1+df(w)} + 1$$

where n_d is the number of documents and $df(w)$ is the number of documents in which word w appears. Finally, the tf-idf is obtained as :

$$\text{tf-idf}(i,w) = tf(i,w) \cdot idf(w)$$

In particular, rare words are assumed to be distinctive and carry a high weight in the idf term.

6.4.3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) (a.k.a. *Latent Semantic Indexing*) was proposed by [Deerwester et al., 1990] for information retrieval to take advantage of the *semantic structure* of terms, *i.e.* term co-occurrences in the natural language organization. It is based on singular value decomposition (SVD), stating that any matrix of size (m, n) may be decomposed as the product of two orthogonal matrices U and V (respectively of dimensions (m, p) and (n, p)) and a Σ matrix, all 0 except for positive values on its first diagonal, referred to as *singular values*. The SVD decomposition of the tf-idf matrix is defined as:

$$\text{tf-idf} = U\Sigma V^T$$

The best k-rank approximation of tf-idf in the sense of the mean squared error over all coordinate, (MSE, or here, Frobenius norm), denoted $\overline{\text{tf-idf}}$, is obtained by keeping only the first k columns of U, V and Σ, denoted U_k, V_k and Σ_k, respectively:

$$\overline{\text{tf-idf}} = U_k\Sigma_k V_k^T$$

U_k (V_k) is called the k-dimensional LSA representation of the documents (resp. words). LSA is a widely popular approach in NLP, as it achieves dimensionality reduction, with k a few hundreds as opposed to the initial D dimension in the hundred thousands. Additionally, LSA offers some robustness w.r.t. synonymy [Deerwester et al., 1990]: words with similar sense tend to appear in similar contexts and are captured in the same SVD dimensions. On the other hand, LSA representations are dense, whereas BOW and tf-idf define sparse matrices.

Most interestingly, LSA can support cold start recommendation, as brand new documents tf-idf representation, denoted tf-idf_{new} can be mapped onto the LSA representation trained from the previous documents, denoted U_{new}, without requiring to train the model again:

$$U_{new} = \text{tf-idf}_{new} V_k \Sigma_k^{-1}$$

6.4.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation [Blei et al., 2003] is a Bayesian probabilistic model. The main assumption is that a document d involves a mixture of topics θ_d. The overall number of topics is a hyperparameter of the method. The number of topics in a document follows a Dirichlet distribution of parameter η. The k-th topic is defined as a distribution on the dictionary. noted β_k the distribution of words thereof. LDA defines a generative model:

1. The number n_w of words is sampled, with $n_w \sim \text{Poisson}(\xi)$;
2. The number of topics is drawn from $\text{Dirichlet}(\eta)$;
3. The probability of each selected topic in the document is sampled;
4. For each document word, i) a topic k is sampled proportionally to its probability; ii) a word is sampled after the corresponding topic distribution $\text{Multinomial}(\beta_k)$.

Parameters β_k and θ_d are estimated using a variational EM algorithm [Blei et al., 2003] to maximize (an approximation of) the posterior distribution. In particular, $\boldsymbol{\beta}$ defines a representation of documents in the topics base.

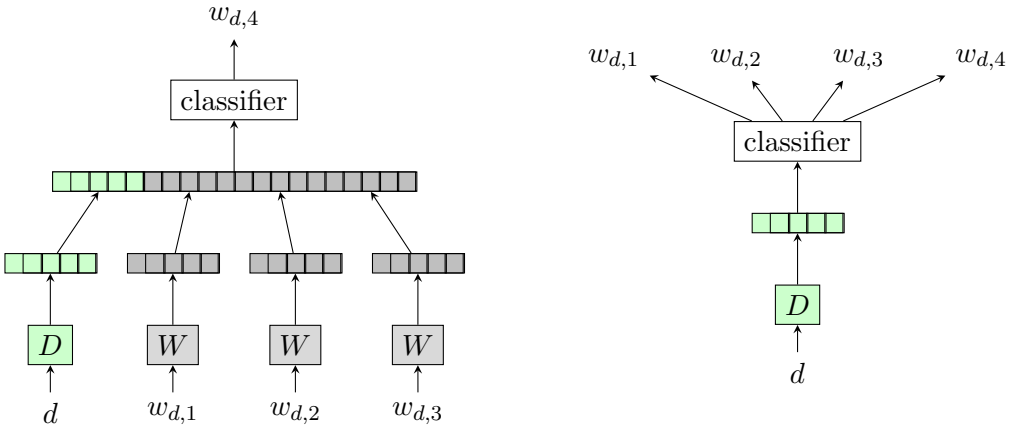

Figure 6.4: DM (left) and DBOW (right) neural model architectures in *Paragraph Vector*.

6.4.5 Paragraph Vector

Paragraph Vector, proposed by [Le and Mikolov, 2014], extends previous work on word embedding [Mikolov et al., 2013b] to a continuous (fixed-length) representation of both words and variable-length documents. Each document of the corpus (resp., each word) is mapped to a unique vector in a lookup table D (resp., a lookup table W).

Parameters D and W are learned as the projection parameters of a neural network. Two architectures are considered in [Le and Mikolov, 2014]:

- the *distributed memory* (DM) model architecture (Fig. 6.4, left) is inspired by Word2Vec *CBOW* model. The weights are optimized to learn a word from its *context*, that is, the representation of the T preceding words and the representation of the document, where T is a hyper-parameter. Formally, given words $\{w_{d,1}, \ldots, w_{d,T}\}$, the word representation look-up table W and the document representation, D_d the network is trained to predict the next word $w_{d,T+1}$.

- the *distributed bag-of-words* (DBOW) model architecture (Fig. 6.4, right) is inspired by Word2Vec *skip-gram* model [Mikolov et al., 2013b]. The weights are optimized to predict all words of document d in a small text window given the sole document representation D_d. Specifically, during training, text windows are sampled from the document and a randomly sampled word within the text window is to be predicted. Note that this model is order-independent.

The classifier layer is a hierarchical softmax [Morin and Bengio, 2005]; its weights are learned along with W and D matrices.

In [Le and Mikolov, 2014], the document representation combines the DM and DBOW models. Contrasting with Word2Vec, where new words cannot be taken into account after training, *Paragraph Vector* is said to be able of computing the representation of a new document with little effort. Conceptually, a new document is added to D while W and the softmax weights are kept fixed (the rest of matrix D does not intervene). The new document representation is obtained by gradient descent, using the same DM or DBOW model loss function.

6.5 Representations for OCCF

As discussed in Sec. 6.3, our target setting is a OCCF recommendation problem, as only positive feedback is available. This section, however, reviews a broader class of RS approaches, specifically examining how users are modeled to form personalized recommendations. User modeling is indeed a key issue for JAM problems (we shall come back to this point in Chap. 7). An original classification focusing on user modeling is devised.

Specifically, two categories of RS are distinguished. In the former category, called *single region of interest* (SROI) RS, a user is associated with a single, compact representation capturing all of their interests or tastes. Such representations lie in a single low-dimensional latent space where users interests and items distinctive features are abstracted. The RS is trained by bringing users closer to their selected items in the latent space; a user is recommended the nearest items in the latent space.

In the latter category, dubbed *multiple regions of interest* (MROI) RS, the user is associated with a set of representations, each capturing one or several particular aspects of the user specifics. While a SROI rather recommends items that match all of the user interests, MROI recommends items that best match (at least) one user's representation.

Prominent RSs are reviewed along the line of this classification, before discussing how the SROI/ MROI classification is related to the classical *neighborhood-based/model-based* classification.

6.5.1 Single region of interest user models

Matrix factorization

Matrix factorization (MF) approaches [Weimer et al., 2008; Koren et al., 2009; Shi et al., 2012] are based on the decomposition (or factorization) of $\mathcal{M}$ into two matrices U and I respectively explaining the contribution of each user (resp. each item) to the interactions. Equivalently, the goal is to find the best representations U (resp. I) of the users (resp. the items) in the same latent space, such that $U \cdot I$ approximates $\mathcal{M}$ well (in other words, explains the user's interest in each selected item). Among the decomposition methods is the *Singular Value Decomposition* (SVD), already introduced in Sec. 6.4.3. A difference however is that most elements of $\mathcal{M}$ are unknown. The optimization problem defining U and I thus is redefined as follows. Let the set of known interactions be $\mathcal{S} = \{(u,i)|\mathcal{M}_{u,i} \text{ is observed}\}$, the interaction matrix decompositions seeks U and I such that:

$$U, I = \underset{U,I}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{S}} \left(\mathcal{M}_{u,i} - U_u \cdot I_i^\top\right)^2$$

Another difference is that U and I must not only explain well the observed data but achieve a good generalization on unknown data. To this end, special care must be taken to prevent U and I from overfitting the known data. A first possibility is to restrict the rank of U and I. Another possibility is based on the L_2 norm regularization of both user and item representations [Koren et al., 2009]:

$$U, I = \underset{U,I}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{S}} \left(\mathcal{M}_{u,i} - U_u \cdot I_i^\top\right)^2 + \lambda(\|U\|_{1,2}^2 + \|I\|_{1,2}^2)$$

where $\|X\|_{1,2} = \sum_i \|X_{i,\cdot}\|_2$ and hyper-parameter λ controls the trade-off between the reconstruction error and the regularization term.[6]

[6]The overall model also involves some user bias terms b_u, item bias terms b_i and an overall bias term μ; these are omitted here as they do not change the analysis.

The factorization equation is further adapted to the OCCF setting, as follows. On the one hand, one should mostly reconstruct positive interactions ("ones"). On the other hand, some of the non-observed interactions ("zeroes") must also be reconstructed (otherwise a trivial solution would be to set U and I to the All-One vector). Hu et al. [2008] introduce a small positive weight $c_{u,i} = 1 + \alpha \mathcal{M}_{u,i}$ on the non-observed interactions[7], and required the matrix factorization to minimize a weighted loss, where the reconstruction of the zeroes in $\mathcal{M}$ has a weight much smaller than the reconstruction of the ones (as α is typically close to 100). The factorization problem then reads:

$$U, I = \underset{U,I}{\operatorname{argmin}} \sum_{(u,i)} c_{u,i} \left(\mathcal{M}_{u,i} - U_u \cdot I_i^\top \right)^2 + \lambda (\|U\|_{1,2}^2 + \|I\|_{1,2}^2)$$

In other approaches (CoFiRank [Weimer et al., 2008], CLiMF [Shi et al., 2012]) the reconstruction loss is replaced with a ranking loss (resp. the NDCG and the MRR measures, see Sec. 6.3.4).

In all previously mentioned approaches, the (warm-start) recommendations for user u are based on the reconstruction of $\mathcal{M}$:

$$\widehat{\mathcal{M}}_{u,i} = U_u \cdot I_i^\top \tag{6.1}$$

Items are then recommended to user u by decreasing value of $\left[\widehat{\mathcal{M}}_{u,i}\right]_i$.

A Bayesian approach, BPR-MF, is proposed by Rendle et al. [2009], achieving matrix factorization in an OCCF setting based on a triplet ranking loss. Formally, U and I are trained to maximize:

$$P\left(\langle U_u, I_i \rangle > \langle U_u, I_j \rangle \mid \mathcal{M}_{u,i} > \mathcal{M}_{u,j}\right)$$

User u preference for item i over item j is modeled as $\sigma\left(\widehat{\mathcal{M}}_{u,i} - \widehat{\mathcal{M}}_{u,j}\right)$ where σ denotes the sigmoid function. Dos Santos et al. [2017] extend BPR-MF and learn latent representations of users and items as Gaussian distributions rather than deterministic vectors. The effect of variance helps in modeling uncertainty in the data (*e.g.* a user with few known interactions) as well as inconsistency (*e.g.* in the user behavior). Key is that the reconstruction of the u-th row in $\mathcal{M}$ is obtained as:

$$\widehat{\mathcal{M}}_{u,i} = P(\mathbf{U}_u \cdot \mathbf{I}_i^\top + b_i > 0)$$

where $\mathbf{U}_u$ and $\mathbf{I_i}$ – and thus $(\mathbf{U}_u \cdot \mathbf{I}_i^\top + b_i)$ – are Gaussian variables. In particular, considering user u and item i, if the affinity $\widehat{\mathcal{M}}_{u,i}$ has high mean prediction but large variance, this may make item i less suitable for recommendation than another item j for which $\widehat{\mathcal{M}}_{u,j}$ has slightly lower mean prediction and low variance.

Collaborative topic modeling

An extension of LDA (Sec. 6.4.4) is proposed by [Wang and Blei, 2011], defining a method called collaborative topic regression (CTR). The motivating application concerns the recommendation of scientific articles in an online scientific community framework; the collaborative filtering matrix indicates whether a user's library contains a given article. CTR

[7] In the original work of Hu et al. [2008], $\mathcal{M}$ counts number of occurrences item i has been "consumed" by user u; thus there are a whole range of values possible for $c_{u,i}$. Instead, in JAM problem, items are consumed at most once; hence, $c_{u,i}$ takes only two different values, 1 for negative and $1 + \alpha$ for positive interactions.

combines ideas from collaborative filtering based on latent factor models, and content analysis based on probabilistic topic modeling and specifically Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. Each LDA topic (e.g., "artificial intelligence") is characterized as a distribution on the vocabulary space. The i-th article is associated a latent description Θ_i, with $\Theta_i[k]$ the percentage of the k-th topic in the article (*e.g.* , a given article might be described as 70% "artificial intelligence", 20% "high performance computing" and 10% "application"). Eventually, latent descriptions U_u of the user and $\Theta_i + \boldsymbol{\epsilon_i}$ of the article are optimized[8] (note that $\boldsymbol{\epsilon}$ is a matrix), such that $\langle U_u, \Theta_i + \boldsymbol{\epsilon_i} \rangle$ best fits the collaborative matrix $\mathcal{M}_{u,i}$,

$$(U^*, \boldsymbol{\epsilon}^*) \in \underset{U,\boldsymbol{\epsilon}}{\operatorname{argmin}} \left\{ \sum_{u,i} c_{u,i} (\mathcal{M}_{u,i} - \langle U_u, \Theta_i + \boldsymbol{\epsilon}_i \rangle)^2 + \lambda_u ||U||_2^2 + \lambda_v ||\boldsymbol{\epsilon}||_2^2 \right\}$$

with λ_u and λ_v respectively the weights of the regularization terms preventing CTR from overfitting the data, $c_{u,i} = 1$ if $\mathcal{M}_{u,i} = 1$ (the i-th document is in the u-th user's library), and $c_{u,i} = 0.01$ otherwise (note how similar to the confidence coefficient in the matrix factorization of [Hu et al., 2008] $c_{u,i}$ is).

With this approach, user interests are captured by a single "topic" vector U_u, which is trained to maximally span the topics of all documents that user u bookmarked, while being orthogonal to topics the user is supposedly not interested in (i.e., $\mathcal{M}_{u,i} = 0$); as above, the recovery of positive interactions has a much higher weight than that of the negative interactions.

To summarize, CTR achieves matrix factorization and content modeling using a single, joint learning process. Gopalan et al. [2014] use a similar approach as CTR but with different generative processes to model documents and ratings.

Restricted Boltzmann Machines based recommendation

[Salakhutdinov et al., 2007] propose RBM-CF, a two-layer network based on Restricted Boltzmann Machines (RBMs) to model the observed values of $\mathcal{M}$. Specifically, each user u is modeled by a RBM; its visible units are the observed values of row $\mathcal{M}_u$ and its (binary) hidden units represent latent user features.

The original formulation of [Salakhutdinov et al., 2007] is designed for K-level ratings collaborative matrices: for each observed rating, K visible units are connected to the hidden units, and the RBM is used to explain the observed ratings as a classification task. The implicit data (*i.e.* the fact that a user u rated an item i) is integrated as another Boolean visible unit per user-item pair. Recommendation is based on the reconstruction of missing ratings: for each unobserved user-item pair, the estimated rating is taken as the expectation across all possible ratings (probabilities thereof are computed by the user-specific RBM).

Key to this approach is that the weights between the visible and hidden units are shared across all RBMs: for any two users who rated the same item i, the weights from the K visible units to the hidden units of each user are the same. Of course, if the ratings differ this item would affect the state of the hidden units differently, and as a consequence, the hidden unit states are specific to each user. Note that this weight sharing is necessary to learn the weights in a collaborative way; otherwise, only the item biases would connect the different user RBMs .

[8] $\boldsymbol{\epsilon_i}$ is added, allowing the article latent description to diverge from its (fixed) topic proportion Θ_i, to better fit the collaborative information.

In this approach, users are modeled as the states of the hidden units $\boldsymbol{h}$. Rating predictions are obtained as linear combinations of the hidden units states with the RBMs weights W^k. We may think of the network weights (connecting the hidden units to each item) as item representations (as mentioned, weights are the same for all RBMs). The non-normalized probability that the user gives rating k to an item i is obtained as $W_i^k \cdot \boldsymbol{h}$, resembling matrix factorization (Eq. 6.1).[9]

RBM-CF has been substantially extended by [Georgiev and Nakov, 2013] in two directions. Firstly, an ordinal cost function is considered (if the observed rating is "4", the system should be more penalized predicting a "1" than a "3"[10]) by considering one real-valued rating instead of a multinomial variable. Secondly, the same user-specific RBM approach is adapted as item-oriented RBMs to capture the user-user similarity as well; both user- and item- oriented RBMs are combined in a single hybrid RBM model.

Neural Nets

A number of approaches have recently been proposed to predict missing interaction data from known user ratings. AutoRec [Sedhain et al., 2015] is trained as an autoencoder neural network [Hinton and Salakhutdinov, 2006] (the low-dimensional latent hidden layer is used *in lieu of* the hidden units in RBM-CF) that predicts missing ratings from all observed data. Collaborative Denoising Auto-Encoders [Wu et al., 2016] use similar ideas but is suitable for the OCCF setting: a corrupted version of the user set of interacted items is fed to the network to reconstruct the (true) set of interactions. CF-NADE [Zheng et al., 2016] models each rating from all previously observed ratings of the same user.[11] These approaches can be straightforwardly extended to deep neural architectures.

User embedding

Borrowing the DM framework of Paragraph Vector (Sec. 6.4.5), User2vec [Grbovic et al., 2015] learns a user embedding where the user is a considered as a global context (like a document) while the items she interacted with are seen as words. [Sun et al., 2017] proposes a Bayesian approach to model all users and items in a same representation space, taking into consideration the user-item interaction and whether two items have a connection to a same user. Furthermore, they extend their framework by considering two levels of granularity: the item category (centroid of all items in that category) and the individual items that deviate from it. Other feed-forward neural architectures, such as DSSM [Huang et al., 2013], MV-DSSM [Elkahky et al., 2015] or NCF [He et al., 2017] have been proposed for the similar purpose of learning a shared representation space for both user and items given content data.

Matchbox [Stern et al., 2009] is a probabilistic approach that can accommodate user descriptions $\mathbf{x}_u$, item descriptions $\mathbf{y}_i$ as well as a description of the "context" of the interaction Φ. The match between the u-th user and the i-th item is modeled as a Gaussian variable:

$$\mathcal{N}(\langle U\mathbf{x}_u, V\mathbf{y}_i\rangle + b_{u,i}, \beta),$$

[9] As for the MF description, item biases have been omitted. Note that there is one bias coefficient for each one the K rating units.

[10] This is similar to the so-called concept of "cost-sensitive classification" introduced in Sec. 3.2.

[11] As training is affected by the order in which ratings are provided, the method would ideally keep the true time-ordered sequence of rating. However [Zheng et al., 2016] note that a randomly drawn ordering performs well.

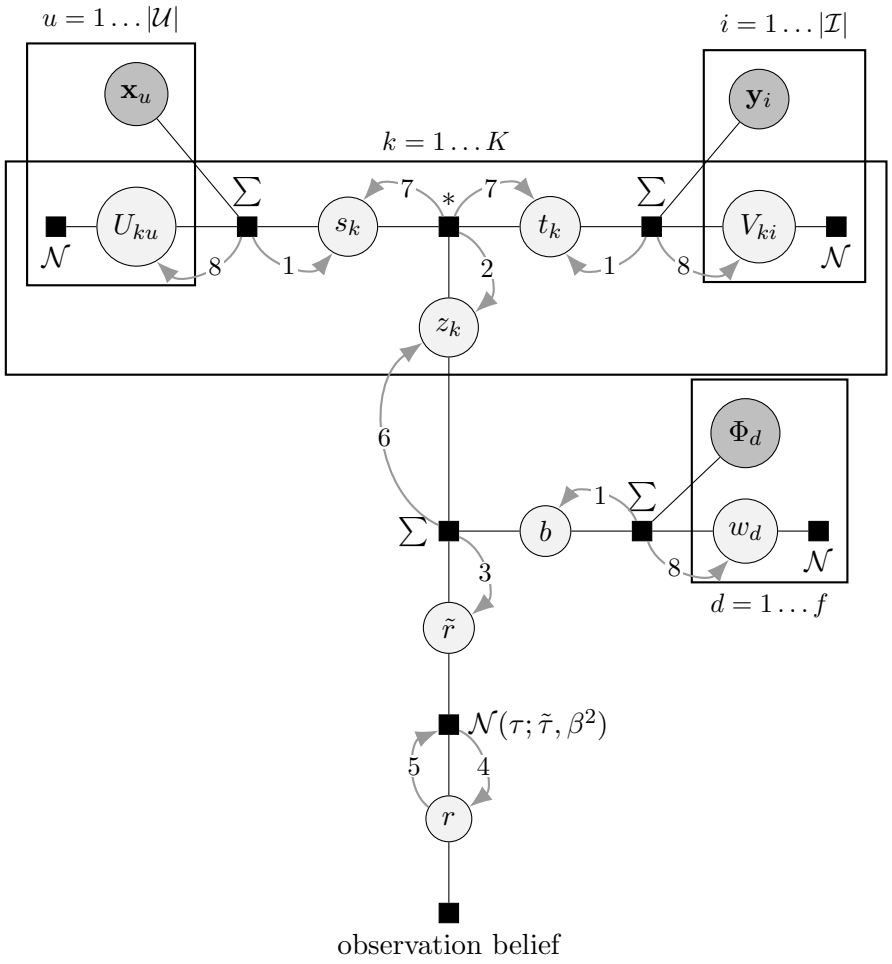

Figure 6.5: Graph of the Matchbox bi-linear rating model [Stern et al., 2009].

where $b_{u,i}$ is a bias term that may incorporate Φ, and standard deviation β. Matrices U and V (of size resp. $|\mathcal{U}| \times K$ and $|\mathcal{I}| \times K$) and bias b are learned by the model. In particular, if no side information is available, $\mathbf{x}_u$ and $\mathbf{y}_i$ can be set to $\mathbf{e}_u$ and $\mathbf{e}_i$, respectively, and the expected rating reduces to that of standard MF (Eq. 6.1) with biases.

Several components are responsible for the flexibility of Matchbox. One of them is to consider a decoding mechanism from $(\langle U\mathbf{x}_u, V\mathbf{y}_i \rangle + b_{u,i})$ to the observed $\mathcal{M}_{u,i}$. This decoding mechanism is used to account for ranking or binary feedback, when $\mathcal{M}_{u,i}$ respectively is a rank or a binary variable. The Matchbox bi-linear rating model is depicted on Fig. 6.5.

Matchbox relies on incremental learning, with two motivations. The first one is to accommodate preference drift of the user, whose tastes generally evolve over time; any new information is used to update the model. To this end, Gaussian noise is added at each timestep. The second one is to allow for fast learning if needed, by achieving a single pass through the data. Specifically, Matchbox uses an approximate message passing algorithm: from the initial user/item descriptions to the latent descriptions and biases; from the latent descriptions to their product; from their product to the observed rating (the decoding mechanism); and a Gaussian noise factor is last accounted for to model the user preference drift. Some care must be exercised to enforce a decrease of variance, using the minimization of the KL divergence in lieu of Expectation Propagation.

6.5.2 Multiple regions of interest user models

Neighborhood based approaches

Early RS approaches are based on the user-oriented collaborative filtering approach (Sec. 6.3). [Sarwar et al., 2000] identifies 3 steps in user-oriented CF: representation, neighborhood formation and generation of recommendation. While user modeling is usually based on their interactions, it can also be based on side information (content-based recommendation), or leverage both interaction and content information. The key point is that recommendations made to user u rely on the interaction history of users in her neighborhood $\mathcal{V}(u)$, usually made of her κ nearest neighbors. In other words, a given user is modeled through the set of her neighbouring users. Several methods have been proposed to define a user-user similarity (*e.g.* cosine similarity, Pearson correlation [Sarwar et al., 2000]) and to build user neighborhoods (*e.g.* vanilla nearest neighbors or iterative construction). The recommendations are then built by aggregating the positive interactions of the neighbors, for instance through a similarity-weighted sum:

$$\mathcal{R}_k(u) = \text{top@}k \left\{ \sum_{v \in \mathcal{V}(u)} \text{sim}(u,v)\mathcal{M}_{v,i} \right\}_i \tag{6.2}$$

Symmetrically, item-oriented CF [Deshpande and Karypis, 2004] sees a user as a basket of the items she interacted with. Items similar to those are then recommended through: i) defining an item neighborhood based on the item-item similarity; and ii) aggregating the nearest items (see, *e.g.* Item2Vec [Barkan and Koenigstein, 2016] and the "item-item SVD based" method therein).

The differences between the two approaches can be described in terms of exploration, or *serendipity*, *i.e.* the ability to recommend/discover unexpected relevant options. In user-oriented CF, the exploration is achieved through the community of users: the user is recommended the items liked by many of her neighbors. In item-oriented CF, the exploration actually depends on the diversity of the items selected by the user: only items similar to those will be recommended. In the former case, users with many centers of interest can cause over-exploration. In the latter case, exploration might be limited to the "more of the same" strategy. Both approaches have been merged into a single framework in [Wang et al., 2006; Verstrepen and Goethals, 2014]. For a detailed survey, the interested reader is referred to [Desrosiers and Karypis, 2011].

Note that BPR [Rendle et al., 2009] and matrix factorization can also be used within neighborhood-based approaches.

Graph-based recommendation

In the OCCF framework, $\mathcal{M}$ can be interpreted as a bipartite weighted graph with user nodes and item nodes, where edges represent the user-item connections [Huang et al., 2004]. Besides OCCF, the approach of [Aggarwal et al., 1999] consists in walking on the user-user graph to predict ratings *not only* from the neighbors of the current user. A number of methods (*e.g.* , [Fouss et al., 2007; Gori and Pucci, 2007]) use random walk on the item-item graph to provide recommendations ("spreading" from a user's connected items); the key to these approaches are the weights, or alternatively how to attenuate the spreading. All have in common that a user is seen as the set of vertices that originate from her user node or equivalently as the set of her items she interacted with, allowing propagation of her interests in several parts of the graph at the same time.

6.5.3 Discussion

The difference between the SROI and MROI settings and its impact can be illustrated as follows. Let us consider a jobseeker looking for a position either in Paris or in London, and thus applying to jobs in both cities. Such a user can be viewed as a 2-profile person. In a SROI system, the user is attached a single profile. This profile would either discard the geolocation feature[12] or would retain an over-generalization thereof (e.g., recommend jobs in the entire area centered around Lille). As the size of the database increases, and hence the granularity of the latent SROI space also increases, the latent space will eventually capture the concept of "these people that look for a position in both Paris and London" aggregating job ads located in any of these cities. At this point, suggesting jobs in both cities (and not only one of them) may become easier. As opposed to SROI, MROI systems simply handle such cases by attaching several profiles of the same job seeker.

The MROI setting, termed "recommendation for shared account" in [Verstrepen and Goethals, 2015], occurs naturally in applications where a single account is shared by several users, *e.g.* a family, and there exists no explicit means to distinguish among these users at shopping time. The MROI setting is analogous to that of multiple instance learning (MIL) [Dietterich et al., 1997; Andrews et al., 2003], an extension of supervised classification where each example involves a set of instances and one of these instances is responsible for the example belonging to the target class. Allegedly, the recommendation for shared account in the OCCF setting is more challenging than MIL, as the positive class is only partially observed.

Our claim is that the classical *memory-based vs model-based* distinction is less general and less informative than the SROI *vs* MROI distinction: the former distinction focuses on the information provided to the RS while the latter regards the internal structure of the RS.
Formally, model-based RSs a.k.a. *latent factor models* [Koren, 2008], seek a shared representation space for both users and items that explains the interactions. Memory based RSs a.k.a. *neighborhood approaches* focus on finding good similarity to explain the relations between pairs of users (alternatively, pairs of items) and thence construct neighborhoods; recommendations follow from the memorized interactions of users in the neighborhood (alternatively, from the items neighboring those the current user interacted with).

In most cases, a model-based RS falls within the SROI setting, and a memory-based RS falls within the MROI setting. However, this is not necessarily the case:

- A memory-based approach only recommending the items selected by the 1-nearest neighbor of the current user achieves a SROI recommendation as the current user is only seen through the perspective of her closest neighbor.

- Inversely, a model-based approach can achieve a MROI recommendation if p views are defined on the n-dimensional latent space, and each view supports a different RS − capturing different aspects of users.

Along this line, hybrid models combining recommendations from an enhanced latent factor model component and a neighborhood-based approach [Koren, 2008] can be categorized as MROI systems: the user is attached to a single model captured by the latent factor model, which is finely exploited based on local neighborhoods to adjust the predicted ratings.

[12] See Sec. 8.6 for the analysis and discussion of the importance of geolocation in the JAM problem, depending on the job sector.

6.6 Organization of the work

6.6.1 Scientific questions addressed

The goal is to produce high-quality recommendations for a OCCF problem: JAM. The challenge is to bridge the gap between the textual and the collaborative JAM data. In order to do so, the relationship among both representations must be understood in detail – this statement widely echoes the main lessons from Part I.

Specifically, the first scientific question tackled in the following chapters is whether the information contained in job ads and resumes is sufficient to determine whether an applicant is qualified for a job position – in other words, whether JAM can be tackled as a pure information retrieval or textual entailment task [Büttcher et al., 2010; Dagan et al., 2006].

At the other extreme of the spectrum, the relevance of a job ad to an applicant can only be determined from the (hidden) preferences of the applicant, as manifested by her past behavior – in other words, JAM is a pure collaborative filtering task [Koren and Bell, 2015]. The second scientific question thus is how to tackle the item cold-start JAM problem in such a case, and how to exploit and combine the item content information together with the CF matrix. Such a RS falls in the hybrid RS category (Sec. 6.3.2).

A third possibility is that the information within the job ads, the resumes and the CF matrix can be leveraged to build a bridge between the job ad and the resume spaces, similar in spirit to a machine translation task. The third scientific question is to investigate the necessary conditions for establishing such a bridge in terms of consistency. Such a bridge would eventually allow to tackle the full cold start JAM problem, and recommend brand new job ads to brand new applicants.

Along the first direction, the user is essentially modeled based on her resume, following a SROI approach. Along the second direction, the user is essentially defined from her applications on previous job ads, following a MROI approach. Along the third direction, distinct latent spaces are defined for users and items, and a mapping is defined between both latent spaces along a MROI approach. All three questions are investigated in Chap. 7 along with a detailed presentation of the data.

6.6.2 Using artificial neural networks to tackle the JAM problem

The complexity of the JAM problem makes it desirable to go beyond the classical methods considered in Sec. 6.5 in order to identify textual and behavioral patterns and leverage them for recommendation.

In this manuscript, two original approaches are proposed along this line, taking inspiration from machine translation (Chap. 8). The first approach, dubbed LAJAM, exploits the CF matrix together with the rich description of the job ads; it discards the actual description of the users (their resumes). A two-step approach is defined: i) the two descriptions of the users (respectively, the textual description of the job ads they selected, and the behavioral description of the users, *i.e.* this set of job ads) are aligned; ii) an indirect recommendation scheme is then used to form recommendation concerning brand-new job ads.

The second approach, named MATJAM, achieves the "translation" between resumes and jobs by embedding them into a same latent representation space, viewed as a pivot language; the recommendations follow from the similarities in this pivot language.

Note that both approaches take opposite directions to modeling users. While LAJAM considers each user as the non-aggregated set of the jobs she applied to (*i.e.* it is a MROI

system), MaTJam summarizes in a single location in the latent representation space (and thus falls into the SROI category). They are designed to have similar representation capabilities; they are thus compared along an item cold start setting on the same Jam problems. Note that by construction, MaTJam can also deal with the full cold start setting.

Chapter 7

Data analysis

Access to large datasets representative of a Jam problem is a major concern for scientific research in the field. As mentioned earlier, this lack of extensive data has been partially overcome with the introduction of large-scale datasets provided by the professional network XING[1] (pronounced cross-ing) in the context of the RecSys challenges 2016 and 2017. It must be emphasized, however, that these challenges focus on the recommendation tasks, completely setting aside the natural language aspects involved in Jam: resume and job documents are encoded using manually defined numerical IDs meant to represent *concepts*. Besides, both content and interaction data have been subject to noise addition to preserve the confidentiality of the network – as well as protect the data value for the company. Indeed, neither XING nor the challenge organizers should be blamed for these noise adding operations; such preprocessing is both unavoidable and desirable before making public individual data. Nevertheless, this preprocessing limits the possibility of learning what are the Jam problem peculiarities as well as understanding the failures of a RS tackling it.

In this chapter, two proprietary datasets, kindly provided by Web hiring agency Qapa[2] and non-profit organization ABG[3], are introduced. Insights on the Jam problem are then provided by looking at the datasets through the prism of simple RS approaches.

7.1 Introduction to the data

This section describes both anonymized large-scale and proprietary databases considered in this study. These datasets are compared to the well studied and publicly available CiteULike benchmark[4], which will also serve to support the comparative empirical validation (Section 8.6).

7.1.1 Databases

The Qapa platform is targeted at low-paid temporary jobs and unskilled applicants. An applicant is not required to provide a resume as they often do not have any; she might access the Qapa platform using her mobile phone and writing a few sentences. We had access to the 5 million job ads and 4 million resumes recorded in the Qapa database over the 2012-2015 period. The study considers the excerpt recorded over 2015 May-July as the

[1] `https://www.xing.com/`
[2] `https://www.qapa.fr/`
[3] `http://www.intelliagence.fr/`
[4] `http://www.citeulike.org/faq/data.adp`

platform was significantly extended in April 2015, offering 2,000 skill additional features from the official French job ontology[5] to applicants and recruiters to enrich their job ads or resumes. The ABG platform is targeted at PhD applicants and job ads for PhDs in industry; the database contains 10,000 job ads, 14,000 resumes and 68,000 clicks, recorded over the 2010-2015 period. The study considers an excerpt of the ABG dataset, retaining applicants who clicked on at least two job ads, and job ads that have been clicked upon by at least two applicants. To a job ad (resp. a resume) are usually attached a date and a city converted in geolocation (respectively, a date; the applicant geolocation is not available due to privacy constraints).

Besides the job ad and resume corpora, each database involves the collaborative filtering matrix $\mathcal{M}$, with $\mathcal{M}_{i,j} = 1$ iff the i-th user applied on the j-th job ad (throughout this work, "click" and "application" are used interchangeably; clicking on a job ad to read it is not included in the interaction data). Job ads and resumes are written in French, except for the ABG database where a large fraction (circa 30%) of the job ads are written in English or contain a job description written both in French and in English.

The CiteUlike database considered[6] includes 16,980 articles including 67 words on average (only titles and abstracts being considered) using a tf-idf description on a 8,000 word vocabulary. The collaborative filtering matrix $\mathcal{M}$ describes the virtual library of the 5,551 platform users, with $\mathcal{M}_{i,j} = 1$ iff the j-th article belongs to the i-th user's library. $\mathcal{M}$ is less sparse than for Qapa and ABG by an order of magnitude. All datasets are summarized in Table 7.1.

	Qapa	**ABG**	**CiteULike**
# items (# wd)	56 (60)	10.4 (113)	17 (67)
# users (# wd)	31 (46)	8.4 (NA[7])	5.6 (NA)
# clicks (spars.)	226 ($13 \cdot 10^{-5}$)	63 ($71 \cdot 10^{-5}$)	205 ($22 \cdot 10^{-4}$)

Table 7.1: The Qapa, ABG and CiteULike excerpt databases: number of items (job ads or articles) in thousands with average number of words per item in parenthesis, number of users (applicants) in thousands with average number of words in resumes for Qapa and ABG in parenthesis, number of clicks in thousands with sparsity of the collaborative filtering matrix in parenthesis. The vocabulary consists of the most frequent 10,000 (resp. 8,000) words in Qapa and ABG (resp. CiteUlike).

7.1.2 Preprocessing

Qapa-specific preprocessing

Applicants are described using a bag of words, concatenating a list of desired position titles, their resume, and a list of skills. Job ads are similarly described by the concatenation of a title, the job description, a list of desired skills, augmented with side information such as the ad posting time and the location of the position (see below).

[5]These skills are part of the ROME ontology `http://www.pole-emploi.fr/candidat/le-code-rome-et-les-fiches-metiers-@/article.jspz?id=60702`.

[6]Available at: `http://www.cs.cmu.edu/~chongw/data/citeulike/`.

[7]The ABG database includes resumes for 6,600 users, and interaction data for 8,400 users. Unfortunately, the number of users for whom both the resume and the interaction data are known is only 2,700. For the sake of the problem relevance, resumes are not considered when handling the ABG recommendation problem.

Skills are represented via Boolean features (extra words) and concatenated with the job ad or resume description. Another option would have been to concatenate the textual description of every skill involved in a job ad or resume to the job ad or resume description; this option would, however, lose the skill "concept", identified through a single synthetic word.

The side description (not used for further preprocessing steps unless mentioned otherwise) attached to each job ad is defined as follows:

- jobs time information: the posting time is transformed into a difference in seconds to a reference time;
- jobs geolocation: city names are extracted from about 70% of the jobs titles; they are transformed into longitude/latitude coordinates; there again, these may be considered as a difference in coordinates w.r.t. the reference point (0,0). When missing, the median value along each coordinate is imputed.

The proposed recommender systems usually take the side information into account through considering the differences of posting time or geolocations, thereby filtering out the effects of the chosen reference point (using *e.g.* biases within neural network architectures). For this reason, posting time and geolocation do not need to be rescaled (e.g. mapped onto the $[0, 1]$ interval). Keeping the original time and geolocation coordinates makes it easier to compare their impact in the different datasets.

ABG-specific preprocessing

Likewise, applicants are described with a bag of words, obtained by concatenating: profile title, highest degree, thesis title, scientific and technical fields, skills, professional objectives...

Job ads are also described as a bag of words concatenating the type of employment (*e.g.* "PhD candidate position" or "permanent position"), name of the employer, scientific domains (*e.g.* "computer science"), requirements (skills, degree), job category (mostly "research and development" and "higher education and research") title and description of the position. Additionally, the city and posting time are provided as separate attributes and handled as for Qapa.

Common preprocessing pipeline

Text of jobs and resumes are first passed through a series of manually crafted rules to remove, *e.g.* punctuation and HTML tags, and to transform all letters to lower-case. The resulting text is then stemmatized using the Python `NLTK` library [Bird et al., 2009] to reduce the diverse inflected forms of a word to an artificial root word (different from its lemma, which is the form of the word that would appear in a dictionary), *e.g.* "utiliser" (to use) and "utilisation" (usage) are both reduced to "utilis". This step helps significantly reducing the size of the vocabulary by about 25% while globally preserving the meaning of all words. Additionally, a list of all language-specific "stop words" is used to dismiss all words that carry little sense (such as "and","above", "during"). The list used comprises 465 French stop words including all 26 single-character words (from "a" to "z"). The resulting texts of resumes and jobs is encoded using a single bag of word (BOW) model to align the dictionary among the two types of documents. The resulting description is called the **BOW representation**.

For Qapa only, skills are considered and added to the natural language dictionary as extra-words (one word per skill). The same skill dictionary is used for both resumes and jobs to preserve the BOW representation consistency.

A tf-idf representation

At this point, resumes and jobs have aligned representations and are merged into two sets of documents (the resume and the job corpora, respectively). This representation will support the next two steps: vocabulary selection and weighting. For each resume or job, the number of occurrences of the word in the resume or job document is replaced using the so-called tf-idf operator (see Section 6.4.2). The tf-idf recoding is followed by a vocabulary selection step, retaining only the top 10,000 words sorted by decreasing idf. The resulting document $\times$ word matrix is referred to as **tf-idf representation** of the data in the following.

An LSA representation

On top of this tf-idf representation, another representation referred to as **LSA representation** in the following, is constructed. Again, all documents are used to compute the singular vectors, resulting in an aligned representation of resumes and jobs.

Unfolding the skills

The analysis of the corpus from an NLP perspective is conducted by replacing the extra-word associated with each skill by its textual description, concatenated to each document (resume or job) including this skill. This representation (without the tf-idf weighting) is referred to as **BBOW-NL** representation (Boolean BOW-NL).

7.2 Exploratory analysis from an NLP perspective

7.2.1 Alignment of NL representations

This section aims at investigating the natural language aspects of the considered application; it only considers the BBOW-NL representation. Four corpora are considered: resumes in Qapa and ABG, jobs in Qapaand ABG. The overall set of documents involves a 129,211-word vocabulary. Tab. 7.2 details for each corpus the size of the dictionary. In particular, note that the Qapa resumes size is far greater than that of Tab. 7.1. This fact is due to the occurrence of many rare words (presumably misspelled words).

	Qapa jobs	**Qapa resumes**	**ABG jobs**	**ABG resumes**
# documents	56,512	30,669	10,473	6,600
# boolean word occurrences	4,496,997	3,603,362	1,240,791	384,013
dictionary size	44,224	68,436	44,937	27,988
Avg. # different words per document	79.6	117.5	118.5	58.2

Table 7.2: Dictionary sizes for all 4 corpora.

Fig. 7.1 illustrates the word distribution per corpus, showing the ranked word frequencies on a log-log scale. Both datasets feature different characteristics. Firstly, ABG involves a large set of rare words. This fact is partially explained from the highly technical description of the resumes (*e.g.* integrating the PhD thesis titles). Another possible

explanation is that circa 30% of ABG ja s are written in English, or augmented with an English translation.

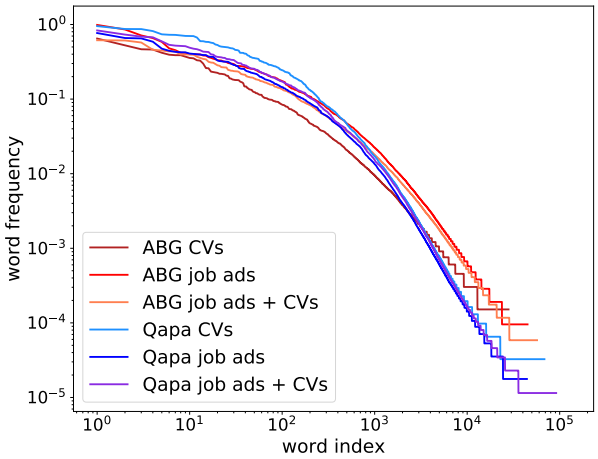

Figure 7.1: Word frequency in the BBOW-NL representations of all 4 corpora and globally in the ABG and Qapa datasets (log-log scale).

The differences between corpora are further investigated by comparing the vocabulary used in resumes and jobs, respectively. These word distributions are illustrated on Fig. 7.2. It is worth noting that most words belong to two areas (the red areas in Fig. 7.2). Specifically, an overwhelming fraction of the words is either used in jobs only (bottom right), or in resumes only (top left). The mismatch between the job and resume word distribution *a posteriori* explains why the use of a simple matching (*e.g.* using a cosine similarity) between job and resume documents proves to be ineffective for recommendation.

Most interestingly, the observation that people asking an information and people querying an information tend to use different vocabularies has been made before: see [Ruotsalo et al., 2015] and references therein for a discussion of this phenomenon.

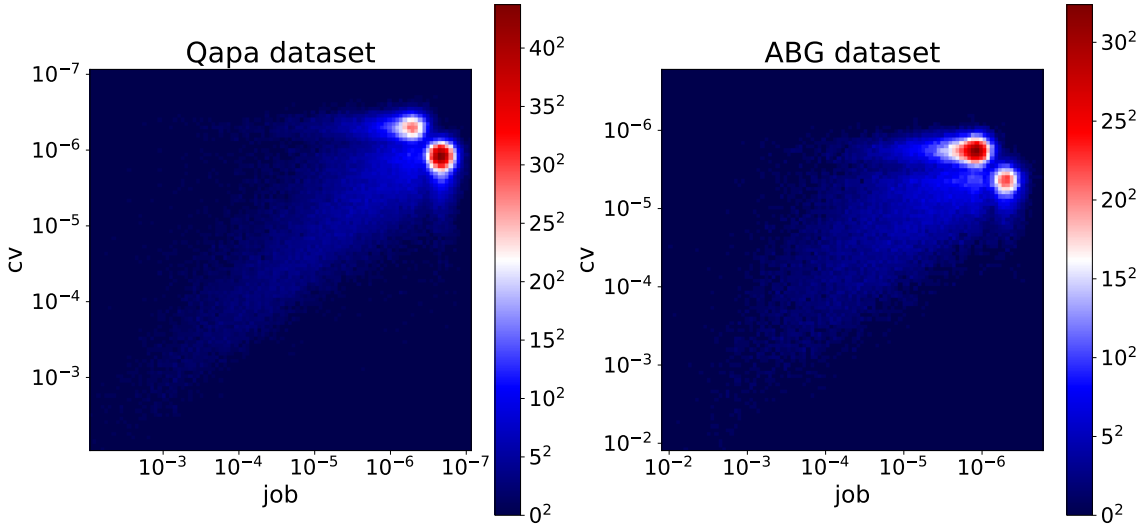

Figure 7.2: Word distributions in Qapa (left) and ABG (right), on a negative log scale. Word w with frequency p_x (resp. p_y) in the resume (resp. job) corpus is located at $(-\log(p_x), -\log(p_y))$. Color indicates density, *i.e.* how many words are located at the same coordinates.

The – visually observed – distribution differences between the corpora are quantified using the Kullback-Leibler (KL) divergence. Considering P and Q two distributions defined on the (finite) vocabulary W, the KL divergence between P and Q is defined as:

$$KL(P||Q) = \sum_{w \in W} P(w) \log \frac{P(w)}{Q(w)}$$

To avoid numerical issues, the considered vocabulary is set to the set of words appearing at least once in each corpus.[8]

The stability of the KL divergence is further characterized by considering the variance of the distribution in each corpora, estimated from $KL(P||P')$ with P and P' the empirical distributions estimated from a random bi-partition of each corpus. Specifically, each corpus is divided in two, with P and P' the empirical distributions on each subset and $KL(P||P')$ (averaged over 10 random split of the corpus) is used to estimate the variance of the corpus distribution (Tab. 7.3).

	CV^1	CV^2	Job^1	Job^2
CV^1		0.005	0.323	0.323
CV^2	0.005		0.324	0.324
Job^1	0.470	0.472		0.007
Job^2	0.471	0.473	0.007	

Table 7.3: Quantitative estimates of the distribution differences between Qapa resumes (CV) and job ads, measured from the KL divergence KL(row||column) and averaged over 10 runs. Standard deviations are under 0.001.

Tab. 7.3 yields the following conclusions. While resumes (resp. jobs) are homogeneous sets (low KL variance within the set) they significantly differ from each other. Furthermore, KL(job||CV) is much higher than KL(CV||job). A tentative interpretation for this fact goes as follows: KL($P||Q$) is affected by elements with low Q probabilities. In our case, resumes involve many rare words, much more than jobs, as shown in Fig. 7.1. The occurrences of these rare words (due to mispellings, rare acronyms, enterprise names and adress...) might thus explain these differences.

The difference between the two datasets is also visually and quantitatively estimated. As illustrated on Fig. 7.3, the vocabularies used in both datasets among the two users corpora (resp. among the two jobs corpora) are different. This finding was expected, as the Qapa dataset focuses on low-qualification jobs whilst the ABG dataset deals with resumes and jobs with high technical expertise. This difference (schematically, between blue-collar and white-collar job sectors) is reflected in the vocabulary used; additionally, the ABG dataset is bi-lingual (70% French and 30% English).

This difference strongly suggests that the two datasets are better considered independently.[9]

[8]Note that the KL divergence is not symmetric, hence it is not a distance. The Jensen-Shannon divergence, half-sum of $KL(P||Q)$ and $KL(Q||P)$ would be used, if we needed the distance properties.

[9]If the amount of data were overwhelming, there might indeed be some benefits in considering the continuum between white and blue-collar domains. This perspective is left for further work with larger amount of data.

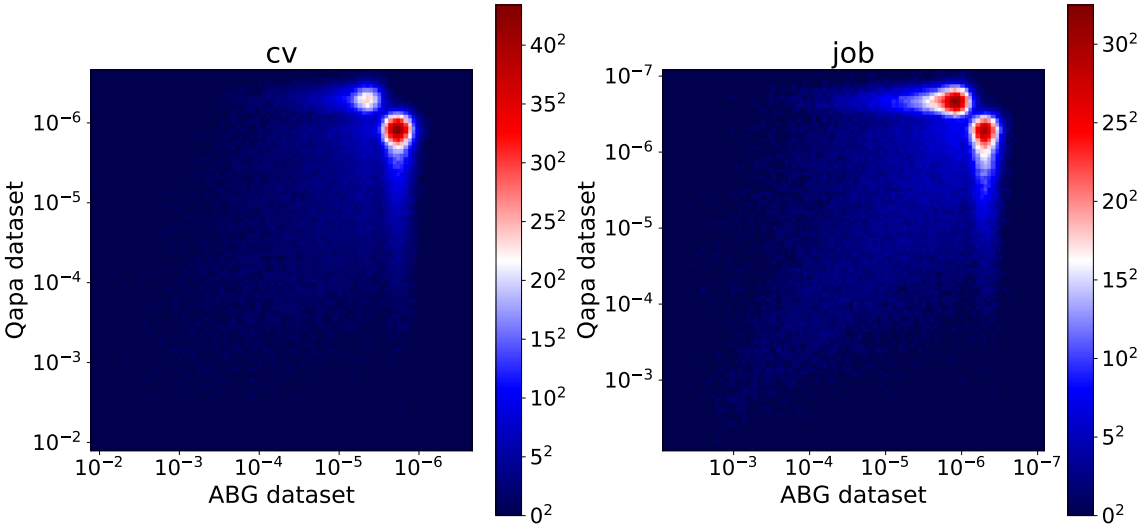

Figure 7.3: Word distributions in resumes (left) and jobs (right), on a negative log scale. Word w with frequency p_x (resp. p_y) in the Qapa (resp. ABG) dataset is located at $(-\log(p_x), -\log(p_y))$. Color indicates density, *i.e.* how many words are located at the same coordinates.

7.2.2 Recommendation based on pure content matching

As previously stated, the mismatch between the resume and job distributions complicates the direct matching of the users and items, using, *e.g.* cosine similarity. A more thorough content-based matching was investigated using the tf-idf and LSA representations to obtain baselines for the neural-network-based approaches presented in Chap. 8. In both cases, the similarity between documents is computed as the cosine similarity of the representation vector [Aizawa, 2003; Landauer and Dumais, 1997]:

$$s_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \cdot \|\mathbf{y}\|_2}$$

This *direct matching* approach considers JAM as a pure information retrieval problem: the user resume is viewed as a "query" tentatively matched with every document (job) by the recommender system.

Tab. 7.4 reports the performances of this pure content-based RS with the tf-idf representation (10k words) and the LSA representation (600 dimensions) on the Qapa dataset in terms of recall.

The experimental setting considers a 10-fold cross validation approach. Firstly, the itemset is equi-partitioned into 10 subsets, and the results are averaged on all 10 subsets. For each subset, for each user (resume), the items (job) are ranked by decreasing cosine similarity with the user and the rank of the relevant items in the fold (circa 5,600 jobs each) is recorded. The recommendation performance is measured from the recall at 20, 100 and 200, where the Recall@T is the fraction of relevant items ranked in the top T.

Note that this content-based RS does not take into account the information from the collaborative filtering matrix. In tf-idf representation, the word weights only reflect their frequency among the documents. In LSA representation, the singular vectors also reflect the structure of the corpus (jobs and resumes) in an unsupervised way. The RS thus is blind to how well the words or LSA dimensions correlate with the interactions, and its poor performances are not particularly surprising. The fact that the LSA-based RS

Recall@	20	100	200
tf-idf-10k	0.095 (0.001)	0.251 (0.002)	0.358 (0.002)
LSA-600	0.091 (0.001)	0.256 (0.002)	0.371 (0.002)

Table 7.4: Content-based recommendation scores using the tf-idf and LSA representations: Recall at 20, 100 and 200, averaged along 10-fold cross-validation.

performs slightly better than the tf-idf RS is explained from the known LSA abilities to capture the word synonymy relations through the dimensionality reduction process (from 10,000 vocabulary to 600 singular vectors).

7.3 Exploratory analysis from a CF perspective

7.3.1 Quality of the collaborative filtering matrix

The quality of the $|\mathcal{U}| \times |\mathcal{I}|$ collaborative filtering matrix $\mathcal{M}$ is assessed as follows. A pruned matrix is defined by selecting 10% of the (u, i) pairs, such that $\mathcal{M}_{u,i} = 1$ and flipping them to 0 (removing at most 1 item per user). Recommendation is done for each user among all items i such that $\mathcal{M}_{u,i} \neq 1$.

Two mainstream neighborhood approaches (see Sec. 6.5) are used to re-estimate all $\mathcal{M}_{u,i}$ such that $\mathcal{M}_{u,i} = 0$, with:

$$\widehat{\mathcal{M}}_{u,i} = \sum_{j=1}^{|\mathcal{I}|} \text{sim}(i,j)\mathcal{M}_{u,i} \tag{7.1}$$

- **content neighborhood recommendation**: $\text{sim}(i, j)$ stands for the cosine similarity among the initial or reduced representations of the i-th and j-th items. Only the reduced LSA representation is considered in this section (the tf-idf representation brings no improvement upon LSA); the associated recommendation will be referred to as LSA-based recommendation and denoted **LSA-bl** in the following.
- $\mathcal{M}$ **neighborhood recommendation**, denoted $\mathcal{M}$-bl in the following: $\text{sim}(i, j)$ stands for the cosine similarity between their collaborative representations $\mathcal{M}_{\cdot,i}$ and $\mathcal{M}_{\cdot,j}$, that is, each item is represented as the set of users that interacted with it.

An item-oriented neighborhood approach is used here to support the item semi-cold start setting. Additionally, the neighborhood of the i-th item is defined considering the full item set for simplicity.

As previously, the quality of the data is estimated from the *recall@T* performance indicator averaged over 10 runs, for both neighborhood methods. The LSA representation uses 600 dimensions for the Qapa and ABG datasets, and 200 dimensions for the CiteULike dataset.

As displayed in Fig. 7.4, for 52% of Qapa (resp. 44% of ABG) applicants, the relevant recommendations appear in the top-200 (out of 56,000 job ads for Qapa and 14,000 for ABG). For a significant fraction of the applicants however, the first relevant recommendation is ranked below the top 10,000, being thus useless in a real-world setting.

For all databases, the $\mathcal{M}$-based (item-based) similarity is more effective than the LSA one when very few recommendations are displayed; a tentative interpretation is that the similarity of any two job ads is better captured from the applicant behaviors, than from

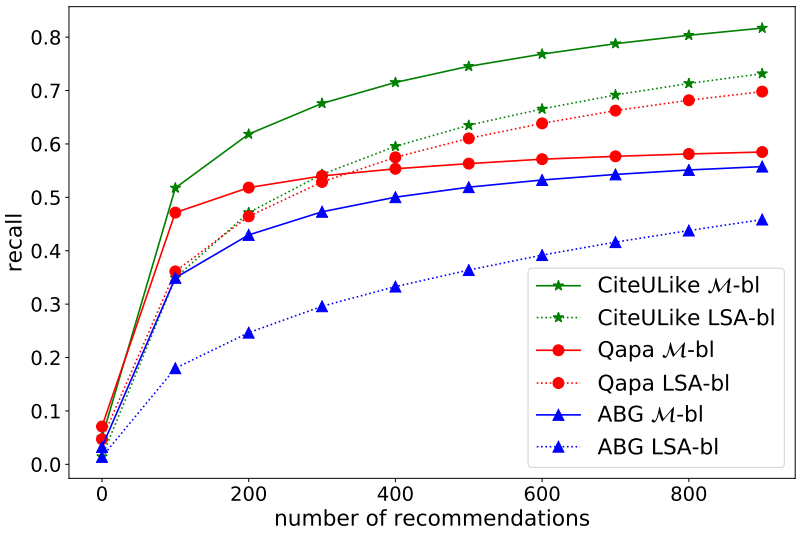

Figure 7.4: Recall curves for warm-start recommendation on CiteULike (average on 5-fold cross-validation, LSA with dimension $D = 200$), Qapa and ABG (average on 10 runs, LSA with $D = 600$). For CiteULike and ABG, $\mathcal{M}$-bl dominates LSA-bl: user behaviors are more informative than item contents. For Qapa, $\mathcal{M}$-bl dominates LSA-bl at first, and LSA-bl catches up when $\mathcal{M}$-bl reaches a plateau, blamed on the $\mathcal{M}$-sparsity. Standard deviation, omitted for readability, is less than 0.005 except for $\mathcal{M}$-based on CiteULike (≈ 0.03), due to the 5-th fold.

the actual wording of the job ad (even after the dimensionality reduction achieved by LSA).

7.3.2 Alignment of content and collaborative data

The consistency of the textual representation of item documents with the interaction data is investigated by comparing the cosine similarities on $\mathcal{M}$-based and on LSA representations, respectively, through a neighbors retrieval task.

Specifically, to each item is associated the set of its κ nearest neighbors according to the $\mathcal{M}$ similarity, considered as the "relevant" documents to be retrieved. The "recall@T" for an item is set to 1 iff one of its neighbors is among the top T-ones, ranked by decreasing LSA similarity. As seen from the average recall curve (Fig. 7.5), the standalone LSA similarity provides relevant hints for CiteULike and Qapa (at least at a coarse level: when κ is increased from 1 to 10, all recall curves decrease very fast). For ABG, the curve is almost linear, suggesting that LSA similarity is locally uncorrelated with $\mathcal{M}$ similarity.

7.3.3 User regions of interest

A complementary experiment is conducted to confirm the adverse impact of the user's interest diversity on recommendation, as follows. Let us associate with each u-th CiteULike user the center of mass of his selected articles in LSA representation denoted Θ_u. The user is characterized from:

- the local density of his interest region (normalized sum of the squared distances of the 10-nearest items w.r.t Θ_u);

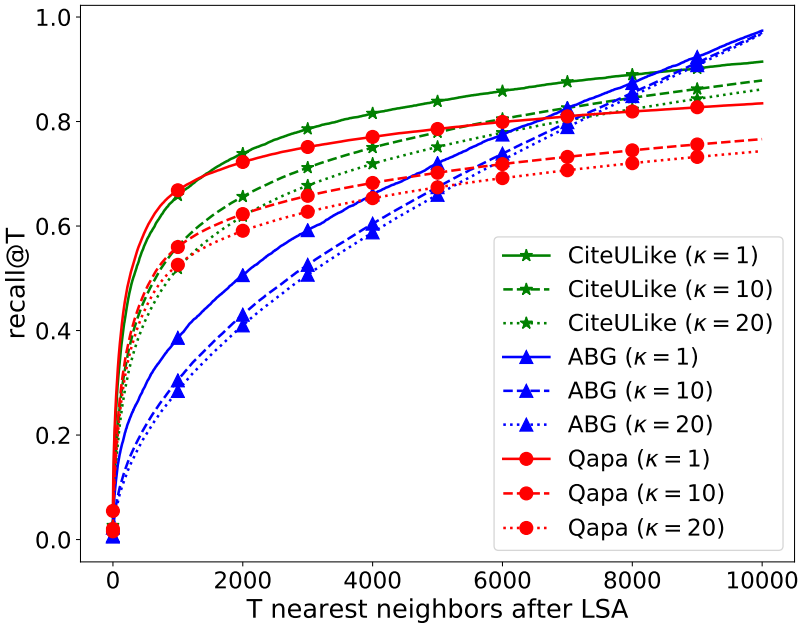

Figure 7.5: Local correlation of the $\mathcal{M}$ and LSA metrics for Qapa, ABG and CiteULike, measured from the neighbor retrieval task (see text): average fraction $y(T)$ of the items top-κ $\mathcal{M}$-neighbors that are among the top-T LSA neighbors, for $\kappa = 1, 10, 20$. The fact that the recall curve ultimately reaches 100% on ABG is explained as ABG involves about 10,000 items.

- the dispersion of his interests (normalized sum of the squared distances of the selected items w.r.t Θ_u).

The "recommendation difficulty" of the user is measured by the rank of the first relevant recommendation after LSA-bl. This experiment is only conducted on the CiteULike dataset as applicants of Qapa (resp. ABG) have interacted with too few items (on average resp. 4 and 6), making the dispersion estimate unreliable.

This experiment shows (Fig. 7.6) that the difficulty increases as the user's center of interest lies in a lower density region, as could have been expected (rightmost points represent users who gets inaccurate recommendations). Conditionally on a given density, however, the users with a high dispersion are better served, confirming that LSA-bl can retrieve relevant items close to previously selected items. This is a strong point in favour of the use of a MROI approach (recall that LSA-bl is a neighborhood approach, Sec. 6.5) for the CiteULike dataset.

7.4 First conclusions on JAM problem

This chapter has presented and analyzed the two JAM datasets that drove this work. A thoroughly studied reference dataset, the CiteULike benchmark, is also presented for the sake of comparisons.

The NLP analysis indicates that applicants and recruiters speak different languages, limiting the usefulness of recommendation approaches based on pure word matching operators. The direct matching approach turns out to be unable to extract knowledge from the CF matrix when operating at the level of words (or even at the level of more complex structures such as LSA singular vectors) in applicant or recruiter documents. This calls

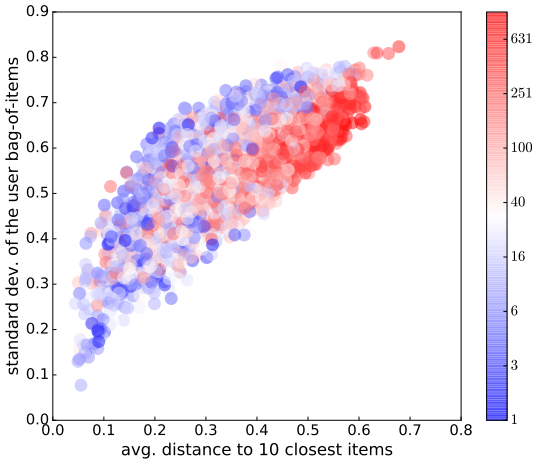

Figure 7.6: Characterization of recommendation difficulty on CiteULike. Each user is associated with the barycenter Θ_u of his items in LSA representation, and located at position (x, y), where x is the 10-nearest item dispersion in the neighborhood of Θ_u (the highest, the lowest the density), and y is the dispersion of the selected items w.r.t. Θ_u. The user's color reflect the rank of the first relevant recommendation (in color, log-scale) by LSA-bl.

for a more complex approach as far as NLP is involved: some *machine translation* between resumes and jobs is needed.

In a collaborative filtering perspective, the pure CF $\mathcal{M}$-based method is evaluated on the Qapa and ABG datasets in warm-start mode to assess the quality of the CF matrix. Though it efficiently provides relevant recommendations for half the users, this approach is unable to provide the other half with relevant recommendations; such disappointing result is blamed on the sparsity of the interaction matrix. Besides, this approach is inadequate for item semi cold start. Note that for the CiteULike dataset, $\mathcal{M}$-based is comparatively more successful; this is explained by the lesser sparsity of $\mathcal{M}$ (about 10 times that of the Qapa and ABG datasets).

As illustrated on the Qapa dataset, vanilla CF methods solely based on interaction data are overall far more successful than content-based approaches. In a nutshell, interactions contain more information relevant to recommendation than contents.

The interest of using an MROI approach over an SROI algorithm is experimentally demonstrated on the Qapa dataset in the Jam context. Using the publicly available CiteULike data indicates that the difficulty of providing good recommendations is related the multiple interests of users.

Chapter 8

Neural architectures for recommendation

As emphasized in Sec. 6.5, a central issue for recommender systems is the representation(s) of the users and items. These representations are thereafter used along different learning schemes, chiefly learning to rank and metric learning.

The dominant approach in representation learning since a decade is based on neural networks, powerful and multi-purpose learning frameworks. As far as JAM problems are concerned, they have successfully been used for recommendation [Salakhutdinov et al., 2007], ranking for information retrieval [Hoffer and Ailon, 2015], text similarity learning [Mueller and Thyagarajan, 2016], text embedding [Le and Mikolov, 2014], machine translation [Cho et al., 2014].

The contributions presented in this chapter are two feed-forward neural network architectures learned as *language models* for recommendation. They are validated on the JAM problem datasets introduced in Chap. 7, namely the Qapa and ABG datasets. For the sake of comparative evaluation, they are also assessed on the publicly available CiteULike benchmark concerned with scientific article recommendation.

After presenting the problem statement and briefly describing the state of the art in learning to rank and metric learning with neural nets, this chapter details our contributions. This chapter is partially based on [Schmitt et al., 2017].

8.1 Positioning of the problem

This section restates the problem statement first introduced in Sec. 6.3.1, focusing on the hypothesis space defined by neural networks (NNs).

Item-oriented neighborhood (ION) methods have shown to be appropriate to JAM, as far as LSA-bl yields reasonable performances in comparison with simple content-based approaches (Sec. 7.3.1). In the warm-start setting, ION methods are dominated by mainstream collaborative filtering when considering the recall@T performance indicator for small values of T. However, ION methods handle the semi-cold start setting and can leverage both items textual descriptions and interaction data. The aggregation function to form the recommendation list (Eq. 6.2) is as follows:

$$\forall u \in \mathcal{U}, \widehat{\mathcal{M}}_{u,i} = \sum_{j \in \mathcal{I}} \text{sim}(i,j)\mathcal{M}_{u,j} = \sum_{j:\mathcal{M}_{u,j}=1} \text{sim}(i,j), \tag{8.1}$$

where $\text{sim}(i,j)$ denotes the similarity between items i and j. In comparison to some other

neighborhood approaches (Sec. 6.5.2), all items are retained in the neighborhood of u ($\mathcal{V}(u) = \mathcal{I}$) and not only a fixed number of nearest neighbors.

Recommendation list $\mathcal{R}_T(u)$ is built by ranking in decreasing order the estimated $\widehat{\mathcal{M}}_{u,i}$ values. The fact that item i be recommended to user u depends on the estimated $\widehat{\mathcal{M}}_{u,i}$, and specifically on the rank $r_{u,i}$ of $\widehat{\mathcal{M}}_{u,i}$ in the set of values $\widehat{\mathcal{M}}_{u,\cdot}$; item i is recommended to user u iff $r_{u,i}$ is less than T:

$$\tau(\widehat{\mathcal{M}}_{u,i}, \widehat{\mathcal{M}}_{u,\cdot}, T) = \begin{cases} 1 & \text{if } r_{u,i} < T \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

Overall, performance indicator $R@T$ (Sec. 6.3.4) averages over all users the recall@T per user:

$$R@T = \sum_{u \in \mathcal{U}} \frac{\sum_{i:\mathcal{M}_{u,i}=1} \tau(\widehat{\mathcal{M}}_{u,i}, \widehat{\mathcal{M}}_{u,\cdot}, T)}{\|\mathcal{M}_{u,\cdot}\|_1}, \tag{8.3}$$

with $\|\mathcal{M}_{u,\cdot}\|_1 = \sum_{l \in \mathcal{I}} \mathcal{M}_{u,l}$ is the number of items clicked upon by user u.

As far as $r_{u,i}$ depends on the ordering of $\widehat{\mathcal{M}}_{u,\cdot}$ coefficients, the recall@T indicator can hardly be optimized directly. Besides, the size of the problem makes it intractable to apply combinatorial optimization algorithms. The next section briefly presents the state of the art in neural networks applied to ranking and metric learning.

8.2 Formal background

8.2.1 Neural learning to rank

Linear mapping

In a similar ranking context, [Weimer et al., 2008] use the Polya-Littlewood-Hardy inequality that states that the inner product between to vectors is maximized if they share the same ranking:

$$\forall a, b \in \mathbb{R}^d, \langle a, b \rangle \leq \langle sort(a), sort(b) \rangle$$

Accordingly, the optimization of Eq. (8.3) is tackled by aligning the top-T elements of the indicator vectors for each user u and the ground truth vector $[\mathcal{M}_{u,i}]_i$. The linear approach proceeds by optimizing a proxy of the recall, considering for each user:

$$\forall u \in \mathcal{U}, r(u) = \frac{\sum_{i:\mathcal{M}_{u,i}=1} \widehat{\mathcal{M}}_{u,i}}{\|\mathcal{M}_{u,\cdot}\|_1} \tag{8.4}$$

Optimizing the alignment of both vectors is particularly relevant as it enables to simultaneously consider the recall@T performance indicator for various values of T.

Most importantly, $r(u)$ (Eq. 8.4) has the nice property of being linear with respect to $\widehat{\mathcal{M}}_{u,i}$, and thus to sim (Eq. 8.1), which is expected to ease the optimization w.r.t., *e.g.* parameters of sim.

In [Weimer et al., 2008], the recommendation problem involves *ratings*; in such cases the goal is to reproduce to the best extent possible the item ratings. In contrast, OCCF virtually considers two classes of items, those that are relevant and the others; the relevant items thus have same rank. A trivial solution (*e.g.* mapping all items on the same location and thus maximizing their similarity) would therefore maximize the considered criterion. A proper disincentive mechanism must thus be designed to prevent the discovery of such trivial solutions; we shall return to this point in Sec. 8.2.5.

In Eq. (8.3), all users equally affect the recommendation score. To enforce this property, sim should not favor "active" users who interacted with numerous items over those who only clicked a limited number of items. We shall return to the user activity imbalance in Sec. 8.2.5. In order to avoid active users to bias the sought solution, assuming sim to be in interval $[0, 1]$ and noticing that:

$$\widehat{\mathcal{M}}_{u,i} \leq \sum_{l \in \mathcal{I}} \mathcal{M}_{u,l} = \|\mathcal{M}_{u,\cdot}\|_1$$

(the equality occurs when sim is 1 for all relevant items), $r(u)$ is revised by considering an L_2 norm normalization. $r_c(u)$, a revised version of $r(u)$, is defined as:

$$\forall u \in \mathcal{U},\ r_c(u) = \frac{\sum_{i:\mathcal{M}_{u,i}=1} \widehat{\mathcal{M}}_{u,i}}{\|\mathcal{M}_{u,\cdot}\|_1^2} \tag{8.5}$$

Note that as the recommender system independently considers each user and ranks items according to this user, the above normalization does not modify the *per user* recommendation. It still supports the overall assessment of the recommender, aggregating the performance over all users whatever their individual activity:

$$\begin{aligned} r &= \sum_{u \in \mathcal{U}} r_c(u) \\ &= \sum_{u \in \mathcal{U}} \frac{\sum_{i:\mathcal{M}_{u,i}=1} \widehat{\mathcal{M}}_{u,i}}{\|\mathcal{M}_{u,\cdot}\|_1^2} \\ &= \sum_{u \in \mathcal{U}} \frac{\sum_{i} \widehat{\mathcal{M}}_{u,i} \cdot \mathcal{M}_{u,i}}{\|\mathcal{M}_{u,\cdot}\|_1^2} \end{aligned} \tag{8.6}$$

Replacing $\widehat{\mathcal{M}}_{u,i}$ using Eq. (8.1) leads to the following form:

$$\begin{aligned} r &= \sum_{u \in \mathcal{U}} \frac{1}{\|\mathcal{M}_{u,\cdot}\|_1^2} \sum_{i \in \mathcal{I}} \left(\sum_{j \in \mathcal{I}} \text{sim}(i,j) \mathcal{M}_{u,j} \right) \mathcal{M}_{u,i} \\ &= \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \text{sim}(i,j) \cdot \sum_{u \in \mathcal{U}} \frac{1}{\|\mathcal{M}_{u,\cdot}\|_1^2} \mathcal{M}_{u,j} \cdot \mathcal{M}_{u,i} \end{aligned}$$

Rewriting the above using matrix notations gives:

$$r = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \text{sim}(i,j) \cdot \text{sim}_{\mathcal{M}}(i,j) \tag{8.7}$$

where:

$$\forall (i,j) \in \mathcal{I} \times \mathcal{I},\ \text{sim}_{\mathcal{M}}(i,j) = \overline{\mathcal{M}}_{\cdot,j}^{\top} \cdot \overline{\mathcal{M}}_{\cdot,i} \tag{8.8}$$

and $\overline{\mathcal{M}}$ denotes the row-normalized version of $\mathcal{M}$:

$$\forall u \in \mathcal{U},\ \overline{\mathcal{M}}_{u,\cdot} = \frac{1}{\|\mathcal{M}_{u,\cdot}\|_1} \mathcal{M}_{u,\cdot}$$

Note that the user dependency no longer explicitly appears in Eq. (8.7). Intuitively, the similarity of two items reflects the number of users clicks on both items, modulated by the user activity (factor $1/\|\mathcal{M}_{u,\cdot}\|_1$).

The $[\text{sim}_{\mathcal{M}}]$ matrix can be analysed in comparison with the item-item correlation matrix $\mathcal{C}_{i,j}$ in ItemRank [Gori and Pucci, 2007]. In ItemRank, a graph-based approach, $\mathcal{C}_{i,j}$ is initialized from $\mathcal{M}_{u,j} \cdot \mathcal{M}_{u,i}$; a normalization process is then applied in order for matrix $\mathcal{C}$ to be interpretable as a transition matrix where $\mathcal{C}_{i,j}$ represents the probability to move from node j to node i. In contrast with [Gori and Pucci, 2007], the proposed approach defines a symmetric $[\text{sim}_{\mathcal{M}}]$ similarity.

Note that in some cases, a symmetrical similarity might be inappropriate, *e.g.* when several users interact with items j and k while some users only interact with item j. In such cases, one might want item k to be similar to item j and at the same time item j be not too similar to item k. The design of asymmetrical similarities is left for further research.

8.2.2 Distance metric learning

A metric is defined as a mapping $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ over a vector space $\mathcal{X}$ which satisfies the following properties [Weinberger and Saul, 2009]:

1. $\forall x, y \in \mathcal{X},\ d(x, y) \geq 0$ (non-negativity)
2. $\forall x, y \in \mathcal{X},\ d(x, y) = d(y, x)$ (symmetry)
3. $\forall x, y, z \in \mathcal{X},\ d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)
4. $\forall x, y \in \mathcal{X},\ d(x, y) = 0 \implies x = y$ (distinguishability)

The problem of distance metric learning has been extensively considered in the machine learning literature; see among many others [Xing et al., 2003; Weinberger and Saul, 2009]. The metric learning problem is defined as learning a mapping over pairs of elements in the instance space $\mathcal{X}$ which emulates some similarity relation S which is usually given on some training triplets:

$$\forall x, y, z \in \mathcal{X},\ S(x, y) = 1 \wedge S(x, z) = 0 \implies d(x, y) \leq d(x, z) + m \tag{8.9}$$

where $m > 0$ is referred to as the *margin*; it enforces that similar and dissimilar pairs should appear significantly different. As Eq. (8.9) still holds if d is rescaled, m is usually set to 1.

As usual, the goal of metric learning is to define some d which both fits the training triplets and generalizes to further triplets.

The linear mapping introduced in Eq. (8.4) directly connects to metric learning, with $\text{sim}_{\mathcal{M}}$ in Eq. (8.8) defining a pairwise similarity between items.

As shown by [Xing et al., 2003; Shalev-Shwartz et al., 2004], there exist computationally efficient approaches to build a distance complying with similarity constraints of the type $sim(x, y) < sim(x, z)$. Such approaches mostly consider Mahalanobis distances or semi-distances. A Mahalanobis semi-distance on vector space $\mathcal{X}$ is defined from a symmetric positive semi-definite matrix A as

$$\forall x, y \in \mathcal{X}^2,\ d_A(x, y) = \sqrt{(x - y)^\top A (x - y)}$$

This semi-distance satisfies axioms 1-3. It satisfies the 4th axiom of distinguishability iff A is definite; in this case it is a distance. Note that the distinguishability is not always required in metric learning.

Granted that items are expressed in natural language, a non-linear similarity approach might be desirable in some cases. Related work on non-linear metrics learning mostly

extend the linear setting using kernelization [Kwok and Tsang, 2003]. Another option consists of learning a linear metric based on *local* constraints only. Yang et al. [2006] use a k-nearest neighbor approach to predict the class of a point x. If each point x falls in a neighborhood that is "sufficiently" populated with points of the same class, it does not matter if there exist other points similar to x that are far away. [Weinberger and Saul, 2009] enforces the property that all k nearest neighbors of a point share its same label by attracting them and repelling neighboring points with other labels. To ensure robustness, a fixed safety margin between the farthest point of the same class and the nearest point from another class is maintained. This algorithmic approach is flexible and allows to consider non-symmetric similarity during training. [Qamar and Gaussier, 2012] focus on learning similarities of the form:

$$\forall x, y \in \mathcal{X}^2,\ s_A(x, y) = \frac{x^\top A y}{N(x, y)},$$

where A is a square similarity matrix and $N(x, y)$ is a normalization coefficient.[1] They note that minimizing the number of mistakes a k-nearest neighbor classifier would make is more relevant for classification applications than maximizing the average safety margin. Such local approaches are well suited to problem domains where a class can involve several modes or sub-classes. As a conclusion, [Yang et al., 2006] note that the main difficulty of pseudo-metrics learning lies in the positive semi-definiteness constraint, which is not required in a recommendation setting.

8.2.3 Siamese network

Siamese network have been introduced for signature [Bromley et al., 1993] and face [Chopra et al., 2005]) verification problems. A Siamese neural network architecture is composed of two branches with shared architecture and weights (Fig. 8.1). The aim is to be able to recognize if two inputs fed to the two branches refer to a same entity (two signatures or two face images of a same person).

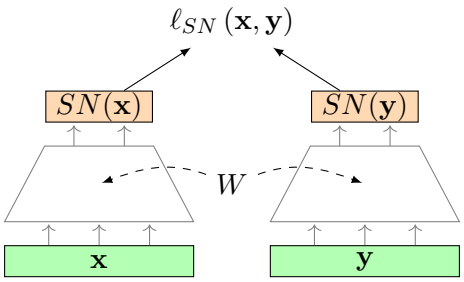

Figure 8.1: The Siamese network architecture: the two network branches share the same weights W.

During the training phase, Siamese networks are presented with so-called *positive* or *genuine* pairs (*i.e.* pairs of inputs related to a same object) with the goal of minimizing the distance between their images in the output or representation space. This goal is enforced by the so-called *attractive loss* defined as:

$$\ell^+(x, y^+) = d(SN(x), SN(y^+))^2 \tag{8.10}$$

[1] For $A = I$ and $N(x, y) = \|x\|_2 \cdot \|y\|_2$, s_A is the cosine similarity.

with $SN(z)$ the image of z produced by the Siamese network.

This loss function, however, admits a trivial solution, as $SN : z \mapsto 0$ minimizes it. Therefore, Siamese network training also relies on *negative* or *impostor* pairs, *i.e.* inputs not related to a same object that the network has to push away from each other. Several *repulsive loss* functions have been considered in the literature (see for instance [Hadsell et al., 2006] and references therein). Among the most studied losses is the convex loss [Hadsell et al., 2006]:

$$\ell^-(x, y^-) = \left[m - d(SN(x), SN(y^-)\right]_+^2 \tag{8.11}$$

where m is a hyperparameter referred to as *margin* and $[A]_+ = max(A, 0)$). This repulsive term is designed to take into account only pairs of points that are currently close (with a distance lesser than m) and that would be perceived as positive by a classifier only relying on the distance value[2]. The pairs of points that are far away from each other do not need to be considered as they do not have any impact on the prediction of the current point.

The repulsive term conveys the *context* of the decision making problem achieved by the Siamese network output space; it characterizes to which extent x and y are similar with respect to the decision context. To illustrate this point, consider the faces from two humans from different people: they are *close* if the point is to discriminate mammals from other beings, while they are *different* from the perspective of human identity verification. This repulsive term accounts for the context, as it tends to push away $SN(y)$ from the m-neighborhood of $SN(x)$ when relevant.

The final loss sums the attractive terms (active for positive pairs) and the repulsive terms (active for negative pairs). Denoting $\delta^+(x, y)$ the indicator function that evaluates to 1 if (x, y) is a positive pair and 0 otherwise, the network is finally learned to minimize:

$$\ell_{SN}(x, y) = \delta^+(x, y) \cdot \ell^+(x, y) + (1 - \delta^+(x, y)) \cdot \ell^-(x, y) \tag{8.12}$$

As generally acknowledged (see [Chopra et al., 2005] among others), a desirable property of the loss is to be convex with respect to d; the loss we use (Eq. (8.12) is indeed convex; attractive and repulsive terms are quadratic functions of d.

In principle, the number of pairs used to train the Siamese network scales quadratically with the size of the training set. For recommendation applications, however, Siamese networks are trained to emulate a (very) sparse collaborative matrix. The number of training pairs should thus preferably grow linearly with the number of known interactions (pairs of items that are known to be similar or dissimilar). Note that in the OCCF context, known interactions are all positive. Our approach takes inspiration from the related problem of face verification [Chopra et al., 2005], where negative pairs include all non-positive pairs and outnumber the positive pairs by several orders of magnitude. Accordingly, the negative pairs used to train the Siamese network are sampled among the overall negative pairs to achieve a good trade-off between the computational cost and the training efficiency.

Siamese networks have been successfully used to achieve topology-preserving dimensionality reduction [Hadsell et al., 2006] and for text applications, such as sentence similarity learning [Mueller and Thyagarajan, 2016] or title normalization and classification [Neculoiu et al., 2016].

[2]This contrasts with distance metric learning approaches seen in the Sec. 8.2.2, where the distance *relatively to other points* is used to infer the current point label [Weinberger and Saul, 2009; Yang et al., 2006].

8.2.4 Triplet network for preference learning

Triplet networks extend the Siamese architecture and consider three branches instead of two, with all branches sharing the same architecture and weights; Fig. 8.2). The network is simultaneously provided with one reference and two test items, and the aim is to learn which one out of the two test items is the nearest to the reference in terms of the Euclidean distance in the network output representation. This learning scheme is well suited to recommendation or information retrieval [Wang et al., 2014] as most recommender systems performance measures (Sec. 6.3.4) are based on ranking.

Let (x, y^+, y^-) denote a triplet, such that x is more similar to y^+ than to y^-. The loss function is meant to enforce the same similarity relations in the metric space defined by the network output. Accordingly, the triplet network loss function is defined as follows (taking inspiration from the Hinge loss for classification problems):

$$\ell_{TN}(x, y^+, y^-) = \left[d(TN(x), TN(y^+)) - d(TN(x), TN(y^-))\right]_+ \tag{8.13}$$

A fixed margin (omitted here) is optionally added to increase the contrast in distances in the output space.

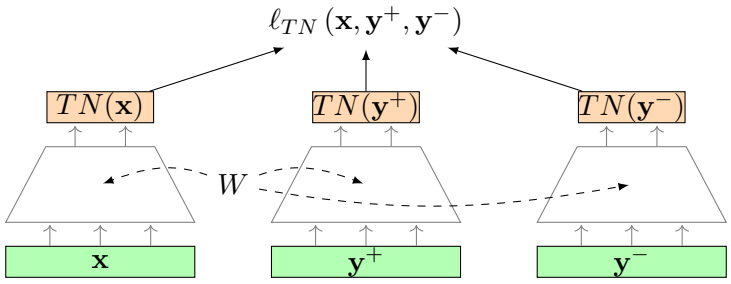

Figure 8.2: The triplet network architecture: All three network branches share the same weights W.

Triplet networks, however, face a strong limitation in terms of scalability, with a computational complexity cubic in the size of the training set. Again, some aggressive subsampling of the triplets of examples is required to support the approach [Wang et al., 2014].

8.2.5 Negative feedback

Both metrics learning and the previously discussed NN-based approaches require to provide positive and negative examples during training. In standard recommendation settings, users give explicit feedback on items through ratings (*e.g.* number of stars in the Netflix datase [Bennett et al., 2007] or like/dislike on online streaming platforms) or annotations (user comments after purchase on e-commerce websites[3]). In the OCCF setting, positive examples are classically sampled among the known positive interactions (1s of the collaborative matrix); most usually, there are no negative examples, as explicit feedback denotes some interest. The lack of explicit negative feedback is seen in many recommendation settings; for OCCF settings, this lack is even more severe, as explicit negative interactions would be needed to counterbalance the positive interactions.

When possible, negative examples can be extracted from the list of items displayed to the user that have not been clicked [Das et al., 2007]. Still, the reliability of these negative

[3]The field of sentiment analysis is beyond the scope of the present document.

examples is questionable. On the one hand, there is no guarantee that the user actually looked at the item (*e.g.* in information retrieval users are often only interested in the first few results and do not even look at the next few suggestions [Joachims et al., 2005, 2007]). The user's choices might themselves be unreliable: if the user is only looking for one result (*e.g.* for a film recommendation), they might simply neglect other options even if they *were* actually relevant. Obviously the same could be said about positive examples: they may be relevant not intrinsically but only in comparison to the other recommendations. [Joachims et al., 2005] conclude that clicks are more accurately interpreted as positive feedback *relatively* to other recommended items (and depending on the user trust in the recommendation engine) than as *absolute* positive signal. To overcome such biases, [Radlinski and Joachims, 2007] investigate how *active* data collection can lead to quicker learning. Ideally, an active learning setting would require feedback from real users about the recommender system under training. However, this requirement is hardly met in practice − with the notable exception of the online evaluation of the RecSys challenge 2017 [Abel et al., 2017]. The authors of [Radlinski and Joachims, 2007] propose a simplified user behavior model to be able to simulate unknown data and perform experiments on offline datasets.

Specifically for OCCF, [Paquet and Koenigstein, 2013] investigate two learning problems: i) predict whether a user has encountered a particular item; and ii) whether the user has positively interacted with the item. The former is achieved through modeling the unobserved data as a hidden graph of which the observed data (considered as a user-item graph) should be a subgraph. Note that if the data were (partially) obtained using a RS, this amounts to learning this RS behavior.

When no "soft-negative" examples are available or user behavior cannot be reliably simulated, OCCF relies on sampling negative examples from the unobserved data (that is, all non-positive elements of $\mathcal{M}$). The two extreme strategies presented below have been investigated by [Pan et al., 2008], together with intermediate ones.

All Missing As Negative (AMAN)

In the AMAN paradigm, all examples not observed as positive are considered to be negative, yielding a severely imbalanced problem (negative examples outnumber the positive ones by several orders of magnitude) and adversely affecting the learning task for computational and efficiency reasons. Furthermore, it is likely that some of the unobserved data *would* be positive if observed – this is even the base hypothesis of warm-start recommendation. The AMAN strategy thus induces a novelty-adverse bias.

All Missing As Unknown (AMAU)

The other extreme case considers that all non-observed data are unknown and might actually be positive. As already said, this paradigm is not appropriate for a metric learning goal as it is compatible with the trivial solution where all items are relevant to all users. However, it is meaningful when there exist different feedback levels (*e.g.* , ratings), as the user behavior can be learned from the observed data only.

Intermediate approaches

Intermediate approaches proceed by weighting or sampling unlabelled examples. One may want to consider all missing as possible negative values, but with a low confidence (as they can be unobserved "ones"). Following this idea, [Pan et al., 2008] propose a

matrix decomposition loss with low weights assigned to reconstruct the "zeroes" (while "ones" should be reconstructed with unitary weight). A similar approach is taken by [Hu et al., 2008], seeking low-rank matrices to approximate the (positive interaction only) $\mathcal{M}$ and using regularization terms. A similar idea is found in [Wang and Blei, 2011], where both the baseline and the contributed approach are based on generative probabilistic models reconstructing "ones" with weight 1 and "zeroes" with a lower weight (0.01 in the experiments). To the best of our knowledge, this lower weight is not derived from, *e.g.* the matrix sparsity; it is empirical adjusted. Another option is to sub-sample the unlabelled pairs and consider the sample as "zeroes". When using such a sampling scheme, particular attention should be payed to the "zeroes" samples so that no bias is introduced: for instance, this set should be periodically refreshed with new samples.

Both the weighting and the sampling scheme may encapsulate prior knowledge on the users behavior and/or the items, such as:

- *uniformity*: all weights assigned to negative samples are equal / negative samples are uniformly drawn from the set of unlabelled examples;
- *user-oriented*: a user with (comparatively to the other) many interactions is likely to have examined more carefully their list of recommendation. Therefore, their unlabelled data are thus more likely to be true "zeroes", and are thus decorated with higher weights or sampled with higher probability than by default [Pan et al., 2008]. This kind of reasoning implicitly assumes that each user has roughly the same amount of interest that he or she distributes over the set of items.
- *item-oriented*: the basic idea is to favor "popular" items, be it for the sake of providing higher quality recommendations (the item is popular because it is intrinsically better than average) or increasing the evaluation score (this item has statistically higher chances to draw the interest of a new user). It follows that the weight or the probability to sample an item as a "zero" should decrease with the number of known positive interactions. The non-personalized, popularity-based recommendation strategy [Rashid et al., 2002] is based on this assumption.

8.3 LaJam: Learning a Language Model for recommendation

This section presents the proposed *LAnguage model-based* Jam (LaJam) system, focusing on the semi-cold start problem of recommending brand new job ads to a known user. As stated previously, this approach is suited to the temporary work sector, and therefore of prime importance in today's labor market.[4]

Exploiting the fact that neighborhood methods can accommodate any representation, LaJam proceeds in two steps:

1. learning a mapping from the job ad space onto $\mathbb{R}^d$, such that this mapping reflects the job similarity derived from matrix $\mathcal{M}$;
2. using the similarity that result from this mapping together with $\mathcal{M}$ in a neighborhood approach to rank the job ads for each applicant.

[4] In France, the proportion of the temporary work sector has reached its all-time high in June 2017 (source: `http://dares.travail-emploi.gouv.fr/IMG/pdf/cahier_graphique_conjoncture_2017t2.pdf` (in French)).

As such, LaJam is applicable to the item semi cold-start setting (like LSA-bl, Sec. 7.3.1) but uses an item representation suited for recommendation (while LSA-bl results in purely unsupervised learning).

8.3.1 Continuous language model for job ads

LaJam is designed to produce recommendations that optimize the per-user recall measure. A tractable optimization problem is derived in Sec. 8.2.1 to tackle the per-user recall measure and results in the optimization of Eq. (8.7), in which users do not appear explicitly. Recall that the similarity induced by the collaborative filtering matrix is denoted $\text{sim}_{\mathcal{M}}$:

$$\text{sim}_{\mathcal{M}} = \overline{\mathcal{M}}^{\top} \cdot \overline{\mathcal{M}} \tag{8.14}$$

The language model ϕ, represented by a parameter vector $W \in \mathbb{R}^L$, maps the initial or reduced description space (tf-idf, LSA or LDA) of the job ads onto $\mathbb{R}^d$, such that it induces a similarity noted sim_W compliant with $\text{sim}_{\mathcal{M}}$. Eq. (8.7) then reads as the inner-product of two similarity distributions defined on the same discrete set $\mathcal{I}$:

$$r(W) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \text{sim}_W(i,j) \cdot \text{sim}_{\mathcal{M}}(i,j) \tag{8.15}$$

To avoid trivial cases (such as sim_W locally unbounded or all pairs having the maximum positive similarity value) we will make two further assumptions: i) sim_W is bounded inside interval $[-1,1]$ and ii) $\text{sim}_{\mathcal{M}}(i,j)$ takes both positive and negative values. Using once again the Polya-Littlewood-Hardy inequality, $r(W)$ is then maximized w.r.t. W if sim_W is compliant with $\text{sim}_{\mathcal{M}}$ ordering on $\mathcal{I}$. Mapping ϕ is learned using a Siamese neural architecture (see Sec. 8.2.3 and Fig. 8.3). The elementary loss associated with any $(\mathbf{y}_i, \mathbf{y}_j)$ job ad pair is the sum of two terms to be minimized:

$$\ell_W(\mathbf{y}_i, \mathbf{y}_j) = \delta^+(\mathbf{y}_i, \mathbf{y}_j) \cdot d(\phi_W(\mathbf{y}_i), \phi_W(\mathbf{y}_j))^2 + \delta^-(\mathbf{y}_i, \mathbf{y}_j) \cdot [m - d(\phi_W(\mathbf{y}_i), \phi_W(\mathbf{y}_j))]_+^2 \tag{8.16}$$

where $\delta^+(\mathbf{y}_i, \mathbf{y}_j)$ indicates whether the pair $(\mathbf{y}_i, \mathbf{y}_j)$ is positive or not and $\delta^-(\mathbf{y}_i, \mathbf{y}_j)$ is introduced for convenience and defined as:

$$\delta^-(\mathbf{y}_i, \mathbf{y}_j) = (1 - \delta^+(\mathbf{y}_i, \mathbf{y}_j))$$

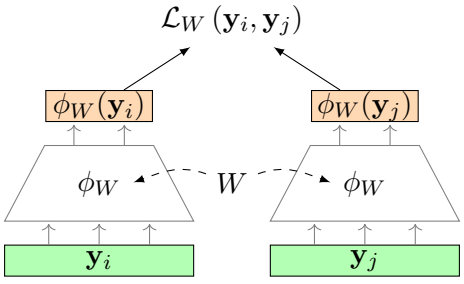

Figure 8.3: LaJam neural architecture: Mapping ϕ from the job ad space onto $\mathbb{R}^d$ is trained using a Siamese neural net architecture, such that the sim_W (ϕ-based similarity) agrees with $\text{sim}_{\mathcal{M}}$.

Following previous works on word and document embedding [Mikolov et al., 2013b] as well as metric learning [Nguyen and Bai, 2010], the similarity $\text{sim}_W(\mathbf{y}_i, \mathbf{y}_j)$ between the

embeddings of items $\mathbf{y}_i$ and $\mathbf{y}_j$ is taken as the cosine of their representation, and a pseudo-distance d_W is derived therefrom:

$$\mathrm{d}_W(\mathbf{y}_i, \mathbf{y}_j) = d_{\cos}\left(\phi_W(\mathbf{y}_i), \phi_W(\mathbf{y}_j)\right) = 1 - \frac{\langle \phi_W(\mathbf{y}_i), \phi_W(\mathbf{y}_j) \rangle}{\|\phi_W(\mathbf{y}_i)\|_2 \cdot \|\phi_W(\mathbf{y}_j)\|_2} \tag{8.17}$$

Taking inspiration from Eq. (8.16), the Siamese network elementary loss function is defined as:

$$\begin{aligned} \mathcal{L}_W(\mathbf{y}_i, \mathbf{y}_j) = \delta^+(\mathbf{y}_i, \mathbf{y}_j) \cdot \mathrm{d}_W(\mathbf{y}_i, \mathbf{y}_j) \\ + \delta^-(\mathbf{y}_i, \mathbf{y}_j) \cdot [m - \mathrm{d}_W(\mathbf{y}_i, \mathbf{y}_j))]_+ \end{aligned} \tag{8.18}$$

This loss function differs from that of the original Siamese network (Eq. 8.12) in several ways that are briefly discussed.

The distance over the network output space has been replaced by a pseudo-distance which does not satisfy the triangle inequality property.[5] Recall that Siamese networks were originally designed for verification (or classification with a high number of classes). The transitivity property is sensible in that context: if both items x and y are separately taken as positive examples with the same object z, then the pair (x, y) has to be positive as well since all three objects share the same class. This is automatically learned by the Siamese network when d verifies the triangle inequality: $d(x, y) \leq d(x, z) + d(y, z)$ making $d(x, y)$ being *also* minimized during training even if (x, y) is not provided as a positive pair. Such transitivity typically does not holds in the recommendation context. As a consequence, triangule inequality is undesirable for LaJam. This makes the pseudo-cosine distance preferable to the Euclidean distance.

Besides, the terms in Eq. (8.18) are no longer quadratic in the distance. Recall that the sole purpose of using square distances is to enforce the convexity of the optimization problem; [Nguyen and Bai, 2010] show for a slightly different objective function that cosine similarity-based objective function can be efficiently optimized through gradient-descent algorithms without requiring extra transformation.

A final decision regards the margin m, which is set to 1. This is a natural choice when considering the cosine distance: pairs of similar items are encouraged to have positively similar representations, while dissimilar pairs do not require their images to be negatively similar (a similarity of zero is sufficient). We, however, did not try to optimize this parameter. To make the connection with Eq. (8.15), we define:

$$\mathrm{sim}_W(i, j) = 1 - \mathrm{d}_W(\mathbf{y}_i, \mathbf{y}_j) = \cos\left(\phi_W(\mathbf{y}_i), \phi_W(\mathbf{y}_j)\right)$$

LaJam elementary loss finally reads:

$$\begin{aligned} \mathcal{L}_W(\mathbf{y}_i, \mathbf{y}_j) = \delta^+(\mathbf{y}_i, \mathbf{y}_j) \cdot (1 - \mathrm{sim}_W(i, j)) \\ + \delta^-(\mathbf{y}_i, \mathbf{y}_j) \cdot [\mathrm{sim}_W(i, j))]_+ \end{aligned} \tag{8.19}$$

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

$$\mathcal{L}(W) = \sum_{\mathbf{y}_i \in \mathcal{I}} \sum_{\mathbf{y}_j \in V(\mathbf{y}_i)} \mathcal{L}_W(\mathbf{y}_i, \mathbf{y}_j) \tag{8.20}$$

where $\mathbf{y}_i$ ranges over all job ads and $\mathbf{y}_j$ ranges over a subset $V(\mathbf{y}_i) \subset \mathcal{I}$ that has yet to be defined (see Sec. 8.3.3).

[5] In $\mathbb{R}^2$, consider three vectors $a = [1, 0], b = \left[\sqrt{2}/2, \sqrt{2}/2\right], c = [0, 1]$ each of unit Euclidean norm. It is easy to show that $d_{\cos}(a, c) = 1 > 2 - \sqrt{2} = d_{\cos}(a, b) + d_{\cos}(b, c)$.

8.3.2 Deriving an item-item similarity from users interactions

Binary similarity

Recommendation problems differ largely from classification as in the latter case the similarity to mimic is i) binary (two items belong to the same class or not) and ii) consistent (in the sense that it carries a transitivity property). A first option considers that the mapping onto $\mathbb{R}^d$ should indeed mimic a binary similarity, denoted as δ^+. Letting $(\mathbf{y}_i, \mathbf{y}_j)$ be a pair of job ads, $\delta^+(\mathbf{y}_i, \mathbf{y}_j)$ is set to 1 iff the i-th and j-th items have drawn the interest of at least one same user, and 0 otherwise. This corresponds to the following crude binarization of $\text{sim}_{\mathcal{M}}$:

$$\begin{aligned} \delta^+(\mathbf{y}_i, \mathbf{y}_j) &= \begin{cases} 1 & \text{if } \text{sim}_{\mathcal{M}}(\mathbf{y}_i, \mathbf{y}_j) > 0 \\ 0 & \text{otherwise} \end{cases} \\ \delta^-(\mathbf{y}_i, \mathbf{y}_j) &= 1 - \delta^+(\mathbf{y}_i, \mathbf{y}_j) \end{aligned} \tag{8.21}$$

Refined similarity: continuous weighting

However, the similarity learned by LaJam would desirably be able to reproduce an entire spectrum of similarity intensities. At the same time, learning through a Siamese-like loss function still requires that examples should be divided into those more similar (that would be captured by the attractive loss) and those dissimilar (that would be considered in the repulsive loss term).

As obtained from the product of sparse matrices (Eq. 8.14), $\text{sim}_{\mathcal{M}}$ keeps a sparse structure (although it is obviously less sparse than $\mathcal{M}$) and a simple way of providing the Siamese network negative pairs examples is to sample among the unknown values of $\text{sim}_{\mathcal{M}}$. Although it may be tempting to use all nonzero-values of $\text{sim}_{\mathcal{M}}$ as positive pairs, these may result from two items being clicked by only one shared user: this set is highly sensitive to noise. Depending on the sparsity of $\text{sim}_{\mathcal{M}}$, one may want to build a robust version of $\text{sim}_{\mathcal{M}}$[6]; however, at the cost of throwing away a number of positive pairs, while the major difficulty of training a Siamese network in the OCCF context is already that positive pairs are overwhelmed by negative ones. To mitigate the effect of noise (say, a user misclicked or clicked on an item not by interest but out of curiosity) and still keep the maximum number of positive pairs, an alternative consists in weighting positive pairs:

$$\begin{aligned} \delta^+(\mathbf{y}_i, \mathbf{y}_j) &= \begin{cases} \text{sim}_{\mathcal{M}}(\mathbf{y}_i, \mathbf{y}_j) & \text{if } \text{sim}_{\mathcal{M}}(\mathbf{y}_i, \mathbf{y}_j) > 0 \\ 0 & \text{otherwise} \end{cases} \\ \delta^-(\mathbf{y}_i, \mathbf{y}_j) &= \begin{cases} 0 & \text{if } \text{sim}_{\mathcal{M}}(\mathbf{y}_i, \mathbf{y}_j) > 0 \\ \alpha & \text{otherwise} \end{cases} \end{aligned} \tag{8.22}$$

Recall that $\text{sim}_{\mathcal{M}}$ is the cosine similarity between columns of $\mathcal{M}$, and therefore takes into account the number of users shared by a pair of items as well as a confidence weight in each user interactions. Formulated differently, users have a sort of "interest budget" that they share between items; therefore, a user who clicked two items has a weight on each of her clicks of 0.5, suggesting those two offers have a strong common degree of interest; another user who clicked 1,000 items is likely to have poor demands on the items – in other words, items are likely to have only weak similarity – and each click of that user is given weight 1/1000. A similar approach – though outside the OCCF scope – considers

[6] *e.g.* keeping as positive pairs only pairs (i, j) so that $\text{sim}_{\mathcal{M}}(i, j)$ is above a threshold or if at least two users clicked on those items

using the Pearson-r correlation coefficient between columns of $\mathcal{M}$ as the similarity between items [Gori and Pucci, 2007]. Compared to the binary similarity considered previously, the refined similarity have been experimentally shown to convey more information relevant to ranking of items.

As there is no level of dissimilarity among negative pairs, they are all assigned the same weight α, set to the average value of $\text{sim}_{\mathcal{M}}$ over its non-zeroes, *i.e.* such that the cumulative weight of the dissimilar pairs is same as the one of the similar pairs:

$$\alpha = \mathop{\mathbb{E}}_{\text{sim}_{\mathcal{M}}(i,j)>0} [\text{sim}_{\mathcal{M}}(i,j)]$$

8.3.3 Negative sampling

In principle, the computational cost is quadratic in the number of items, limiting the computational tractability of the optimization problem (Eq. 8.20). The challenge is to both enforce the compliance of sim_W w.r.t. δ^+, and the computational tractability of the approach. In practice, the dissimilar pairs outnumber the similar pairs by several orders. The idea is thus to restrict the number of considered pairs (i,j) by imposing $j \in V(i)$, where $V(i)$ includes:

1. all items $\mathbf{y}_j$ that constitute a positive pair with $\mathbf{y}_i$, *i.e.* $\{j \mid \delta^+(\mathbf{y}_i, \mathbf{y}_j) > 0\}$;
2. items selected through a negative sampling process.

In contrast with the negative feedback schemes introduced in Sec. 8.2.5, we are to select (item,item) pairs from a (symmetrical) similarity matrix and not (user, item) pairs from an interaction matrix. This makes irrelevant the *user-oriented* sampling and to a lesser extent the *item-oriented* sampling. Instead, several sampling schemes are investigated.

Row-wise uniform sampling

Following previous work on Siamese and triplet network [Chopra et al., 2005; Hadsell et al., 2006; Wang et al., 2014], each item $\mathbf{y}_i$ is associated at the same time with positive neighbors (*i.e.* items $\mathbf{y}_j^+$ that form a positive pair when associated with $\mathbf{y}_i$) and negative neighbors (*i.e.* items $\mathbf{y}_j^-$ that form a positive pair with $\mathbf{y}_i$). While $\phi(\mathbf{y}_i)$ is attracted towards the $\phi(\mathbf{y}_j^+)$[7], the negative neighbors prevent all $\phi(\mathbf{y}_j)$ from collapse towards $\phi(\mathbf{y}_i)$ and may carry relevant dissimilarity information that helps the training. Suppose that an item $\mathbf{y}_i$ has many positive neighbors; the risk is that uniform sampling would not produce a sufficient number of negative neighbors to prevent ϕ from collapsing in the $\mathbb{R}^d$-neighborhood of $\phi(\mathbf{y}_i)$[8]. Suppose now that $\mathbf{y}_i$ has few positive neighbors, then a comparatively high number of negative would overpower the attractive term and repel $\phi(\mathbf{y}_i)$ far from all other $\phi(\mathbf{y}_j)$. The *row-wise uniform sampling* (UNS) thus aims at subsampling $\delta^-(\mathbf{y}_i, \cdot)$ so that it has the same row-wise structure as δ^+, *i.e.* K as many negative as $\mathbf{y}_i$ has positive neighbors:

$$\left|\left\{\mathbf{y}_j^- \mid \delta^-(\mathbf{y}_i, \mathbf{y}_j^-) > 0\right\}\right| = K \left|\left\{\mathbf{y}_j^+ \mid \delta^+(\mathbf{y}_i, \mathbf{y}_j^+) > 0\right\}\right|$$

with K a hyper-parameter typically set to 10, thus aggressively subsampling the dissimilar items. A large K is required to help the training (recall that images $\phi(\mathbf{y}_j^-)$ of items

[7] At the same time, each $\phi(\mathbf{y}_j^+)$ is itself attracted towards $\phi(\mathbf{y}_i)$ and contributes also to $\nabla_W \mathcal{L}_{\mathcal{W}}$.

[8] For an aggressive subsampling of the negative pairs, these positive neighbors themselves, when considered in the place of $\mathbf{y}_i$, would likely not be sampled the other $\mathbf{y}_j^+$ as negative – at least we should not rely on this to prevent the collapsing.

dissimilar to $\mathbf{y}_i$ beyond the margin m – set to 0 in Eq. (8.19) – are dismissed by the use of the positive part in the repulsive loss and the dissimilarity information they carry thus does not affect the training). The K dissimilar items are uniformly sampled among the set of non-similar items.

Collaborative near-miss sampling

We introduce in this section *collaborative near-miss sampling* (CNS) which, like UNS, retains K times as many dissimilar items as similar items. The difference regards the sampling of the dissimilar items, that are selected in order to resist the transitive closure of the ϕ similarity. Specifically, if items i, j are similar, and items (j, ℓ) are also similar, CNS is meant to prevent i and ℓ to be close according to ϕ (unless i and ℓ are similar according to $\text{sim}_{\mathcal{M}}$ naturally). In this scheme, dissimilar items are sampled from the items with distance 2 in the item-item graph built from matrix $[\text{sim}_{\mathcal{M}}]_{i,j}$ (where the (i, j) edge exists iff $\text{sim}_{\mathcal{M}}(i, j) > 0$). Notably, collaborative near-miss sampling is not substantially different from row-wise uniform sampling for CiteULike: due to the comparatively lesser sparsity of $\mathcal{M}$, the diameter[9] of the item-item graph is 2.3, implying that the 2-step neighbors of an item include most items. For the Qapa and ABG item graphs, however, the diameter is much larger (respectively 6.3 for Qapa and 8 for ABG).

An interpretation of CNS considers a two-steps diffusion on the graph induced by matrix $[\text{sim}_{\mathcal{M}}]_{i,j}$. Given a user, initially represented in the graph as the set of items she interacted with, CNS samples items that are reachable in exactly two steps (that is, in two steps, but not reachable in one step). Following this idea of diffusion in graphs, other schemes based on $[\text{sim}_{\mathcal{M}}]_{i,j}$ are subject to further investigations, especially considering longer diffusion paths in the graph, diffusion taking account continuous item-item similarity (instead of binary similarity) and diffusion decay. The general idea behind diffusion processes in the item-item graph is to make better-considered decisions regarding the selection of negative examples that are *reliably* dissimilar to the item in hand. In this regard, CNS relates to the construction of a hidden graph representing the set of items each user may have observed [Paquet and Koenigstein, 2013]. As such a hidden graph predicts reliable user-item negative interactions (in the form of: user has encountered the item *but* did not take action) it is also possible to infer the item-item similarity (or dissimilarity) from, *e.g.* the set of users that have observed both items in a pair, and make the distinction between users that have interacted with both items (suggesting the items are similar), users that have interacted with exactly one (suggesting the items are dissimilar) or users that have interacted with neither (no clear conclusion can be drawn in that case).

Model near-miss sampling

One of the drawbacks of UNS is that it often samples negatives that turn out to be dissimilar to the image of reference item $\mathbf{y}_i$ in $\mathbb{R}^d$ (recall that $\phi(\mathbf{y}_i)$ and $\phi(\mathbf{y}_j)$ are d-dimensional vectors with $d = 200$). To overcome this effect and provide the network with local information relevant to improve ϕ in the neighborhood of $\mathbf{y}_j$, *model near-miss sampling* (MNS) considers as negative the set of non-positive $\mathbf{y}_j$ whose representations are the closest to that of $\mathbf{y}_i$ with respect to the current language model ϕ:

$$\{\mathbf{y}_j \mid \phi(\mathbf{y}_j) \in K_{NN}NN(\phi(\mathbf{y}_i))\} \subset V(i)$$

with $\kappa NN(\mathbf{x})$ the set of the κ nearest neighbors of $\mathbf{x}$ and K_{NN} a constant that may be chosen as a function of K so that as many negatives are considered as in UNS to facilitate

[9]Defined as the average over the nodes of the graph distance to its farthest neighbor.

a direct comparison. Preliminary experiments show that when a very low value of K_{NN} is chosen, the network succeeds in pushing away the dissimilar items, indicating that the desired anti-collapsing effect of negative examples is fulfilled. However, these are replaced by other dissimilar items and the loss function only very slowly decreases over the training epochs, suggesting that the training should benefit from more "dissimilar" information carried by negative examples.

On the one hand, MNS requires a forward pass over all items and to compute all the distances to $\phi(\mathbf{y}_i)$ before selecting the negative items. Distances computation can typically be done while monitoring the recall score, see Sec. 8.3.4. A partitioning of the items is used instead of a full sort on the distances to reduce the computational cost. On the other hand, the expected benefit is to reduce the number of epochs required to train the Siamese network with a curriculum [Bengio et al., 2009]. As noted by [Loshchilov and Hutter, 2015] for classification networks, oversampling the most challenging labelled examples (i.e., with lowest margin) speeds up learning by focusing the attention on the most discriminant features. Likewise, [Wu et al., 2017] uses *distance-weighted sampling* for training Siamese and triplet networks and optimize a Euclidean distance-based loss: examples are sampled inversely proportionally to the distance with the reference, thus favoring the nearest miss examples.

8.3.4 LaJam Recommendation

Eventually, the fit between the (known) u-th user and a new job of index i is computed by replacing the similarity in Eq. (8.1) by the ϕ_W-based similarity:

$$\widehat{\mathcal{M}}_{u,i} = \sum_{j=1}^{|\mathcal{I}|} \text{sim}_W(i,j)\mathcal{M}_{u,j} \tag{8.23}$$

The recommendations to the u-th user are ordered by decreasing value of $\left[\widehat{\mathcal{M}}_{u,i}\right]_i$.

8.4 MaTJam: Learning Machine Translation Models

This section describes a *single region of interest* (SROI) approach with capabilities analogous to those of LaJam is devised.

The *MAchine Translation-based* Jam (MaTJam) approach assumes that users and items can be mapped onto a single latent space, where the standard similarity accounts for the user's preferences. MaTJam thus paves the way toward full cold-start recommendation in the domain of employment, through using the user's resume to locate the user in the latent space.

8.4.1 Joint learning of two continuous language models

By definition, resumes and job ads are expressed using the same dictionary. Still, as shown in Section 7.2.1, the word distribution in both resume and job ad corpora are too different to achieve the translation using a single embedding from the job ad and the resume spaces into the latent space. Therefore, MaTJam learns two embeddings ϕ and ψ, respectively, from the job and the resume spaces into $\mathbb{R}^d$ (Fig. 8.4).

Let W and W', respectively, denote the weight vectors of the ϕ and ψ neural nets. The elementary loss associated with the u-th user (with initial or reduced representation $\mathbf{x}_u$) and the i-th job ad (with initial or reduced representation $\mathbf{y}_i$) is defined analogously

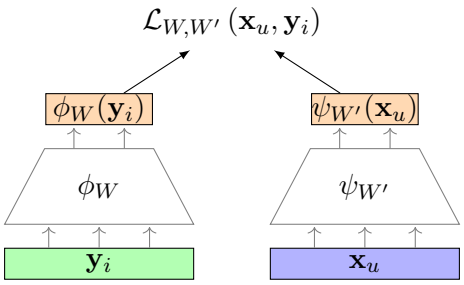

Figure 8.4: MATJAM architecture: two branches with independent weights map the representation $\mathbf{x}_u$ of the u-th user and the representation $\mathbf{y}_i$ of the i-th job into a common representation space.

to Eq. (8.19). Let $\text{sim}_{W,W'}$ denote the cosine similarity in the common latent space, then:

$$\begin{aligned}\mathcal{L}_{W,W'}(\mathbf{x}_u, \mathbf{y}_i) = \delta^+(\mathbf{x}_u, \mathbf{y}_i) \times (1 - \text{sim}_{W,W'}(\mathbf{x}_u, \mathbf{y}_i)) \\ + \delta^-(\mathbf{x}_u, \mathbf{y}_i) \times \left[\text{sim}_{W,W'}(\mathbf{x}_u, \mathbf{y}_i)\right]_+\end{aligned} \tag{8.24}$$

where $\delta^+(\mathbf{x}_u, \mathbf{y}_i)$ is the weight associated to the (u, i) pair.

Like for LAJAM, the two options of binary and continuous weights have been considered.

- For **binary weighting**, $\delta^+(\mathbf{x}_u, \mathbf{y}_j i)$ is 1 iff user u clicked onto job ad i, and 0 otherwise, or equivalently:
$$\delta^+(\mathbf{x}_u, \mathbf{y}_i) = \mathcal{M}_{u,i}$$
$$\delta^-(\mathbf{x}_u, \mathbf{y}_i) = 1 - \delta^+(\mathbf{x}_u, \mathbf{y}_i)$$
- For **continuous weighting**, $\delta^+(\mathbf{x}_u, \mathbf{y}_i)$ is set to $1/\left\|\mathcal{M}_{u,\cdot}\right\|_1$, *i.e.* :
$$\delta^+(\mathbf{x}_u, \mathbf{y}_i) = \overline{\mathcal{M}}_{u,i}$$
This setting is a counterpart of LAJAM positive weighting, which takes its weights according to $[\text{sim}_{\mathcal{M}}] = \overline{\mathcal{M}}^\top \cdot \overline{\mathcal{M}}$. Likewise, $\delta^-(\mathbf{x}_u, \mathbf{y}_i)$ is set to a constant α in order to balance the average weight of positive and negative pairs.

The joint loss used to train ϕ and ψ is the sum over all pairs (u, i), of the elementary loss above:

$$\mathcal{L}(W, W') = \sum_{u=1}^{|\mathcal{U}|} \sum_{i \in V(u)} \mathcal{L}_{W,W'}(\mathbf{x}_u, \mathbf{y}_i) \tag{8.25}$$

Likewise, the tractability of learning is enforced by considering a limited number of job ads for each user. $V(u)$ includes all items selected by the u-th user, and K times as many negative items (not clicked by user u). The selection of the negative items is achieved using a negative sampling scheme similar to LAJAM's uniform negative sampling: K items are selected for each item user u interacted with.[10] The other two options considered for LAJAM – collaborative negative sampling (retaining items selected by users close to the u-th user), and model-based negative sampling (such that $\phi(\mathbf{y}_i)$ is close to $\psi(\mathbf{x}_u)$) – are left for future work.

[10]This sampling is similar to the user-oriented sampling examined in Sec. 8.2.5.

8.4.2 MATJAM Recommendation

Eventually, the similarity learned by MATJAM is directly used for recommendation; the relevance of item $\mathbf{y}_i$ for user $\mathbf{x}_u$ is computed as:

$$\widehat{\mathcal{M}}_{u,i} = \mathrm{sim}_{W,W'}(\mathbf{x}_u, \mathbf{y}_i) \tag{8.26}$$

and recommendations to the u-th user are ranked by decreasing value of $\left[\widehat{\mathcal{M}}_{u,i}\right]_i$.

By construction, LAJAM and MATJAM have different recommendation strategies. LAJAM is a MROI approach: relying on the assumption that user interests are better captured by her history, it operates on the exploitation side (more of the same) as it only recommends items that are similar to the items previously selected by the user. MATJAM is more on the exploration side; it can also exploit the diversity of the user community, based on the fact that user resumes containing a similar set of features discriminating for recommendation are likely to have similar images through ψ, even if they do not share any interacted item, thus enabling serendipitous recommendations.

Naturally, serendipity can be undesirable in the application domain of employment: a user selects a very tiny fraction of the items, and the Qapa items are structured as a 6-diameter small world (or 8-diameter for ABG). The exploratory trend in MATJAM can thus lead to overly general and unfocused recommendations.

Besides, notice that the factorization of matrix $\mathcal{M}$ is a particular case of MATJAM with linear embeddings ϕ_W and $\psi_{W'}$ and normalized representation of user and items (so that the cosine similarity of $\psi_{W'}(\mathbf{x})$ and $\phi_W(\mathbf{y})$ reduces to their dot product).

8.4.3 Further work: dual-LAJAM

Inspired from hybridization of user-oriented and item-oriented neighborhood methods, a dual-language modeling approach, named dual-LAJAM (as it extends LAJAM), may be devised. Dual-LAJAM estimates the relevance of item $\mathbf{y}_i$ for user $\mathbf{x}_u$ as:

$$\widehat{\mathcal{M}}_{u,i} = \left(\sum_{j=1}^{|\mathcal{I}|} \mathrm{sim}_W(i,j)\mathcal{M}_{u,j}\right) \cdot \left(\sum_{v=1}^{|\mathcal{U}|} \mathrm{sim}_{W'}(u,v)\mathcal{M}_{v,i}\right) \tag{8.27}$$

Dual-LAJAM is able to leverage both the user and item textual descriptions as well as the interaction data. As highlighted in Eq. (8.27), the dual-LAJAM recommendation scheme involves separately that of LAJAM (leftmost term) and a user-oriented version of LAJAM (rightmost term). This allows to train separately the item and user embeddings ϕ_W and $\psi_{W'}$ with LAJAM training loss and a user-oriented version thereof, respectively. The development of Dual-LAJAM and its comparison with LAJAM and MATJAM are left for future work.

8.5 Experimental Setting

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

8.5.1 Goals of Experiments

The primary goal of experiments is to determine the recommendation strategy most appropriate to job ad recommendation. On the one hand we have the "more of the same"

strategy at the root of LaJam, learning a similarity on the job ad space supporting the recommendation of the job ads most similar to those selected by the user in the past. On the other hand we have the model-based strategy at the root of MaTJam, bridging the gap between the job ad and the resume spaces through embedding them into a common latent space.

Another goal is to assess the performance of LaJam and MaTJam in warm-start and semi-cold start modes, and to compare them with the state-of-the-art algorithms. The performance indicators are the fraction of relevant items ranked in the top-20 and top-100 recommendations, denoted R@20 and R@100, respectively.

All computational times are measured on an Intel Xeon E7 @ 2.20GHz (16 CPU cores, 4MB cache per core) computer with 128GB of RAM and running under the Ubuntu 14.04 operating system.

8.5.2 Databases and baselines

A Jam approach can best be evaluated on real-world databases. As discussed in Chap. 7 however, such databases are not public, due to privacy constraints. For the sake of a reproducible evaluation, besides the Qapa and ABG databases, the proposed approaches are also evaluated on the public CiteULike benchmark, using the same excerpt and setting used in [Wang and Blei, 2011][11].

The baselines include: i) CTR [Wang and Blei, 2011] (Sec. 6.5); ii) the content neighborhood recommendation using the LSA representation with cosine similarity (Sec. 7.3.1), noted LSA-bl; iii) the content neighborhood recommendation using a LDA representation with dot product similarity, noted LDA-bl; and (in warm-start mode only) iv) the $\mathcal{M}$ neighborhood recommendation, noted $\mathcal{M}$-bl. Emphasis is put on the difference between CTR and LDA-bl: although both are based on the same LDA representation, the former is a SROI approach, optimizing the fit between a user's representation and those of the items she interacted with, while the latter is a MROI approach taking advantage of each of the user's past interaction to find new relevant items.

On CiteULike, LaJam is assessed in semi-cold start mode and compared to CTR ("out-of-matrix" setting in [Wang and Blei, 2011]), LSA-bl and LDA-bl using the same experimental setting as in [Wang and Blei, 2011]. The public implementation of CTR[12] is used, with 250 iterations to extract the LDA representation. Note that MaTJam is not applicable on CiteULike, since no user documents are available.

On Qapa and ABG, LaJam is evaluated in warm-start and semi-cold start modes, and compared to LSA-bl and $\mathcal{M}$-bl; LDA-bl is omitted, as it is dominated by LSA-bl. MaTJam is evaluated in semi-cold start on Qapa only, because only a small fraction of the resumes (less than 10% on the 8,000 users) is provided on ABG.

In warm-start mode, the performance indicators are averaged over 10 runs. Each run considers a collaborative matrix where 10% of the clicks have been removed, subject to removing at most 1 click per user.[13] In semi-cold start mode, the performance indicators are assessed along a standard 10 fold CV procedure (where each fold includes all users and 1/10th of the job ads).

[11] Data are available at `http://www.cs.cmu.edu/~chongw/data/citeulike/`

[12] Available at `http://www.cs.cmu.edu/~chongw/citeulike`

[13] The selection of clicks to be removed is carried out in two steps: firstly, the desired number of rows of $\mathcal{M}$ are sampled uniformly and without replacement; in a second step, for each of this rows, the click to be removed is uniformly sampled among the non-zero values of the row.

8.5.3 Hyper-parameters

On CiteULike, the number of LDA topics is 200 [Wang and Blei, 2011]; for a fair comparison, LSA-bl and LDA-bl also use a 200-dimensions representation. For Qapa and ABG, LSA-bl use a 600-dimensions representation.

Embeddings ϕ and ψ learned by LaJam and MaTJam are implemented as neuronal architectures whose input is the primary tf-idf representation of the job ads and resumes (the input dimension D is 8,000 for CiteULike and 10,000 for Qapa and ABG). Experiments using LSA and LDA reduced representations as input of the neural networks yield poorer results and are omitted.

Two neural architectures are considered:

- the **shallow architecture** is a D-200 fully connected network with a single layer of weights and tanh activation function, initialized so as to emulate the LSA representation (using the top-200 LSA singular vectors); the initial image of a job (a job or a resume in MaTJam) through the shallow architecture closely matches its projection on the top 200 LSA dimensions except for the non-linearity of the tanh activation function.

- the **deep architecture** is a D-1,000-1,000-200 network, with 3 fully connected layers and tanh activation function; this architecture was selected based on previous results [Schmitt et al., 2016]. Weights are initialized so as to emulate the LSA representation on the last layer.

  The rest of the weights are initialized as in [Glorot and Bengio, 2010].

All architectures are implemented in Theano [Theano Development Team, 2016]; the learning rate is optimized using Adam [Kingma and Ba, 2014].

An analysis of sensitivity is conducted w.r.t. every algorithmic option in LaJam and MaTJam, concerning the impact of the binary or continuous weights of the pair losses, and that of the negative sampling modes (UNS, CNS and MNS, Section 8.3.3). Another analysis concerns the impact of using the posting time and geo-location of the job ads for the recommendation.

The impact of the posting time and geolocation attached to each job ad for Qapa and ABG is investigated by comparing four representations: i) no extra information (words only); ii) geolocation (plus words); iii) posting time (plus words); iv) all: geolocation, posting time and words.

For a fair comparison, LSA-bl and LDA-bl use an enriched similarity measure accounting for this extra information:

$$\text{sim}(i,j) \to \text{sim}(i,j) - \lambda_1 d_g(i,j) - \lambda_2 d_t(i,j), \tag{8.28}$$

where $d_g(i,j)$ (respectively $d_t(i,j)$) is the Euclidean distance between the geolocations (resp. the posting time) of the i-th and j-th job ads. The λ weights are selected by hierarchical grid search to optimize the recall on the training set. In a first round, values 10^a are considered with $a \in -2, ..., 2$ and the relevant order of magnitude a^* is identified; in a second round, 10 values logarithmically spaced in $[10^{a^*-1}, 10^{a^*+1}]$ are considered and λ is set to the optimal one.

8.6 Experimental Validation

This section reports on the experimental validation of LaJam and MaTJam. All reported computational times are measured on 16 CPU cores (Intel Xeon E7 @ 2.20GHz, 4MB cache

per core) computer with 128GB of RAM, running under the Ubuntu 14.04 operating system.

8.6.1 Comparative assessment on CiteULike

Table 8.1 reports the Recall@20 and Recall@100 on the CiteULike dataset in semi cold-start mode [Wang and Blei, 2011]. For this experiment, LaJam is trained using the uniform negative sampling scheme (UNS). Surprisingly, CTR only slightly improves on LDA-bl; even more surprising is the fact that LSA-bl significantly dominates both LDA-bl and CTR. LaJam yields comparable results as LSA-bl for R@20, and dominates all other approaches for R@100.

	R@20	R@100	Training time
LSA-bl	**.332** (**.001**)	.631 (.003)	2min
LDA-bl	.247 (.005)	.584 (.005)	2h
CTR	.271 (.001)	.587 (.001)	2h
shallow LaJam	**.327** (**.004**)	**.652** (**.005**)	10min
deep LaJam	.279 (.005)	.606 (.006)	20min

Table 8.1: CiteULike: Recall@20 and Recall@100 semi cold-start performances of LSA-bl, LDA-bl, CTR and LaJam learned with UNS (average and standard deviation using 5-fold cross-validation). For LSA-bl and LDA-bl, the training time concerns the unsupervised learning of the LSA (resp. LDA) representation. For each measure, the boldfacing indicates the statistically significantly best performances (based on Welch's t-test with significance level 0.01).

Interpretation of the results

A tentative interpretation of these results, focusing chiefly on the fact that LSA-bl dominates LDA-bl, goes as follows. Let Θ_u denote the sum, over all documents selected by the u-th user, of their LDA representation, with θ_j the LDA representation of the j-th document. As the similarity between LDA representations of documents is the dot product, Eq. (8.1) can be rewritten as:

$$\begin{aligned} \forall u \in \mathcal{U}, \widehat{\mathcal{M}}_{u,i}^{LDA-bl} &= \sum_{j \in \mathcal{I}} \langle \theta_i, \theta_j \rangle \, \mathcal{M}_{u,j} \\ &= \left\langle \theta_i, \sum_{j \in \mathcal{I}} \theta_j \mathcal{M}_{u,j} \right\rangle \\ &= \langle \theta_i, \Theta_u \rangle \end{aligned} \tag{8.29}$$

LDA-bl recommends to the u-th user the documents with LDA representation θ best aligned with Θ_u (with highest dot product $\langle \theta, \Theta_u \rangle$.

On the other hand, LSA-bl recommends to the u-th user the documents with LSA representation $\mathbf{y}_i$ and highest sum of $s_{\cos}(\mathbf{y}_i, \mathbf{y}_j)$ where $\mathbf{y}_i$ ranges over the LSA representation of the documents selected by the u-th user. Without loss of generality (regarding the LSA baseline recommendation), let us assume that all LSA representations have L_2 norm set to 1. Cosine similarity $s_{\cos}$ then reduces to the dot product. Let Λ_u denote the sum of the $\mathbf{y}_i$; analogously to above, it follows:

$$\forall u \in \mathcal{U}, \widehat{\mathcal{M}}_{u,i}^{LSA-bl} = \langle \mathbf{y}_i, \Lambda_u \rangle \tag{8.30}$$

The LSA baseline recommends to the u-th user the documents with LSA representation $\mathbf{y}$ best aligned with Λ_u (with highest dot product $\langle \mathbf{y}_i, \Lambda_u \rangle$).

In both cases, a "center of attraction" for the u-th user is defined from the data (Θ_u for LDA and Λ_u for LSA) and the recommended documents are those nearest to this center in L_2 norm (using a dot product). An important difference, beyond how the representations are defined, is that the document representations lie on the L_1 unit hypersphere in the LDA case, and on the L_2 unit hypersphere in the LSA case. After the celebrated discussion about the comparative effects of L_2 *vs* L_1 regularization [Ng, 2004], given a convex objective (here the distance to the center of attraction), the isolines thereof intersect an L_1 ball in a more sparse *and more instable* manner as for an L_2 ball. Said otherwise, small differences in the document representation have more impact in the LDA case than in the LSA case.

CTR improves on LDA-bl: it operates on the same search space and adjusts the "center of attraction" for the u-th user, adjusting the user representation $\mathbf{u}$ to maximize the fit with the document representations (set to their LDA representation plus a corrective term). It nevertheless remains close to LDA-bl.

Likewise, LaJam with a shallow architecture is close to LSA-bl due to its initialization (recall that the network emulates the 200-dimensional LSA representation at its initialization). The learning process trades off a slight loss on R@20 in counterpart for a significant gain on R@100.

With a deep architecture however, LaJam is dominated by both shallow-LaJam and LSA-bl − though it still improves on CTR in a statistically significant manner (according to Welch's t-test with significance level 0.01).

Sensitivity analysis

The properties of the learned metrics are partially controlled by:

- weighting modes – reflecting whether all job ads must be considered equal, or some are more important than others (Sec. 8.3.2);
- the negative sampling modes, determining how the job ads expand in the latent space;
- the hyper-parameter K, controlling the strength of the expansion through the balance of the positive and negative elementary losses.

The sensitivity analysis is conducted on CiteULike in semi cold-start mode using the shallow LaJam architecture (Fig. 8.5).

Complementary results[14] show that the binary weighting is consistently dominated by the continuous weighting mode, all the more so as the K value is low.

The impact of the negative sampling mode together with the K value is illustrated on Fig. 8.5. The recall curves for UNS and CNS are parallel to each other for the different K values, and they steadily improve as K increases. This is explained from the small graph diameter of the CiteULike item-item graph, which is 2.3. Circa 80% negative items (sampled by UNS) also are at distance 2 of the reference item (sampled by CNS): UNS and CNS thus implement quite similar sampling distributions. The recall performance improvement is very significant when K increases from 1 to 3, and more moderate when K increases from 3 to 10, as the training eventually leverages all available information.

[14]For $K = 1, 3, 10$, the binary weighting mode with UNS yields respectively 0.357, 0.512, 0.636 R@100, compared to 0.501, 0.600, 0.654 for the R@100 with continuous weighting mode with UNS.

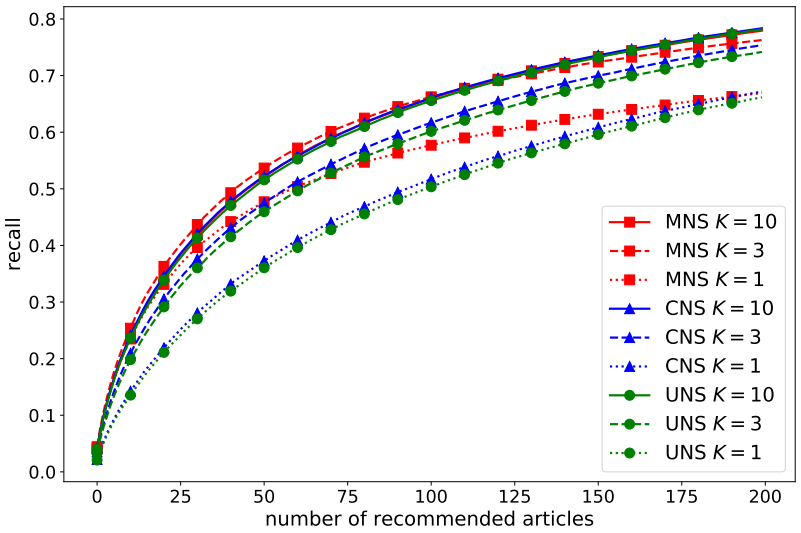

Figure 8.5: Semi cold-start recall on CiteULike: impact of the negative sampling schemes (circle: UNS; triangle: CNS; square: MNS) and of the ratio K of negative *vs* positive pairs (dotted: $K = 1$; dashed: $K = 3$; plain: $K = 10$). The standard deviation is below 1% and omitted for readability.

This is confirmed as all UNS, CNS and MNS yield indistinguishable recall curves for $K = 10$.

For the MNS option, the best performance is obtained for $K = 3$ at the beginning of the recall curve ($T \approx 50$), dominating all other schemes. The recall curve for $K = 10$ catches up and outperforms the MNS-$K = 3$ recall curve later on ($T \geq 100$). The recall improvement on the MNS curves is less steady than for UNS and CNS, suggesting that the MNS option might trade-off better top recommendations for a lower R@200.

8.6.2 Warm-start performances on Qapa and ABG

Fig. 8.6 displays the recall curves in warm-start mode of LSA-bl, $\mathcal{M}$-bl and LaJam (shallow architecture) on Qapa and ABG.

On Qapa, shallow-LaJam is slightly dominated by $\mathcal{M}$-bl at the beginning of the recall curve ($T < 100$); it catches up and significantly dominates the baselines for $T > 150$. From the application perspective, the main improvement is that 20% more users are now more decently served: while their first relevant recommendation has rank higher than 10,000 for LSA-bl and $\mathcal{M}$-bl, LaJam now provides a relevant recommendation in the top 1,000. The running time of LaJam is ca. 7 hours (resp. 50 minutes) for training and 4 minutes (resp. 12') for recommendation for all users for Qapa (resp. ABG).

On ABG, LaJam dominates both baselines for $T \geq 100$. It manages to reach comparable performances in terms of R@900 as on Qapa(Fig. 7.4); note, however, that the recommendation is among 10,400 job ads or ABG (as opposed to 56,000 job ads for Qapa).

8.6.3 Semi cold-start recommendation on Qapa and ABG

Table 8.2 reports the performance of deep and shallow LaJam on the Qapa and ABG databases, comparatively to LSA-bl. LDA-bl is omitted due to its lower performance[15].

[15]On Qapa, LDA-bl yields 0.26 for Recall@20 and 0.62 for Recall@100. On ABG, it yields 0.15 for Recall@20 and 0.41 for Recall@100 (words only).

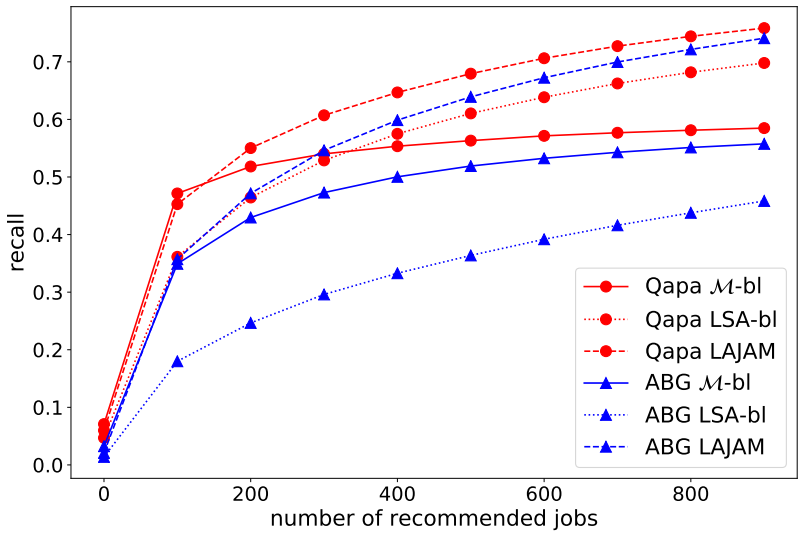

Figure 8.6: Qapa and ABG: Recall curves in warm-start mode of LSA-bl (dotted line), $\mathcal{M}$-bl (plain line) and shallow LaJam (dashed line). R@T is the percentage of recovered clicks among the top T recommendations (out of 56,000 job ads for Qapa, and 10,000 job ads for ABG). All recall curves are averaged on 10 runs. Standard deviation is under 0.003 for Qapa and under 0.006 for ABG.

Four settings are considered, where the job ad representation includes:

- the words only;
- the words and the geolocation;
- the words and the posting time;
- the words, the geolocation and the posting time.

Importance of additional features A first remark regards the utmost importance of the geolocation on Qapa, and of the posting time for ABG. This confirms the difference of the low *vs* highly qualified job sectors and the importance of letting the system learn how to take this information into account.

Specifically, on the Qapa data, the geolocation information yields an improvement of circa 13% for LSA-bl and 4% for LaJam. At the same time, the geolocation information does not yield any extra performance on the ABG dataset; a tentative interpretation for this fact is that PhDs and post-docs are used to geographic mobility.

On the other hand, the time information yields an improvement of circa 30% on the ABG data (spanning over 5 years) and only has moderate effect on the Qapa dataset (spanning over 3 months).

Note that LSA-bl benefits from a considerable advantage, exploiting the geolocation and posting time distances with optimal weights (Eq. (8.28)). In contrast, LaJam learns from scratch how to best exploit the geolocation and time information, expressed via 3 features among 10,000+ ones (the word features).

Nevertheless, this experiment shows how easily additional features can be fed to LaJam with no design change and negligible extra computational cost.

Qapa dataset					
	–	geoloc	time	geo-time	CPU
LSA-bl R@20	.404 (.007)	**.636** (**.003**)	.439 (.007)	**.656** (**.003**)	5min
LSA-bl R@100	.694 (.005)	**.829** (**.003**)	.713 (.005)	**.835** (**.002**)	
shallow LaJam R@20	**.495** (**.006**)	.549 (.006)	**.523** (**.005**)	.558 (.006)	200min
shallow LaJam R@100	**.743** (**.007**)	.784 (.005)	**.764** (**.006**)	.790 (.005)	
shallow MaTJam R@20	.374 (.007)	-	-	-	150min
shallow MaTJam R@100	.714 (.007)	-	-	-	
deep LaJam R@20	.475 (.007)	.499 (.006)	.483 (.006)	.503 (.007)	400min
deep LaJam R@100	.725 (.007)	.748 (.007)	.731 (.006)	.752 (.006)	
ABG dataset					
	–	geoloc	time	geo-time	CPU
LSA-bl R@20	**.254** (**.010**)	**.258** (**.008**)	**.574** (**.008**)	**.579** (**.009**)	5min
LSA-bl R@100	.522 (.012)	.528 (.013)	**.814** (**.007**)	**.817** (**.007**)	
shallow LaJam R@20	.226 (.008)	**.260** (**.009**)	.39 (.009)	.392 (.011)	25min
shallow LaJam R@100	**.544** (**.008**)	**.599** (**.006**)	.761 (.009)	.755 (.008)	
deep LaJam R@20	.200 (.008)	.199 (.008)	.312 (.008)	.336 (.009)	75min
deep LaJam R@100	.493 (.010)	.492 (.011)	.705 (.007)	.717 (.008)	

Table 8.2: Semi cold-start setting: shallow- and deep-LaJam (with binary similarity, UNS, $K = 10$) and LSA-bl Recall@20 and Recall@100 performances on Qapa (5,600 job ads) and ABG (1,040 job ads) (10-fold cross-validation). The training time is reported in the rightmost column; the recommendation time (for all users) on Qapa is 27s for LSA-bl and 46s for LaJam; on ABG, it is 6s for both LSA-bl and LaJam. For each pair (job ad representation setting, measure), the boldfacing indicates the statistically significantly best performances (according to Welch's t-test with significance level 0.01).

Shallow *vs* Deep architectures Deep LaJam is always dominated by shallow LaJam. This is blamed on the lack of regularization for the deep architecture, already noted on the CiteULike problem.

For the word-only setting, shallow LaJam very significantly outperforms the baseline on the Qapa database: by 9% regarding the Recall@20 and 5% regarding the Recall@100. On the ABG database, shallow LaJam is dominated by the baseline in the word-only setting for the Recall@20 performance indicator, and slightly though significantly dominates the baseline for the Recall@100 indicator. Shallow LaJam likewise dominates the baseline in the word+geolocation setting.

Deep LaJam ouperforms LSA-bl on the Qapa dataset in the word only and word + posting time settings.. Though geolocation is of utmost importance on this dataset, deep LaJam is only slightly affected when it is provided.

SROI *vs* MROI approach MaTJam is only applicable in the word only setting. For privacy reasons, user geolocation is not provided as part of the Qapa dataset. Besides, the user profile creation time is not relevant in the context where users periodically visit the platform; therefore there is no time information attached to the users (user account activity, if tracked, is not part of the datasets).

MaTJam yields much poorer results than LaJam, with a decrease of circa 10% for R@20 and 3% for R@100 on Qapa. Still, it outperforms the LSA-bl for the Recall@100 indicator, indicating that it globally positions users and items representations well w.r.t. each other. As stated earlier, MaTJam is not applicable to ABG, as 70% of the resumes are missing.

These results confirm a main merit of the neighborhood-based MROI LaJam approach approach over the SROI approach. As noted by [Koren, 2008], neighborhood methods can capture relevant localized relationships from very few interactions, as in the case of sparse databases like Qapa and ABG. On the contrary, SROI aims at a global co-representation of users and items in a same latent space, aiming to account for the specifics of the different job sectors. Finding such a unified latent representation proves harder than capturing local similarities.

8.6.4 Sensitivity analysis

We also investigate the sensitivity analysis of negative sampling on the Qapa and ABG datasets (Fig. 8.7), for shallow LaJam in the word only setting. Due to small differences in preprocessing, these results are not directly comparable with those presented in Tab. 8.2.

Insights into the negative sampling schemes differ widely from those for CiteULike (Sec. 8.6.1). A first remark regards the significant differences among the considered configurations, with more than 20% difference in terms of recall@100. This difference was expected, as the number of positive neighbors (hence the number of sampled negative neighbors) is much smaller in Qapa and in ABG than in CiteULike. Option MNS thus only considers a few negative neighbors, localized in LaJam output space; option CNS only samples for each item among the few items that are reachable in two steps in the graph induced by $[\mathrm{sim}_{\mathcal{M}}]_{i,j}$ – this set consists of about 7.5% of the total number of jobs for Qapa; in comparison for CiteULike it comprises more than 80% of the articles.

MNS results in very poor recall@T and is widely dominated by most other configurations for $T \geq 20$. Recalling that all sampling strategies provide LaJam with negative examples *proportionally to* the number of positive items pairs, the MNS strategy provides LaJam much fewer examples for Qapa and for ABG than for CiteULike. This suspicion is

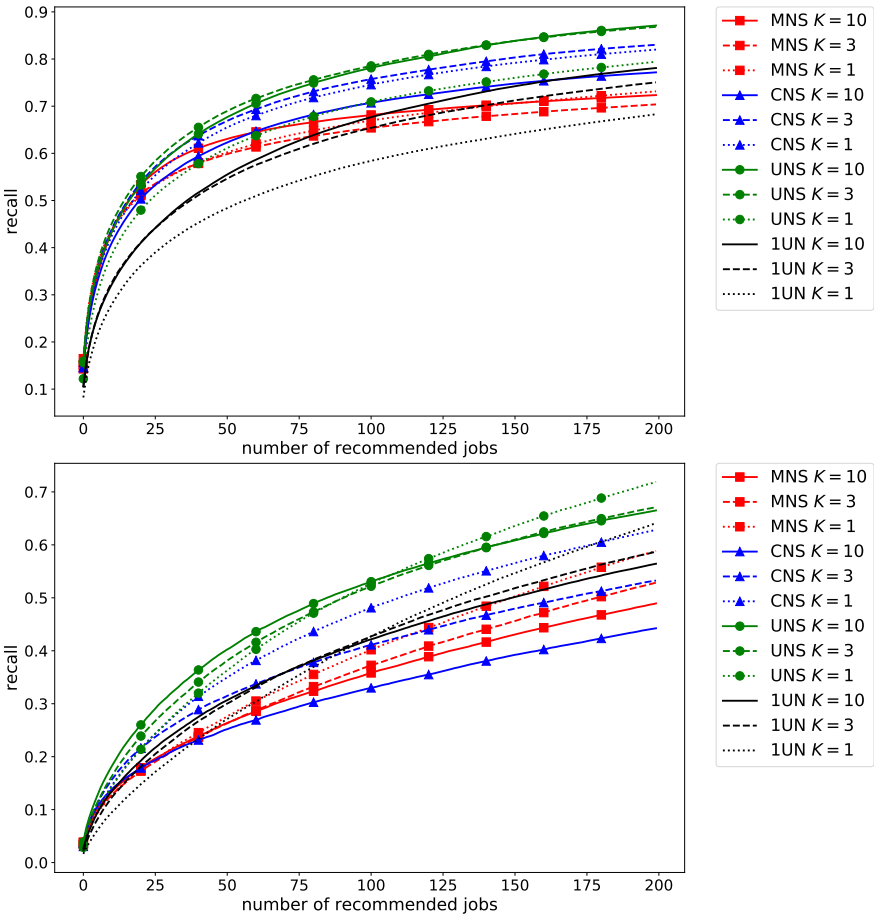

Figure 8.7: Qapa (above) and ABG (below) datasets: impact of the negative sampling schemes (circle: UNS; triangle: CNS; square: MNS; all with continuous weighting) and of the ratio K of negative *vs* positive pairs (dotted: $K = 1$; dashed: $K = 3$; plain: $K = 10$). As a comparison, uniform sampling with the binary similarity is depicted as "1UN" (no markers).

confirmed on Qapa by the slightly higher performance of MNS for $K = 10$ over $K \in \{1, 3\}$ (when $T \leq 20$). A natural remedy to this lack of negative neighbors consists in increasing K at the expense of higher computational cost. For ABG, however, increasing K yields poorer performances, suggesting considering more dissimilar pairs puts unneeded constraints on training. This phenomenon is blamed on the small size of the dataset.

The comparison between the UNS and CNS strategies yields however unexpected results. For a given K, UNS is always superior to CNS, for both Qapa and CiteULike databases. This suggests that the restriction of potential negative neighbors to those obtained based on a two-step diffusion from the reference items on the $[\text{sim}_{\mathcal{M}}]_{i,j}$-induced graph is of limited interest. Following this direction, a more advanced diffusion process will be considered for further work.

Quite unexpectedly, for CNS on Qapa and for all three negative sampling strategies on ABG, smaller K yields better performance. To the best of our knowledge, this is the first time this kind of results has been reported; further studies are needed to understand why ABG really does not leverage the "negative" information included in dissimilar pairs.[16] For CNS, we propose the following explanation: with more negative neighbors sampled, the set of potential negative neighbors (according to this method) is overfitted, *i.e.* LaJam has learned all these items pairs. On the contrary, on Qapa there are about 20 times more potential negative item pairs than positive pairs; hence, a negative pair is sampled circa once every 2 epochs for $K = 10$.

As expected, the binary similarity is outperformed by the continuous weighting one (both with the UNS scheme) in all cases. This confirms the relevance of considering different levels of similarity.

8.7 Discussion

Our empirical results provide some answers to the scientific questions regarding the specifics of the domain of employment in terms of recommendation problem.

The main lesson learned regards the comparison between the SROI MatJam approach and the MROI LaJam approach. In brief, MatJam is always dominated by LaJam. The former approach eventually catches up and might dominate LSA-bl later on the recall curve (*e.g.* for the semi-cold start R@100 on Qapa). However, it is consistently dominated by LaJam, particularly so at the beginning of the recall curve. This lack of performance is blamed on two main reasons, as follows.

It follows from the exploratory data analysis (Chap. 7) that regarding the recommendation goal, user-item interactions provide better information than the mere item descriptions. LaJam uses the interaction data twice: firstly, these are used to define the item similarity and train the neural net; secondly, the interaction matrix is used through convolution while predicting the recommendation values $\widehat{\mathcal{M}}$. On the other hand, MatJam only uses the interaction information once, to train the neural net; it thereafter only relies on the continuous language embedding, and on the assumption that resumes contain enough information to discriminate among relevant/irrelevant job ads. However, this assumption is questionable in light of our data analysis (Sec. 7.2).

A second tentative explanation for the lack of performance of MatJam relates to the diversity of the job ads selected by a same user, already mentioned in Chap. 7. The diversity of the user's selected items adversely affects the modeling of the user in the same latent space as the items: actually, the user latent representation would intuitively be

[16] It is reminded that the main use of dissimilar pairs – counterbalancing the attractive loss – is unquestionable.

situated "in the middle" of the latent representation of the items she interacted with. In other words, the MaTJam single latent space favors a transitive similarity structure: all items relevant to a user should be similar to that user in the latent space, and therefore similar to each other. Indeed this transitivity affects both LaJam and MaTJam. However the loss function used to train LaJam actively counter-acts this transitivity through pushing away dissimilar items. In contrast, the MaTJam loss does not involve any repulsive term among items. The trend toward a transitive similarity is resisted as one uses the cosine similarity in the latent space which offers the possibility for multiple items to be "relatively similar" to a same user while being dissimilar pairwise. However, all else being equal, more diversity among user interactions pushes dissimilar items to cluster together in the latent space, adversely affecting its topology.

Application-wise, it is plausible that a user might feel qualified for more jobs than the ones she already occupied and mentioned in her resume, all the more so as the mainstream resume guidelines impose to stick to a one-page resume. A closer look at the data actually reveals that some users might apply for positions which are considered to be very different. For instance, a user might apply for both positions of clothe sale employee and hostess, although these positions are far from each other according to of the official job ROME ontology.[17] In the domain of employment, a user should thus rather be viewed as a set of profiles.

Note that neither LaJam nor LSA-bl suffer from this diversity phenomenon. They mostly focus on the similarity between an item and items previously selected by the user, and thus seamlessly accommodate user selection diversity.

More generally, the experiments suggest that one should distinguish recommendation problems where the user's interests are *diverse* w.r.t. the item space, from those where the user interests correspond to a single region in item space. Echoing Chap. 6, let us refer to these two categories of recommendation problems as *Multiple Regions of Interest*-recommendation (MROI) problems, and *single region of interest* (SROI) problems, respectively. SROI problems are exemplified by the Netflix challenge: while a user indeed has diverse "interests" (*e.g.* being a mixture of comedy, war, gore, etc., fan), an item offers the same diversity (being also modeled as a mixture of comedy, war, gore, etc, movies). Accordingly, Netflix users and items can consistently live in the same model space. MROI problems are illustrated by, *e.g.* , the e-commerce recommendation problems where a single e-commerce account is used by an entire family (see [Verstrepen, 2015] and references therein), with diverse interests. In the domain of employment, a single user seemingly has multiple regions of interest.

8.8 Perspectives on continuous language modeling for recommendation

The use of natural language processing techniques is essential for automatic recommendation of jobs to applicant in a cold-start setting. Two neural network language models that incorporate interaction data in their training have been proposed in chapter.

LaJam embeds job ads in a latent representation space that accounts for the similarity between them, as perceived by the applicants. Emphasis has been put into LaJam training mechanisms in order to achieve desirable behavioral through training. The merits of LaJam have been satisfactorily demonstrated and compared with respect to the state-of-

[17] `http://www.pole-emploi.fr/candidat/le-code-rome-et-les-fiches-metiers-@/article.jspz?id=60702`.

art CTR, and two baselines (based on the LSA and the LDA representation of the items, respectively) on the public CiteULike benchmark. The strength of the LSA-baseline in particular is manifested as it ranked 15th out of 100 participants on the RecSys 2017 Challenge[18]. In the domain of employment, LaJam has demonstrated its versatility w.r.t. to different job sectors, specifically the low-paid job sectors (rarely considered in the literature) and the highly-qualified job sector of PhDs in industry. The ability of LaJam to seamlessly take into account extra information, such as job ad geolocations and posting times, has also been demonstrated.

A first direction for future research entails the architecture and low-level neural network optimization mechanisms. Recent developments in neural network for NLP have mainly focused on recurrent architectures to keep information about the word alignment to *understand* the documents. In contrast, the current approach is purely based on word weighting as its inputs. The integration of more complex architecture for NL analysis would presumably be beneficial for LaJam. Although it could be easily implemented as document preprocessing prior to using LaJam, further research should investigate the way to learn such a preprocessing scheme so that features *relevant* for Jam are fed to LaJam, and integrate this preprocessing as the first layers in a deep architecture. Another important observation is that the deep architecture of LaJam is already seemingly subject to overfitting; this rises a major concern regarding the amount of data required for this architecture extension. Alternatively, carefully designed mechanisms to prevent overfitting must be implemented, taking inspiration from, *e.g.* dropout [Srivastava et al., 2014], regularization (these first two mechanisms did not significantly improve deep LaJam) or limiting the number of free parameters through factorization [Salakhutdinov et al., 2007].

The choice of a relevant similarity between pairs of items is crucial to guiding the training process of LaJam. The future of Siamese networks in the recommendation context is undoubtedly linked to closer attention to providing the network with a meaningful and ideally denoised similarity. The use of a diffusion process on the bipartite user-item graph, in the spirit of Personalized Page Rank [Page et al., 1999], could be used to define a better loss function to train the LaJam item similarity.

The second approach, dubbed MaTJam, represents both the resumes and jobs in a shared latent representation space. Although this approach is by far preferable to a simple word matching approach, it is limited – like the well-known factorization approaches – by the sparsity of the data and the diversity of the users interactions.

A main direction of improvement is to fully address the user diversity of interests, and to reconsider accordingly the MaTJam approach, taking inspiration from [Verstrepen and Goethals, 2015]. In particular, a primary clustering of the items can be used to identify the multiple regions of interest of a given user and may help MaTJam to construct *several* embeddings of the same resume, bridging the gap between latent factor models and MROI approaches. As MaTJam is designed to be addressing the full cold start problem in the future, any overfitting in the user mapping training will be exacerbated in this setting, making overfitting prevention another crucial ingredient of MaTJam training.

Application-wise, our most important direction of research is to build a professional training recommender on the top of LaJam. Exploiting LaJam in "what if" mode, one should be able to identify the skills that a user should most preferably acquire in order to maximize his job opportunities in a given context.

[18] https://recsys.xing.com/leaders, participant *Taoist*.

Chapter 9

Conclusions on the JAM problem and perspectives

The JAM problem has been addressed through a number of different perspectives ranging from NLP to collaborative filtering. As one of the most promising options, a combination of natural language modeling and standard recommendation approaches have been developed in this thesis. The data-driven approach we followed is summarized in this chapter, with the main conclusions reiterated.

Projections from the current state of our understanding are then exposed; they entail enhancement of the approaches developed in this work as well as further perspectives on real-world recommendation applications.

9.1 Understanding the JAM problems through data analysis

As mentioned earlier, preprocessing steps are key to deriving the maximum possible information from raw data. Inadequate preprocessing might undermine the performances of the most powerful recommender system as well as alter any conclusion drawn from the data. Working with mostly unprocessed data and being able to control these steps represents a chance as well as a heavy responsibility. The lessons exposed thereafter may result from these early choices and must be confirmed on larger subsets of, *e.g.* the Qapa database.

The first striking result regards the poorly informative resumes from the automatic recommendation perspective and the misalignment with the interaction data (applications to job positions). This phenomenon dooms approaches heavily relying on valuable textual information. Still, resumes provide some useful clues on the candidate that may integrated into LAJAM as, *e.g.* a screening process.

As a consequence of misalignment between users' description and behavior, learning a latent factor profile of the user (like MATJAM does) matching the user's textual description is tricky, since the resumes may not account for a significant fraction of the user's interactions. The parametric transform of a resume to a user latent representation cannot properly leverage patterns in the resumes, making it moderately useful for user-oriented semi cold-start.

Another reason for MATJAM's lower performances is that recommendations are computed from similarity with a single user latent representation. This representation is encouraged to be similar to all those of the items the user interacted with. It follows that a new position which is moderately similar to all of the jobs the user applied to is favorably positioned in the recommendation list, even though none of the desires the user expressed

in her past interactions is fully fulfilled. In comparison, another position closely matching only a subset of these desires would be rejected further in the recommendations.

This effect is introduced in Chap. 6 under the terms Single Region Of Interest (SROI) and Multiple Region Of Interest (MROI), and preliminary experiments in Chap. 7 have demonstrated that MROI approaches were better suited for the JAM context. A sensible argument on the difference in performances between latent factor approaches (which generally belong to the SROI category) and neighborhood approaches (MROI) is given in [Koren, 2008]. The main strength of neighborhood methods is to efficiently leverage very localized and specific similarities, *e.g.* recommending more items with the exact same characteristics as at least one item the user interacted with; latent factor approaches however are best to capture general patterns in the interaction data.

This argument possibly explains why MATJAM fails to provide accurate recommendations on very sparse databases, where the weak signal in the past user interactions is not sufficient for reconstructing reliably her profile in a whole. A major claim of this thesis is that there exists another underlying phenomenon accounting for the observed diversity of a user behavior: a user is not a mixture of interests but should rather be viewed as a collection of profiles, and any of these "personalities" is possibly responsible alone for an application to a job position. As already mentioned, this problem has been studied in the literature under *multiple instance learning* [Dietterich et al., 1997], and more closely in a recommendation context as the *recommendation for shared account* problem [Verstrepen and Goethals, 2015], in which the account is "split" into the number of users sharing the account (which is to be inferred). It is yet to be investigated whether this last problem is equivalent to MROI recommendation in JAM.

9.2 Perspectives

This section mentions and motivates a few extensions to LAJAM and MATJAM that look promising. Further ideas to investigate regarding the specific requirements of the JAM recommendation problem are presented thereafter, keeping in mind that the relevance of these aspects may be dataset-specific and is yet to be demonstrated for a larger span of applications.

9.2.1 Enhancing the language models

Non-linear formulation of similarity learning Motivated by the need of a more realistic similarity for analysis purposes, a further extension considers an alternative to the linear similarity learning formulation at the core of LAJAM training. A close problem is that of unsupervised dimensionality reduction while preserving local similarity, which has been tackled by, *e.g.* t-SNE [van der Maaten and Hinton, 2008] through the alignment of the distribution of similarities (more precisely, a function of the distance) in the original and in the reduced space.

While the original method requires knowing the distances between each pair of points and only computes the reduced representations, the method is extended in [van der Maaten, 2009] to parametric embeddings implemented as neural networks. With slight changes, it is possible to make the connection with similarity learning: (parametric) embeddings are learned from the jobs description space to the reduced space so that the similarity in the reduced representation is aligned with $\text{sim}_{\mathcal{M}}$ (instead of the similarity in the original jobs description space).

The main interest of t-SNE formulation lies in how item-item pairs incur the training

cost. Let us consider two distinct positive pairs of respective similarity 1 and 0.5. In LaJam, they incur the same cost in the attractive term of the loss (with possibly different weights) if their reduced representations have the same similarity. As a consequence, the network tends to maximally bring together either pair, independently of any other pair similarity in the reduced space; the only effect of pairs on each others resides in the trade-off – controlled by positive and negative pairs weights – to "maximally satisfy" all pairs. One of the drawbacks of this phenomenon is that, as there are fewer constraints on jobs with few interactions, they are likely to be more easily satisfied, bringing together pairs while there is no strong clue that they are very similar. Using a cost function such as that of the t-SNE approach instead links the cost of all pairs in the reduced space altogether, mitigating the risk of bringing together moderately similar items because it would affect all other item pairs costs. Another advantage of t-SNE is its capability to focus on reproducing pairs with high similarity while items with a lower similarity have little effect on the loss in comparison to vanilla LaJam (thanks to the use of a heavy-tail probability distribution to model distances).

A related approach better suited for ranking is based on presenting preference triplets during training [van der Maaten and Weinberger, 2012]. Expanding this method with parametric embedding is a natural following-up of this thesis.

Diffusion process based similarity As already mentioned, the interpretation of non-observed user-item pairs is key in the OCCF context. Negative sampling schemes are investigated through the angle of which dissimilar pairs of items (for LaJam) should be shown to the language models so that properties relevant to recommendation are learned. Another aspect suggested in [Pan et al., 2008] is to favor sampling data from the interaction matrix that are reliably negative. Such an idea is implemented in the work of [Paquet and Koenigstein, 2013] as an unobserved graph whose edge links users to items they might to have encountered; the observed graph – induced by the interaction matrix – is a subgraph thereof. A first option considers sampling negative user-item pairs from the difference between the edges sets of these two graphs to provide MaTJam with relevant negative examples.

An alternative, taking inspiration from diffusion processes in graphs, is proposed for further investigation. While diffusion and random walk on graphs are common techniques for information retrieval [Page et al., 1999] or recommendation [Fouss et al., 2007; Gori and Pucci, 2007], by construction they cannot deal with cold-start, *i.e.* a situation where a node (user or item) with no connection yet is added to the graph. However, diffusion processes are promising candidates to process the bipartite interaction graph and from which to derive an educated similarity for MaTJam to mimic. Such an approach would be favorably coupled with bipartite graph projection [Zhou et al., 2007] to provide LaJam with an accurate item-item similarity, extending the weighting schemes introduced in this work (Sec. 8.3.2).

9.2.2 One-to-many matching

A most exciting extension of MaTJam regards changing its perspective from an SROI to an MROI approach. The ideas presented thereafter are also applicable to a wider range of SROI systems, typically all those based on neural architectures.

The basic principle consists in extracting *several views* of a user from a *single* latent factor representation. Let us assume that the item latent representation $\mathbf{y}$ lies in $\mathbb{R}^d$. MaTJam is trained to learn higher-dimensional representations of users $\mathbf{x}$ (*e.g.*, in $\mathbb{R}^{2d}$) and a set of projections $\varphi_1, ..., \varphi_q$ from $\mathbb{R}^{2d}$ to $\mathbb{R}^d$. Recommendations are then produced

considering the best match between $\varphi_1(\mathbf{x}), ..., \varphi_q(\mathbf{x})$ and $\mathbf{y}$. During training, the loss would only affect the user latent representation $\mathbf{x}$ through its best-matching view. This setting allows $\mathbf{x}$ to encode both *Paris* and *London* locations of interest. The "choice" of the most relevant view of $\mathbf{x}$ can be simply implemented in MATJAM through a *softmax* transform after the computation of all of the user's views similarities with the item latent representation. Other algorithms related to *competitive learning* [Ahalt et al., 1990] are to be investigated. Interestingly, the view most similar to the item representation is not only useful for positive examples but also for negative ones, as it is the most likely explanation for a recommendation the negative sampling has to avoid.

It is desirable that all of the user's views share a fraction of the latent representation, *e.g.* diplomas or experience level. As a consequence, any positive example for this user may contribute to learning the high-dimensional user latent representation. In contrast, [Verstrepen and Goethals, 2015] tackle the recommendation for shared account problem by dividing the set of a user's interactions into the (possibly overlapping) subsets that best explain the data. This is done using an inner algorithm that would not allow integration into a neural network architecture.

A long-term perspective regards how many of such q views to build for each user and how different they should be. If similar views may be manually merged after training, it would be desirable to automatically produce views that capture different – but possibly overlapping – aspects of the user representation, and on a per-user basis. It is worth mentioning that a similar problem arises in statistical machine translation in the approach called "encoder-decoder" [Cho et al., 2014; Sutskever et al., 2014]. Given a sentence encoded in a fixed-length vector representation by the encoder, the decoder part should output a sequence of words of variable length (*e.g.* in the target language for translation). Here, the task of the decoder would consists in producing a sequence of user's views. A sound option thus considers using recurrent neural networks, iteratively forming views based on the user particular latent representation and on the different views already produced.

Part III

General conclusion

Chapter 10

Discussion and conclusion

Two different aspects of cold-start recommendation have been explored in this thesis, applied to two different domains, algorithm portfolios and job-applicant matching. However, a lot remains to be done, be it at the fundamental level of recommendation systems facing the cold-start issue, or in both application domains. Further research directions include for instance considering the collection of recommendations as a diversified, consistent sequence, or including other evaluation dimensions to the problem, *e.g.* considering simultaneously the running time and the solution quality for the algorithm selection in portfolios, or matching both the users interests and skills to a job ad, in order to improve the chances that the application is considered by the recruiter – for Jam.

This final chapter first recalls the main results and lessons learned for algorithm portfolios and job-applicant matching problems, then discusses a natural follow-up to this work: combining both approaches for cold-start recommendation.

10.1 Summary of contributions

10.1.1 Algorithm portfolios to support the optimization expert

The combination of algorithms in a portfolio has been presented from three different perspectives. Portfolio are first positioned with respect to other optimization techniques (Chap. 2). Key points of algorithm portfolio design are then identified and discussed (Chap. 3). A new algorithm portfolio is proposed and extensive experiments are conducted to support the design choices.

The study focuses on the relevance of a few aspects and proposes the ASAP systems as a combination thereof. The technical contributions encompass the way both ASAP.V2 components are alternately optimized to enforce the division of work: the pre-scheduler is devoted to quickly solve "easy" optimization instances while the per-instance algorithm selector mainly addresses "hard" instances, also using features derived from the pre-scheduler performances. Three methods to fight overfitting while optimizing the pre-scheduler are introduced with ASAP.V2, are experimentally compared, and insights on their impact on the output schedule are provided.

This last point drives us to the lessons learned on algorithm portfolios and the databases used to assess them. Overfitting arises because of the limited number of instances in some datasets. As a consequence, it is not surprising that a number of state-of-the-art portfolio systems base their algorithm selector on ensemble methods such as random forests, which are known to be robust [Breiman, 2001]. The need for overfitting prevention has been demonstrated in ASAP.V2 for the construction of an optimized algorithm schedule.

The division of labor between the pre-scheduler and the selector is governed by the budget shares that are allocated to each component. An original contribution of the thesis is to formalize this division as the solution of a bi-objective optimization problem, balancing the number of instances solved in the pre-scheduler and the time deducted from the selector budget. The proposed approach has been shown to work well; however further work is needed to acquire a deeper understanding of this trade-off.

A last and crucial observation regards the diversity of datasets engaged in the algorithm portfolio assessment. Any hyperparameter tuning should better be done on a per-dataset basis, extending the search space of algorithm portfolio design. Recent approaches [Lindauer et al., 2015] have used a state-of-the-art algorithm configurator to achieve this. We stress, however, that such an approach is itself prone to overfitting – since it makes use of the same amount of data for a more complex calibration – and that careful attention should be paid to protect from it.

Taking inspiration from these principles, ASAP systems were favorably compared to state-of-the-art systems in two competition: ASAP.V1 was awarded a honourable mention at the ICON challenge of Algorithm Selection (2015) and ASAP.V2 won the Open Challenge on Algorithm Selection (2017).

Our work on the algorithm portfolio case study ends in Chap. 5 by raising some of the remaining challenges in the field and giving some perspectives for further work on the ASAP systems and sketching more general directions of research.

10.1.2 The job-applicant matching problem

The study of the Jam problem has been carried out in three steps, namely the overview of relevant concepts and approaches (Chap. 6), the in-depth analysis of two proprietary databases (Chap. 7) and the development of artificial neural network architectures to address the recommendation of brand-new jobs to known applicants, a.k.a. the item-oriented semi cold-start setting (Chap. 8).

As a first result, the Jam problem is more adequately formulated as a one-class collaborative filtering problem with item content information rather than as an information retrieval task or a pure collaborative filtering problem. The main challenges come from the misalignment of the three data sources, namely the textual description of users and items and the interaction data (users' clicks and applications). Compared to other similar recommendation problems, Jam involves the leveraging of natural language to a high degree, resulting in the development of specific representations for users and items from their textual descriptions.

A main contribution regards the way recommender systems provide users with recommendations. Distinction is made between *single region-of-interest* (SROI) approaches – which favor items that partially match all of a user's interests – and *multiple regions-of-interest* (MROI) systems – which put more emphasis on items that perfectly match a subset of the user's interests. In essence, recommender systems of this latter category keep multiple representations of the user that characterize all her centers of interests, rather than a compact mixture of interest. Most interestingly, MROI systems can handle inconsistency in a user's behaviors (*e.g.* incompatible features in the set of applications) and can reason with a "maximal match" rather than with a "sum of the matches" over a set of interests. MROI approaches are more relevant for Jam problems where the few user interactions does not allow to infer an accurate global representation of the user.

Taking inspiration from language modeling and metric learning, two neural architectures are proposed for item-oriented semi cold-start recommendation. LaJam leverage the similarity between jobs derived from the interaction data. Technical contributions are

proposed in the training function formulation – derived from the recall measure – and on negative sampling schemes to ensure a reliable training procedure. MaTJam is an SROI counterpart of LaJam, learning continuous language models to directly match resumes and jobs.

Both approaches are assessed against Jam databases. The main lessons include the flexibility of neural architecture to cope with mixed data type and the superiority of LaJam. MaTJam faces a critical challenge in the misalignment between resumes and users interaction data. LaJam is also compared on a publicly available dataset to CTR, a state-of-the-art algorithm based on latent topic modeling. Experiments show the merits of LaJam on CiteULike, confirming the relevance of MROI approaches even when the interaction data are moderately rare.

A few research directions are proposed for further investigation and detailed in Chap. 9, including the extensions of LaJam and MaTJam and a procedure to turn latent factor algorithms like MaTJam into MROI approaches.

10.2 Toward a portfolio of recommender systems

As a conclusion to this thesis, a direction of research – still unexplored to the best of our knowledge – is exposed: combining recommender systems into a portfolio. Recommendation problems contain a number of ingredients suggesting that they could benefit from an approach based on a portfolio of recommendation algorithms: on the one hand, approaches are extremely abundant and diverse; on the other hand the set of users is (as far as Jam is concerned) heterogeneous and all are not receptive to the same recommender systems.

The idea of combining a complementary set of recommender systems is not new. As early as in 2007, the (later) winning team of the Netflix Grand Prize [Bennett et al., 2007] noted how any single prediction method was outperformed by a linear combination of several of them [Bell and Koren, 2007].[1] Notably, neighborhood and latent factor models are complementary because they leverage structures of different scales in the data.

However, we suggest an orthogonal angle to benefit from a portfolio of approaches, based on using (informed) selection rather than a generic weighted average. To make this clear, let us draw a parallel with algorithm portfolio in optimization. A given recommender system is an algorithm which produces a per-user recommendation list, just like an optimization algorithm produces a per-instance solution. Both solutions may be measured against their quality w.r.t. the user or the instance requirements. The goal of a *recommender system selector* would consist in selecting the recommendation algorithm most appropriate to what is known about the user (history, profile, etc.) to provide her with personalized recommendations. As a concrete example in the Jam context, this selector would best use a neighborhood recommendation approach for a user with a limited interactions history, an SROI approach for a user with a larger number of passed interactions, and a direct-matching algorithm for a user with a rich resume. A related concept, *meta-learning*, studies the connections between the features of datasets (called their "metafeatures" to make the distinction with the features of samples clear) and how well the machine learning algorithms perform.[2] However, the portfolio of recommender

[1] The goal of the Netflix Grand Prize is the prediction of user ratings; the combination of several prediction methods is achieved with a weighted average of their predictions.

[2] This is the machine learning version of the algorithm selection problem studied in Part I. The similarity between meta-learning and (optimization) algorithm selection has been illustrated with the inclusion of a subset of OpenML dataset in the Open Algorithm Selection Challenge 2017 (see Sec. 4.7).

systems suggested here would learn both recommender systems and a recommender system selector on a single dataset (while meta-learning considers several datasets).

In Jam, a per-user recommender system portfolio would require a set of meaningful features – aside from a given resume. Throughout this thesis, some of the aspects of Jam have been investigated and would provide meaningful information about the user: NLP features of the resume (Sec. 7.2), features characterizing the set of known interactions (number, variety etc. – see Sec. 7.3), features based on the bipartite user-item graph, probing features (Sec. 3.2.2) based on simple recommender system algorithms, etc. The relevance of such features for a recommender systems selector appears as a promising direction of research.

The longer term perspective is in accordance with that of algorithm portfolios. How the strengths of several optimization algorithms can be combined inside the search and how recommender systems may join forces to produce a consistent, user-specific recommendation list, are two problems that show substantial similarities.

Bibliography

[Abel et al., 2016] Fabian Abel, András Benczúr, Daniel Kohlsdorf, MA Larson, and Róbert Pálovics. Recsys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'2016)*, pages 425–426. ACM, 2016.

[Abel et al., 2017] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'2017)*, pages 372–373. ACM, 2017.

[Abell et al., 2013] Tinus Abell, Yuri Malitsky, and Kevin Tierney. Features for exploiting black-box optimization problem structure. In *International Conference on Learning and Intelligent Optimization (LION7)*, pages 30–36. Springer, 2013.

[Aggarwal et al., 1999] Charu C Aggarwal, Joel L Wolf, Kun-Lung Wu, and Philip S Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 201–212. ACM, 1999.

[Ahalt et al., 1990] Stanley C Ahalt, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. Competitive learning algorithms for vector quantization. *Neural networks*, 3 (3):277–290, 1990.

[Aizawa, 2003] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[Amadini et al., 2014] Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. SUNNY: a lazy portfolio approach for constraint solving. *Theory and Practice of Logic Programming*, 14(4-5):509–524, 2014.

[Andrews et al., 2003] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS 2003)*, pages 577–584, 2003.

[Auger et al., 2012] A Auger, N Hansen, and M Schoenauer. Benchmarking of continuous black box optimization algorithms. *Evolutionary Computation*, 20(4):481, 2012.

[Balabanović and Shoham, 1997] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[Barkan and Koenigstein, 2016] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP'2016)*, pages 1–6. IEEE, 2016.

[Belkhir, 2017] Nacim Belkhir. *Per Instance Algorithm Configuration for Continuous Black Box Optimization*. PhD thesis, Université Paris-Saclay (France), 2017.

[Bell and Koren, 2007] Robert M Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[Bengio et al., 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb): 1137–1155, 2003.

[Bengio et al., 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'2009)*, pages 41–48. ACM, 2009.

[Bennett et al., 2007] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[Bird et al., 2009] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[Bischl et al., 2016] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[Bishop, 2006] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Blei et al., 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[Blum and Langley, 1997] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[Bojanowski et al., 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[Booker et al., 1999] Andrew J Booker, John E Dennis, Paul D Frank, David B Serafini, Virginia Torczon, and Michael W Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13, 1999.

[Branke et al., 2004] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 722–731. Springer, 2004.

[Breiman, 2001] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bromley et al., 1993] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[Burges et al., 2005] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML'2005)*, pages 89–96. ACM, 2005.

[Burges, 2010] Christopher JC Burges. From RankNet to LambdaRank to LambdaMART: An overview. *Learning*, 2010.

[Burke et al., 2010] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.

[Büttcher et al., 2010] Stefan Büttcher, Charles L A Clarke, and Gordon V Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

[Cao et al., 2007] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML'2007)*, pages 129–136. ACM, 2007.

[Carpi et al., 2016] Tommaso Carpi, Marco Edemanti, Ervin Kamberoski, Elena Sacchi, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana. Multi-stack ensemble for job recommendation. In *Proceedings of the Recommender Systems Challenge*, page 8. ACM, 2016.

[Cauwet et al., 2016] Marie-Liesse Cauwet, Jialin Liu, Baptiste Rozière, and Olivier Teytaud. Algorithm portfolios for noisy optimization. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):143–172, 2016.

[Cho et al., 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'2014)*, pages 1724–1734. ACL, 2014.

[Chopra et al., 2005] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, volume 1, pages 539–546. IEEE, 2005.

[Collautti et al., 2013] Marco Collautti, Yuri Malitsky, Deepak Mehta, and Barry O'Sullivan. Snnap: Solver-based nearest neighbor for algorithm portfolios. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD'2013)*, pages 435–450. Springer, 2013.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML'2008)*, pages 160–167. ACM, 2008.

[Culberson, 1998] Joseph C Culberson. On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation*, 6(2):109–127, 1998.

[Dagan et al., 2006] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine learning challenges*, pages 177–190. Springer, 2006.

[Das et al., 2007] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web (WWW'2007)*, pages 271–280. ACM, 2007.

[Das, 1999] Indraneel Das. On characterizing the "knee" of the pareto curve based on normal-boundary intersection. *Structural Optimization*, 18(2-3):107–115, 1999.

[Deb, 2003] Kalyanmoy Deb. Multi-objective evolutionary algorithms: Introducing bias among pareto-optimal solutions. In *Advances in evolutionary computing*, pages 263–292. Springer, 2003. ISBN 3-642-62386-7.

[Deb and Gupta, 2011] Kalyanmoy Deb and Shivam Gupta. Understanding knee points in bi-criteria problems and their implications as preferred solution principles. *Engineering optimization*, 43(11):1175–1204, 2011.

[Deerwester et al., 1990] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391, 1990.

[Degroote et al., 2016] Hans Degroote, Bernd Bischl, Lars Kotthoff, and Patrick De Causmaecker. Reinforcement learning for automatic online algorithm selection-an empirical study. In *Proceedings of the 16th Conference Information Technologies-Applications and Theory (ITAT'2016)*, volume 93, pages 93–101, 2016.

[Deshpande and Karypis, 2004] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22 (1):143–177, 2004.

[Desrosiers and Karypis, 2011] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2011.

[Dietterich et al., 1997] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71, 1997.

[Dos Santos et al., 2017] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Gaussian embeddings for collaborative filtering. In *40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017.

[Droste et al., 1999] Stefan Droste, Thomas Jansen, and Ingo Wegener. Perhaps not a free lunch but at least a free appetizer. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 833–839. Morgan Kaufmann Publishers Inc., 1999.

[Elkahky et al., 2015] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web (WWW'2015)*, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

[Färber et al., 2003] Frank Färber, Tim Weitzel, and Tobias Keim. An automated recommendation approach to selection in personnel recruitment. *Americas Conference on Information Systems (AMCIS'2003) proceedings*, page 302, 2003.

[Feurer et al., 2015] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 2962–2970, 2015.

[Fisher, 1937] Ronald Aylmer Fisher. *The design of experiments*. Oliver And Boyd; Edinburgh; London, 1937.

[Fouss et al., 2007] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[Gagliolo and Schmidhuber, 2006] Matteo Gagliolo and Jürgen Schmidhuber. Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 47(3-4): 295–328, 2006.

[Gantner et al., 2010] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *IEEE 10th International Conference on Data Mining (ICDM'2010)*, pages 176–185. IEEE, 2010.

[Gebser et al., 2011] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Marius Thomas Schneider, and Stefan Ziller. A portfolio solver for answer set programming: Preliminary report. In *International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'2011)*, pages 352–357. Springer, 2011.

[Gendreau and Potvin, 2005] Michel Gendreau and Jean-Yves Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, 2005.

[Gendreau and Potvin, 2010] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[Georgiev and Nakov, 2013] Kostadin Georgiev and Preslav Nakov. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *Proceedings of the 30th Annual International Conference on Machine Learning (ICML'2013)*, pages 1148–1156, 2013.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, 2010.

[Gomes and Selman, 2001] Carla P Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1):43–62, 2001.

[Gomes et al., 1998] Carla P Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 431–437. American Association for Artificial Intelligence, 1998.

[Gonard et al., 2016] François Gonard, Marc Schoenauer, and Michèle Sebag. Algorithm selector and prescheduler in the ICON challenge. In *International Conference on Metaheuristics and Nature Inspired Computing (META'2016)*, 2016.

[Gonard et al., 2017] François Gonard, Marc Schoenauer, and Michèle Sebag. Asap.v2 and asap.v3: Sequential optimization of an algorithm selector and a scheduler. In Marius Lindauer, Jan N. v. Rijn, and Lars Kotthoff, editors, *Proceedings of the Open Algorithm Selection Challenge*, volume 79 of *Proceedings of Machine Learning Research*, pages 8–11, Brussels, Belgium, 11–12 Sep 2017. PMLR. URL `http://proceedings.mlr.press/v79/gonard17a.html`.

[Gopalan et al., 2014] Prem K Gopalan, Laurent Charlin, and David M Blei. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems (NIPS 2014)*, pages 3176–3184, 2014.

[Gori and Pucci, 2007] Marco Gori and Augusto Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, volume 7, pages 2766–2771, 2007.

[Grbovic et al., 2015] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 1809–1818. ACM, 2015.

[Hadsell et al., 2006] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, volume 2, pages 1735–1742. IEEE, 2006.

[Hansen et al., 2003] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized Evolution Strategy with Covariance Matrix Adaptation. *Evolutionary Computation*, 11(1):1–18, 2003.

[Hansen et al., 2016] Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tusar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *arXiv preprint arXiv:1603.08785*, 2016.

[He et al., 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW'2017)*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[Heidrich-Meisner and Igel, 2009] Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'2009)*, pages 401–408. ACM, 2009.

[Hennig and Schuler, 2012] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13 (Jun):1809–1837, 2012.

[Herlocker et al., 1999] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237. ACM, 1999.

[Herlocker et al., 2004] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Hoffer and Ailon, 2015] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[Hoos, 2012] Holger H Hoos. Programming by optimization. *Communications of the ACM*, 55 (2):70–80, 2012.

[Hoos et al., 2014] Holger H Hoos, Marius Lindauer, and Torsten Schaub. claspfolio 2: Advances in algorithm selection for answer set programming. *Theory and Practice of Logic Programming*, 14(4-5):569–585, 2014.

[Hoos et al., 2015] Holger H Hoos, Roland Kaminski, Marius Lindauer, and Torsten Schaub. aspeed: Solver scheduling via answer set programming. *Theory and Practice of Logic Programming*, 15(1):117–142, 2015.

[Hu et al., 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE 8th International Conference on Data Mining (ICDM'08)*, pages 263–272. IEEE, 2008.

[Huang et al., 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[Huang et al., 2004] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[Huberman et al., 1997] Bernardo A Huberman, Rajan M Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997.

[Hutter et al., 2011] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization (LION5)*, 5:507–523, 2011.

[Hutter et al., 2013] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Identifying key algorithm parameters and instance features using forward selection. In *International Conference on Learning and Intelligent Optimization (LION7)*, pages 364–381. Springer, 2013.

[Joachims et al., 2005] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161. ACM, 2005.

[Joachims et al., 2007] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and

query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7, 2007.

[Kadioglu et al., 2010] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC-Instance-Specific Algorithm Configuration. In *European Conference on Artificial Intelligence (ECAI'2010)*, volume 215, pages 751–756, 2010.

[Kadioglu et al., 2011] Serdar Kadioglu, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm selection and scheduling. In *International Conference on Principles and Practice of Constraint Programming (CP'2011)*, pages 454–469. Springer, 2011.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 426–434. ACM, 2008.

[Koren and Bell, 2015] Yehuda Koren and Robert M Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 77–118. 2015.

[Koren et al., 2009] Yehuda Koren, Robert M Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[Kotthoff, 2013] Lars Kotthoff. Llama: leveraging learning to automatically manage algorithms. *arXiv preprint arXiv:1306.1031*, 2013.

[Kotthoff, 2015] Lars Kotthoff. ICON challenge on algorithm selection. *CoRR*, abs/1511.04326, 2015. URL `http://arxiv.org/abs/1511.04326`.

[Kotthoff, 2016] Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. In Christian Bessiere, Luc De Raedt, Lars Kotthoff, Siegfried Nijssen, Barry O'Sullivan, and Dino Pedreschi, editors, *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*, pages 149–190. Springer International Publishing, 2016.

[Kotthoff et al., 2016] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 17:1–5, 2016.

[Kotthoff et al., 2017] Lars Kotthoff, Marius Lindauer, and Jan N. v. Rijn. Open Algorithm Selection Challenge. `http://www.coseal.net/wp-content/uploads/2017/09/OASC_Results_17.pdf` [Online; accessed 02-Oct.-2017], 2017. Workshop COSEAL'2017.

[Krizhevsky et al., 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, pages 1097–1105, 2012.

[Kusner et al., 2015] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd Annual International Conference on Machine Learning (ICML'2015)*, pages 957–966, 2015.

[Kwok and Tsang, 2003] James T Kwok and Ivor W Tsang. Learning with idealized kernels. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML'2003)*, pages 400–407, 2003.

[Landauer and Dumais, 1997] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211, 1997.

[Larochelle and Murray, 2011] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 29–37, 2011.

[Le and Mikolov, 2014] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st Annual International Conference on Machine Learning (ICML'2014)*, pages 1188–1196, 2014.

[Le Berre and Simon, 2003] Daniel Le Berre and Laurent Simon. The essentials of the sat 2003 competition. In *International Conference on Theory and Applications of Satisfiability Testing (SAT'2003)*, pages 452–467. Springer, 2003.

[Leyton-Brown et al., 2002] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *International Conference on Principles and Practice of Constraint Programming (CP'2002)*, pages 556–572. Springer, 2002.

[Leyton-Brown et al., 2003] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. A portfolio approach to algorithm selection. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1542–1543, 2003.

[Lindauer et al., 2015] Marius Lindauer, Holger H Hoos, Frank Hutter, and Torsten Schaub. Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015.

[Lindauer et al., 2016] Marius Lindauer, Rolf-David Bergdoll, and Frank Hutter. An empirical study of per-instance algorithm scheduling. In *International Conference on Learning and Intelligent Optimization (LION10)*, pages 253–259. Springer, 2016.

[Lops et al., 2011] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[Loshchilov and Hutter, 2015] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[Ma et al., 2015] Tinghuai Ma, Jinjuan Zhou, Meili Tang, Yuan Tian, Abdullah Al-Dhelaan, Mznah Al-Rodhaan, and Sungyoung Lee. Social network and tag sources based augmenting collaborative recommender system. *IEICE Transactions on Information and Systems*, 98(4):902–910, 2015.

[Malherbe et al., 2014] 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 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 588–595. IEEE, 2014.

[Malinowski et al., 2006] Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. Matching people and jobs: A bilateral recommendation approach. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 137c–137c. IEEE, 2006.

[Malitsky et al., 2013] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, volume 13, pages 608–614, 2013.

[Maratea et al., 2014] Marco Maratea, Luca Pulina, and Francesco Ricca. A multi-engine approach to answer-set programming. *Theory and Practice of Logic Programming*, 14(6): 841–868, 2014.

[Mcnamee and Mayfield, 2004] Paul Mcnamee and James Mayfield. Character n-gram tokenization for european language text retrieval. *Information Retrieval*, 7(1):73–97, 2004.

[Mersmann et al., 2010] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *International Conference on Parallel Problem Solving from Nature (PPSN XI)*, pages 73–82. Springer, 2010.

[Mersmann et al., 2011] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 829–836. ACM, 2011.

[Michalewicz et al., 1996] Zbigniew Michalewicz, Dipankar Dasgupta, Rodolphe G Le Riche, and Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering*, 30(4):851–870, 1996.

[Mikolov et al., 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations Workshop*, 2013a.

[Mikolov et al., 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119, 2013b.

[Mısır and Sebag, 2017] Mustafa Mısır and Michèle Sebag. Alors: An algorithm recommender system. *Artificial Intelligence*, 244:291–314, 2017.

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 246–252, 2005.

[Mortensen and Pissarides, 1994] Dale T Mortensen and Christopher A Pissarides. Job creation and job destruction in the theory of unemployment. *The Review of Economic Studies*, 61(3):397–415, 1994.

[Mueller and Thyagarajan, 2016] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2786–2792. AAAI Press, 2016.

[Muñoz et al., 2012] Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization problems. In *International Conference on Parallel Problem Solving from Nature (PPSN XII)*, pages 226–235. Springer, 2012.

[Muñoz et al., 2017] Mario A Muñoz, Laura Villanova, Davaatseren Baatar, and Kate A Smith-Miles. Instance spaces for machine learning classification. *Machine Learning*, 2017.

[Neculoiu et al., 2016] Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, and Textkernel BV Amsterdam. Learning text similarity with siamese recurrent networks. *Association for Computational Linguistics 2016*, page 148, 2016.

[Ng, 2004] Andrew Y Ng. Feature selection, L1 vs L2 regularization and rotational invariance. In *Proceedings of the 21st Annual International Conference on Machine Learning (ICML'2004)*, 2004.

[Nguyen and Bai, 2010] Hieu V Nguyen and Li Bai. Cosine similarity metric learning for face verification. In *Asian Conference on Computer Vision*, pages 709–720. Springer, 2010.

[Nikolić et al., 2013] Mladen Nikolić, Filip Marić, and Predrag Janičić. Simple algorithm portfolio for sat. *Artificial Intelligence Review*, pages 1–9, 2013.

[Nudelman et al., 2004a] Eugene Nudelman, Kevin Leyton-Brown, Alex Devkar, Yoav Shoham, and Holger Hoos. Satzilla: An algorithm portfolio for sat. *Solver description, 2004 SAT competition*, 2004a.

[Nudelman et al., 2004b] Eugene Nudelman, Kevin Leyton-Brown, Holger H Hoos, Alex Devkar, and Yoav Shoham. Understanding random sat: Beyond the clauses-to-variables ratio. In *International Conference on Principles and Practice of Constraint Programming (CP'2004)*, pages 438–452. Springer, 2004b.

[Oentaryo et al., 2015] Richard Jayadi Oentaryo, Stephanus Daniel Handoko, and Hoong Chuin Lau. Algorithm selection via ranking. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 1826–1832, 2015.

[O'Mahony et al., 2008] Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan. Using case-based reasoning in an algorithm portfolio for constraint solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 210–216, 2008.

[Page et al., 1999] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[Pan et al., 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *IEEE 8th International Conference on Data Mining (ICDM'08)*, pages 502–511. IEEE, 2008.

[Papadopoulos et al., 2002] Harris Papadopoulos, Volodya Proedrou, Kostas Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *European Conference on Machine Learning (ECML'2002)*, volume 2, pages 345–356. Springer, 2002.

[Paparrizos et al., 2011] Ioannis Paparrizos, B Barla Cambazoglu, and Aristides Gionis. Machine learned job recommendation. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys'2011)*, pages 325–328. ACM, 2011.

[Paquet and Koenigstein, 2013] Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'2013)*, pages 999–1008. ACM, 2013.

[Pazzani and Billsus, 2007] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[Pedregosa et al., 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Pennington et al., 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'2014)*, volume 14, pages 1532–1543, 2014.

[Pfahringer et al., 2000] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In *Proceedings of the 17th Annual International Conference on Machine Learning (ICML'2000)*, pages 743–750, 2000.

[Pizzato and Bhasin, 2013] Luiz Augusto Pizzato and Anmol Bhasin. Beyond friendship: the art, science and applications of recommending people to people in social networks. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys'2013)*, pages 495–496. ACM, 2013.

[Powers, 1998] David MW Powers. Applications and explanations of Zipf's law. In *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, pages 151–160. Association for Computational Linguistics, 1998.

[Qamar and Gaussier, 2012] Ali M Qamar and Eric Gaussier. RELIEF algorithm and similarity learning for k-NN. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, 4:445–458, 2012.

[Radlinski and Joachims, 2007] Filip Radlinski and Thorsten Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 570–579. ACM, 2007.

[Rashid et al., 2002] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 127–134. ACM, 2002.

[Rendle et al., 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'2009)*, pages 452–461, 2009.

[Rice, 1976] John R Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[Ruotsalo et al., 2015] Tuukka Ruotsalo, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski. Interactive intent modeling: Information discovery beyond search. *Communications of the ACM*, 58(1):86–92, 2015.

[Salakhutdinov et al., 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML'2007)*, pages 791–798. ACM, 2007.

[Sarwar et al., 2000] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167. ACM, 2000.

[Sarwar et al., 2001] 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 (WWW'2001)*, pages 285–295. ACM, 2001.

[Sayag et al., 2006] Tzur Sayag, Shai Fine, and Yishay Mansour. Combining multiple heuristics. In *STACS*, volume 3884, pages 242–253. Springer, 2006.

[Schafer and Graham, 2002] Joseph L Schafer and John W Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[Schein et al., 2002] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[Schmitt et al., 2016] Thomas Schmitt, Philippe Caillou, and Michele Sebag. Matching jobs and resumes: a deep collaborative filtering task. In *2nd Global Conference on Artificial Intelligence (GCAI'2016)*, volume 41, pages 124–137, 2016.

[Schmitt et al., 2017] Thomas Schmitt, François Gonard, Philippe Caillou, and Michele Sebag. Language modelling for collaborative filtering: Application to job applicant matching. In *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI'2017)*. IEEE, 2017.

[Sedhain et al., 2015] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web (WWW'2015)*, pages 111–112. ACM, 2015.

[Shalev-Shwartz et al., 2004] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the 21st Annual International Conference on Machine Learning (ICML'2004)*, page 94. ACM, 2004.

[Shi et al., 2012] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM Conference on Recommender Systems (RecSys'2012)*, pages 139–146. ACM, 2012.

[Smith-Miles, 2009] Kate A Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6, 2009.

[Smith-Miles and Lopes, 2012] Kate A Smith-Miles and Leo Lopes. Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39 (5):875–889, 2012.

[Spielman and Teng, 2004] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[Srivastava et al., 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Stern et al., 2010] D Stern, R Herbrich, T Graepel, H Samulowitz, L Pulina, and A Tacchella. Collaborative expert portfolio management. In *Proceedings of the 24th AAAI National Conference on Artificial Intelligence (AAAI-10)*, volume 1, pages 179–184, 2010.

[Stern et al., 2009] David H Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World Wide Web (WWW'2009)*, pages 111–120. ACM, 2009.

[Streeter, 2003] Matthew J Streeter. Two broad classes of functions for which a no free lunch result does not hold. In *Proceedings of the 2003 Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1418–1430. Springer, 2003.

[Streeter and Smith, 2012] Matthew J Streeter and Stephen F Smith. New techniques for algorithm portfolio design. *arXiv preprint arXiv:1206.3286*, 2012.

[Stützle and Hoos, 1997] Thomas Stützle and Holger H Hoos. Max-min ant system and local search for the traveling salesman problem. In *IEEE International Conference on Evolutionary Computation (ICEC'97)*. Citeseer, 1997.

[Sun et al., 2017] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Yu Chen, and Chi Xu. MRLR: Multi-level representation learning for personalized ranking in recommendation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[Sutskever et al., 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112, 2014.

[Talbi, 2009] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009.

[Theano Development Team, 2016] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[Thornton et al., 2013] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 847–855. ACM, 2013.

[van der Maaten, 2009] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 384–391, 2009.

[van der Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey E Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[van der Maaten and Weinberger, 2012] Laurens van der Maaten and Kilian Q Weinberger. Stochastic triplet embedding. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP'2012).*, pages 1–6. IEEE, 2012.

[Verstrepen, 2015] Koen Verstrepen. *Collaborative Filtering with Binary, Positive-only Data.* PhD thesis, Universiteit Antwerpen (Belgium), 2015.

[Verstrepen and Goethals, 2014] Koen Verstrepen and Bart Goethals. Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys'2014)*, pages 177–184. ACM, 2014.

[Verstrepen and Goethals, 2015] Koen Verstrepen and Bart Goethals. Top-n recommendation for shared accounts. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'2015)*, pages 59–66. ACM, 2015.

[Wang and Blei, 2011] 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*, pages 448–456. ACM, 2011.

[Wang et al., 2014] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2014)*, pages 1386–1393, 2014.

[Wang et al., 2006] 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.

[Weimer et al., 2008] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS 2008)*, pages 1593–1600, 2008.

[Weinberger and Saul, 2009] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[Weston et al., 2012] Jason Weston, Chong Wang, Ron Weiss, and Adam Berenzeig. Latent collaborative retrieval. In *Proceedings of the 29th Annual International Conference on Machine Learning (ICML'2012)*, pages 443–450. Omnipress, 2012.

[Wolpert, 1996] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[Wolpert, 2002] David H Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002.

[Wolpert and Macready, 1997] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[Wu et al., 2017] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. *arXiv preprint arXiv:1706.07567*, 2017.

[Wu et al., 2016] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM, 2016.

[Xing et al., 2003] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS 2003)*, pages 521–528, 2003.

[Xu et al., 2007] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla-07: the design and analysis of an algorithm portfolio for sat. In *Principles and Practice of Constraint Programming (CP'2007)*, pages 712–727. Springer, 2007.

[Xu et al., 2008] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, pages 565–606, 2008.

[Xu et al., 2012a] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. *International Conference on Theory and Applications of Satisfiability Testing (SAT'2012)*, pages 228–241, 2012a.

[Xu et al., 2012b] Lin Xu, Frank Hutter, Jonathan Shen, Holger H Hoos, and Kevin Leyton-Brown. Satzilla2012: Improved algorithm selection based on cost-sensitive classification models. *Proceedings of the 2012 SAT Challenge*, pages 57–58, 2012b.

[Yang et al., 2006] Liu Yang, Rong Jin, Rahul Sukthankar, and Yi Liu. An efficient algorithm for local distance metric learning. In *Proceedings of the 21st National/18th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, volume 2, pages 543–548, 2006.

[Zheng et al., 2016] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33th Annual International Conference on Machine Learning (ICML'2016)*, pages 764–773, 2016.

[Zhou et al., 2007] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.

[Zhu et al., 2016] Fan Zhu, Jin Xie, and Yi Fang. Learning cross-domain neural networks for sketch-based 3d shape retrieval. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3683–3689. AAAI Press, 2016.

Appendices

Appendix A

Samples of data from Part II

A.1 Example of resume from Qapa

Opérateur / Opératrice en Publication Assistée par Ordinateur -PAO-
Chargé / Chargée de projet Marketing
Infographiste

- Réalisation des supports de vente commerciaux, emballages et institutionnels - national et export
- Conception et rédaction des communications de la société.
- Gestion des prestations et réalisations externes.
- Réalisation des thématiques et mises en avant des gammes (PLV et displays).- Création des identités visuelles des gammes.
- Réalisation des supports de vente commerciaux, emballages et institutionnels - national et export
- Réalisation des thématiques opérationnelles (PLV et displays).

En marketing
Etudes de marchés { reporting des informations de merchandisage
Gestion du marketing-mix de la marque et par enseigne GMS cliente.
Participation aux business plans par enseigne et élaboration du merchandisage préconisé.
Réalisation et suivi des opérations promotionnelles.
Veille concurrentielle { réponses aux appels d'offres.
Veille aux tendances marketing { force de proposition et préconisation.

En communication

Conception et rédaction de toute la communication de la société.
Création des supports print et web institutionnels de la marque.
Réalisation des supports de vente commerciaux, emballages et institutionnels (national et export) pour la marque et autres marques du groupe.

En infographie

Réalisation des displays permanents et thématiques. Et du

merchandisage attenant. Réalisation des opérations promotionnelles, thématiques.
Refonte et mises en avant des gammes (packagings, création des PLV et displays).
Préparation des éléments visuels des salons et séminaires.
Mises en conformité des chartes graphiques.

En management

Encadrement d'un éxécutant PAO déclinant les réalisations en multilingue pour clients export.
Gestion complète des prestations et réalisations print et media externes. Suivi de production.
Gestion du développement, hébergement et référencement du site web.
Relations avec les fournisseurs print, media, façonneurs de PLV, fabricant de packagings, agences de création des enseignes clientes, des services de qualité et certification. Reporting et gestion des budgets création-production.

A.2 Example of job ad from Qapa

Responsable de la communication - Buc

Responsable de la communication (H/F) - Premier groupe européen de haute technologie, Siemens conjugue depuis toujours performance technique, innovation, qualité et fiabilité.
Nos activités centrées sur les secteurs des infrastructures et des villes, l'industrie, l'énergie et la santé, ciblent des marchés de croissance sur lesquels le Groupe nourrit l'ambition d'être leader.
Vous intégrez l'équipe Communication en charge des entités françaises de Siemens Building Technologies. Notre objectif est de vous faire découvrir l'ensemble des missions d'un chargé de communication au sein d'un groupe international de haute technologie. A ce titre, vous avez pour missions :
* Vous allez au devant des interlocuteurs internes pour centraliser les informations,
* Vous rédigez des articles pour nos supports web, magasines ou médias internes,
* Vous participez à l'organisation d'événements (gestion logistique, coordination avec les agences, gestion ...

A.3 Example of resume from ABG

Etude du Rôle de l'IL-10 et du ligand de CD40 dans la physiopathologie des lymphomes non Hodgkiniens: rôle mitogène, antiapop
Biologie, médecine, santé
nan

1998.0
CHEF DE PROJET/RESPONSABLE R&D
Biologie/Biotechnologie
Gestion de projets avec mise au point, validation et application clinique de méthodes diagnostiques. Ecriture d'articles, brevet et présentations orales lors de congrès internationaux. Rédaction de projets scientifiques pour demande de financement. Veill Culture cellulaire, Test de cytotoxicité, Test de prolifération Immunohistochimie, Hybridation in situ, Cytométrie de flux, Test ELISA Extraction d'ARN, RT-PCR, Construction de plasmide, Transfection, Western/Northern blot Cancérologie, biomarqueur, biotechnologie Accélérez la recherche sur les biomarqueurs. Améliorer des services de surveillance cardio-vasculaire Travailler en équipe et appliquer les méthodes et les règles de la démarche qualité pour la mise en place de nouveaux projets
Gestion de projets R&D en biologie/biotechnologie en tant que Chef de Projet ou Consultant. Responsable R&D Attachée de Recherche Clinique
Autre

A.4 Example of job ad from ABG

Offre d'emploi CDI VALDEPHARM Ingénieur Recherche et Développement
Chimie
Milieux dilués et optique fondamentale
Recherche et développement
Docteur Ingénieur en chimie organique, vous possédez une expertise en chimie organique et une maîtrise des techniques d'analyse GC, HPLC, MS, RMN. Le poste exigeant des contacts à l'international, l'anglais courant est exigé. Une première expérience dans le développement de procédés serait un plus. Rattaché au Responsable Recherche & Développement et avec la responsabilité d'un Technicien supérieur, vous serez en charge des missions suivantes :
* Définir des voies de synthèse industrialisables ;
* Mettre au point, optimiser et développer les procédés de fabrication ;
* Prendre en charge tout ou partie des actions à mener lors des phases d'industrialisation ;
* Conduire les essais en laboratoire, en vue d'améliorer la productivité des opérations de production ;
* Réaliser les analyses des essais mis en œuvre ;
* Mettre en pratique sur le site les meilleurs process et technologies existantes, en faisant appel aux expertises internes, fournisseurs ou organismes compétents ;
* Participer à la formation des personnels de production, pour la mise en place des améliorations des nouveaux produits ou

remises à niveau ;
* S'assurer que les modifications apportées sont conformes à la réglementation en vigueur, en accord avec l'assurance qualité.
Apporter son concours au service Environnement, Hygiène, Sécurité pour la mise en place des informations relatives à la maîtrise des dangers, à la protection de l'environnement, et à la sécurité en général ;
* Participer à l'élaboration des spécifications des matières premières et des intermédiaires utilisés sur le site.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Démarrage à froid en recommandation : des portfolios d'algorithmes à l'appariement automatique d'offres et de chercheurs d'emploi

Mots clés : portfolio d'algorithme, systèmes de recommandation, démarrage à froid, optimisation, traitement du langage naturel

Résumé : La quantité d'informations, de produits et de relations potentielles dans les réseaux sociaux a rendu indispensable la mise à disposition de recommandations personnalisées. L'activité d'un utilisateur est enregistrée et utilisée par des systèmes de recommandation pour apprendre ses centres d'intérêt. Les recommandations sont également utiles lorsqu'estimer la pertinence d'un objet est complexe et repose sur l'expérience. L'apprentissage automatique offre d'excellents moyens de simuler l'expérience par l'emploi de grandes quantités de données.
Cette thèse examine le *démarrage à froid* en recommandation, situation dans laquelle soit un tout nouvel utilisateur désire des recommandations, soit un tout nouvel objet est proposé à la recommandation. En l'absence de données d'interaction, les recommandations reposent sur des descriptions externes. Deux problèmes de recommandation de ce type sont étudiés ici, pour lesquels des systèmes de recommandation spécialisés pour le démarrage à froid sont présentés.
En optimisation, il est possible d'aborder le choix d'algorithme dans un portfolio d'algorithmes comme un problème de recommandation. Notre première contribution concerne un système à deux composants, un sélecteur et un ordonnanceur d'algorithmes, qui vise à réduire le coût de l'optimisation d'une nouvelle instance d'optimisation tout en limitant le risque d'un échec de l'optimisation. Les deux composants sont entrainés sur les données du passé afin de simuler l'expérience, et sont alternativement optimisés afin de les faire coopérer. Ce système a remporté l'*Open Algorithm Selection Challenge* 2017.
L'appariement automatique de chercheurs d'emploi et d'offres est un problème de recommandation très suivi par les plateformes de recrutement en ligne. Une seconde contribution concerne le développement de techniques spécifiques pour la modélisation du langage naturel et leur combinaison avec des techniques de recommandation classiques afin de tirer profit à la fois des interactions passées des utilisateurs et des descriptions textuelles des annonces. Le problème d'appariement d'offres et de chercheurs d'emploi est étudié à travers le prisme du langage naturel et de la recommandation sur deux jeux de données tirés de contextes réels. Une discussion sur la pertinence des différents systèmes de recommandations pour des applications similaires est proposée.

Title: Cold-start recommendation: from algorithm portfolios to job applicant matching

Keywords: algorithm portfolio, recommender systems, cold-start, optimization, natural language processing

Abstract: The need for personalized recommendations is motivated by the overabundance of online information, products, social connections. This typically tackled by recommender systems (RS) that learn users interests from past recorded activities. Another context where recommendation is desirable is when estimating the relevance of an item requires complex reasoning based on experience. Machine learning techniques are good candidates to simulate experience with large amounts of data.
The present thesis focuses on the *cold-start* context in recommendation, *i.e.* the situation where either a new user desires recommendations or a brand-new item is to be recommended. Since no past interaction is available, RSs have to base their reasoning on side descriptions to form recommendations. Two of such recommendation problems are investigated in this work. Recommender systems designed for the cold-start context are designed.
The problem of choosing an optimization algorithm in a portfolio can be cast as a recommendation problem. We propose a two components system combining a per-instance algorithm selector and a sequential scheduler to reduce the optimization cost of a brand-new problem instance and mitigate the risk of optimization failure. Both components are trained with past data to simulate experience, and alternatively optimized to enforce their cooperation. The final system won the Open Algorithm Challenge 2017.
Automatic job-applicant matching (JAM) has recently received considerable attention in the recommendation community for applications in online recruitment platforms. We develop specific natural language (NL) modeling techniques and combine them with standard recommendation procedures to leverage past user interactions and the textual descriptions of job positions. The NL and recommendation aspects of the JAM problem are studied on two real-world datasets. The appropriateness of various RSs on applications similar to the JAM problem are discussed.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France