Knowledge-Based Systems 136 (2017) 37–45

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning approach

Shuo Yang[a,*], Mohammed Korayem[b], Khalifeh AlJadda[b], Trey Grainger[b], Sriraam Natarajan[a]

[a] Indiana University, Bloomington, IN, USA
[b] CareerBuilder, Norcross, GA, USA

ARTICLE INFO

Article history:
Received 15 November 2016
Revised 22 August 2017
Accepted 26 August 2017
Available online 1 September 2017

Keywords:
Recommendation system
Content-based filtering
Collaborative filtering
Statistical Relational Learning
Cost-sensitive learning

ABSTRACT

Recommendation systems usually involve exploiting the relations among known features and content that describe items (content-based filtering) or the overlap of similar users who interacted with or rated the target item (collaborative filtering). To combine these two filtering approaches, current model-based hybrid recommendation systems typically require extensive feature engineering to construct a user profile. Statistical Relational Learning (SRL) provides a straightforward way to combine the two approaches through its ability to directly represent the probabilistic dependencies among the attributes of related objects. However, due to the large scale of the data used in real world recommendation systems, little research exists on applying SRL models to hybrid recommendation systems, and essentially none of that research has been applied to real big-data-scale systems. In this paper, we proposed a way to adapt the state-of-the-art in SRL approaches to construct a real hybrid job recommendation system. Furthermore, in order to satisfy a common requirement in recommendation systems (i.e. that false positives are more undesirable and therefore should be penalized more harshly than false negatives), our approach can also allow tuning the trade-off between the precision and recall of the system in a principled way. Our experimental results demonstrate the efficiency of our proposed approach as well as its improved performance on recommendation precision.

1. Introduction

With their rise in prominence, recommendation systems have greatly alleviated information overload for their users by providing personalized suggestions for countless products such as music, movies, books, housing, jobs, etc. Since the mid-1990s, not only new theories of recommender systems have been presented but their application softwares have also been developed which involves various domains including e-government, e-business, e-commerce/e-shopping, e-learning, etc [1]. We consider a specific recommender system domain, that of job recommendations, and propose a novel method for this domain using statistical relational learning. This domain easily scales to billions of items including user resumes and job postings, as well as even more data in the form of user interactions between these items. CareerBuilder, the source of the data for our experiments, operates one of the largest job boards in the world. It has millions of job postings, more than 60 million actively-searchable resumes, over one billion searchable documents, and receives several million searches per hour [2]. The scale of the data is not the only interesting aspect of this domain, however. The job recommendations use case is inherently relational in nature, readily allowing for graph mining and relational learning algorithms to be employed. As Fig. 1 shows, very similar kinds of relationships exist among the jobs that are applied to by the same user and among the users who share similar preferences. If we treat every single job post or user as an object which has various attributes, the probability of a match between the target user and a job does not only depend on the attributes of these two target objects (i.e. target user and target job) but also the attributes of the related objects such as the patterns of the user's previous applied jobs, behaviors of users living in the same city or having the same education level. As we show in this work, richer modeling techniques can be used to determine these relationships faithfully. However, since most of the statistical relational learning approaches involve a searching space exponential to the number of related objects, how to efficiently build a hybrid recommenda-

* Corresponding author.
E-mail address: shuoyang@indiana.edu (S. Yang).

http://dx.doi.org/10.1016/j.knosys.2017.08.017
0950-7051/

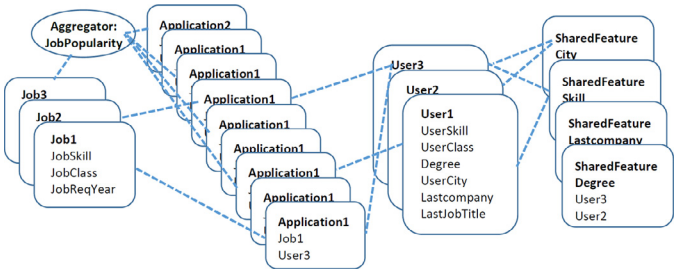

Fig. 1. Job recommendation domain.

tion system with statistical relational learning in such a large scale real-world problem remains a challenge in this field.

One of the most popular recommender approaches is *content-based filtering* [3], which exploits the relations between (historically) applied-to jobs and similar features among new job opportunities for consideration (with features usually derived from textual information). An alternative recommendation approach is based on *collaborative filtering* [4], which makes use of the fact that users who are interested in the same item generally also have similar preferences for additional items. Clearly, using both types of information together can potentially yield a more powerful recommendation system, which is why model-based hybrid recommender systems were developed [5]. While successful, these systems typically need extensive feature engineering to make the combination practical.

The hypothesis which we sought to verify empirically was that recent advancements in the fields of machine learning and artificial intelligence could lead to powerful and deployable recommender systems. In particular, we assessed leveraging Statistical Relational Learning (SRL) [6], which combines the representation abilities of rich formalisms such as first-order logic or relational logic with the ability of probability theory to model uncertainty. We employed a state-of-the-art SRL formalism for combining content-based filtering and collaborative filtering. SRL can directly represent the probabilistic dependencies among the attributes from different objects that are related with each other through certain connections (in our domain, for example, the jobs applied to by the same user or the users who share the same skill or employer). SRL models remove the necessity for an extensive feature engineering process, and they do not require learning separate recommendation models for each individual item or user cluster, a requirement for many standard model-based recommendation systems [4,7].

We propose a hybrid model combining content-based filtering and collaborative filtering that is learned by an efficient statistical relational learning approach - Relational Functional Gradient Boosting (RFGB) [8]. Specifically, we define the target relation as *Match*(*User, Job*) which indicates that the user–job pair is a match when the grounded relation is true, hence that job should be recommended to the target user. The task is to predict the probability of this target relation *Match*(*User, Job*)[1] for users based on the information about the job postings, the user profile, the application history, as well as application histories of users that have the similar preferences or profiles as the target user. RFGB is a boosted model which contains multiple relational regression trees with additive regression values at the sink node of each path. Our hypothesis is that these trees can capture many of the weak relations that exist between the target user and the job with which he/she is matched.

In addition, this domain has practical requirements which must be considered. For example, we would rather overlook some of the candidate jobs that could match the users (false negatives) than send out numerous spam emails to the users with inappropriate job recommendations (false positives). The cost matrix thus does not contain uniform cost values, but instead needs to represent a higher cost for the user–job pairs that are false positives compared to those that are false negatives, i.e. precision is preferred over recall. To incorporate such domain knowledge within the cost matrix, we adapted the previous work from [9], which extended RFGB by introducing a penalty term into the objective function of RFGB so that the trade-off between the precision and recall can be tuned during the learning process.

In summary, we considered the problem of matching a user with a job and developed a hybrid content-based filtering and collaborative filtering approach. We adapted a successful SRL algorithm for learning features and weights and are the first to implement such a system in a real-world big data context. Our algorithm is capable of handling different costs for false positives and false negatives making it extremely attractive for deploying within many kinds of recommendation systems, including those within the domain upon which we tested. Our proposed approach has three main innovations: 1. it is the first work which employs probabilistic logic models to build a real-world large-scale job recommendation system; 2. it is the first work which allows the recommender to incorporate special domain requirements of an imbalanced cost matrix into the model learning process; 3. it is the first to prove the effectiveness of statistical relational learning in combining the collaborative filtering and content-based filtering with real-world job recommendation system data.

2. Related work

Recommendation systems usually handle the task of estimating the relevancy or ratings of items for certain users based on information about the target user–item pair as well as other related items and users. The recommendation problem is usually formulated as f: $U \times I \rightarrow R$ where U is the space of all users, I is the space of all possible items and f is the utility function that projects all combinations of user-item pairs to a set of predicted ratings R which is composed by nonnegative integers. For a certain user u, the recommended item would be the item with the optimal utility value, i.e. $u_i^* = argMax_{i \in I} f(u, i)$. The user space U contains the information about all the users, such as their demographic characteristics, while the item space I contains the feature information of all the items, such as the genre of the music, the director of a movie, or the author of a book.

Generally speaking, the goal of *content-based filtering* is to define recommendations based upon feature similarities between the items being considered and items which a user has previously rated as interesting [10], i.e. for the target user-item rating $f(\hat{u}, \hat{i})$, *content-based filtering* would predict the optimal recommendation based on the utility functions of $f(\hat{u}, I_h)$ which is the historical rating information of user $\hat{u}$ on items (I_h) similar with $\hat{i}$. Given their origins out of the fields of information retrieval and information filtering, most content-based filtering systems are applied to items that are rich in textual information. From this textual information, item features I are extracted and represented as *keywords* with respective weighting measures calculated by certain mechanisms such as the *term frequency/inverse document frequency (TF/IDF)* measure [11]. The feature space of the user U is then constructed from the feature spaces of items that were previously rated by that user through various keyword analysis techniques such as averaging approach [12], Bayesian classifier [7], etc. Finally, the utility function of the target user-item pair $f(\hat{u}, \hat{i})$ is calculated by some scoring heuristic such as the cosine similarity [11] between the user profile vector and the item feature vector or some traditional machine learning models [7]. *Overspecialization* is one

[1] Following standard practice inside the machine learning community, we use the terms *relations* and *predicates* interchangeably.

of the problems with content-based filtering, which includes the cases where users either get recommendations too similar to their previously rated items or otherwise never get recommendations diverse enough from the items they have already seen. Besides, since the model-based content-based filtering builds its recommendation model based on the previous rated items of the target user, it requires a significant amount of items to be rated in advance in order to give accurate recommendations especially for the probabilistic machine learning models which require the number of training examples at the exponential scale of the dimension of the feature space.

On the other hand, the goal of the *collaborative filtering* is to recommend items by learning from users with similar preferences [10,13–15] i.e. for the target user-item rating $f(\hat{u}, \hat{i})$, *collaborative filtering* builds its belief in the best recommendation by learning from the utility functions of $f(U_s, \hat{i})$ which is the rating information of the user set U_s that has similar preferences as the target user $\hat{u}$. The commonly employed approaches fall into two categories: *memory-based (or heuristic-based)* and *model-based* systems. The heuristic-based approaches usually predict the ratings of the target user-item pair by aggregating the ratings of the most similar users for the same item with various aggregation functions such as mean, similarity weighted mean, adjusted similarity weighted mean (which uses relative rating scales instead of the absolute values to address the rating scale differences among users), etc. The set of most similar users and their corresponding weights can be decided by calculating the correlation (such as Pearson Correlation Coefficient [16]) or distance (such as cosine-based [4] or mean squared difference) between the rating vectors of the target user and the candidate user on common items. Whereas model-based algorithms are used to build a recommendation system by training certain machine learning models [4,17–19] based on the ratings of users that belong to the same cluster or class as the target user. Hence, prior research has focused on applying statistical relational models to collaborative filtering systems [20–23]. Although collaborative filtering systems can solve the overspecialization problem present in the content-based filtering approach, it has its own problems as well, such as the new user/item problem (commonly known as the “cold start” problem) and the *sparsity* problem, which occurs when the number of users ratings on certain items is not sufficient. Hence, there are works focusing on enhancing collaborative filtering systems for solving such problems. Shambour et al. [24] proposed a G2B recommendation e-services which alleviated the sparsity and cold start problems by employing additional domain knowledges of trust and trust propagation.

There are *Hybrid approaches* which combine collaborative filtering and content-based filtering into a unified system [5,25,26]. For instance Basilico et al. [5] unified content-based and collaborative filtering by engineering the features based on various kernel functions, then trained a simple linear classifier (Perceptron) in this engineered feature space.

There are some research focusing on job recommender systems. However, most of them only exploit the techniques of content-based filtering [27–31]. Hong et al. [32] proposed a hybrid job recommender system by profiling the users based on the historical applied jobs and behaviors of job applicants. Lu et al. [33] proposed a directed weighted graph which represents the content-based and interaction-based relations among users, jobs and employers with directed or bidirectional edges. It computes the content-based similarity between any two profiles of objects (user, employer or job). The key difference of our model from theirs is that the graph they used is not a machine learning model trained from the historical data, but rather built based on known facts of the target objects whereas our model is a first-order logic probabilistic model trained with historical data and only partially grounded with the related objects when it is necessary for inference on the target objects. Pacuk et al. [34] also exploited gradient boosting. But they only built a content-based filtering recommender by using the standard gradient boosting. We built a hybrid recommender with relational functional gradient boosting, which can capture the dependences among the features not only from the target user-item pair but also from similar users. Besides, our model is a cost-sensitive learning approach which allows the tunning of precision and recall in a principled way.

The most related work to ours is [35], where they proposed to use Markov Logic Networks to build hybrid models combining content-based filtering and collaborative filtering. Their work only employed one type of probabilistic logic model, which is demonstrated later in this paper to not be the best one. Besides, it did not consider the special requirement of many recommendation systems, which is that precision should be preferred over recall (or at least that the relative weights of the two should be configurable).

3. Building hybrid job recommendation systems with SRL models

Traditional machine learning algorithms make a fundamental assumption about the data they try to model: the training samples are independent and identically distributed (i.i.d.), which is not typically the case in recommendation systems. In order to represent the data in a flat table, the standard model-based recommendation systems need an exhaustive feature engineering process to construct the user profile by aggregating the attributes over all the similar users who share the same background or similar preferences as the target user. The aggregation-based strategies are necessary because the standard algorithms require a regular flat table to represent the data. However, the number of similar users related to the target user may vary a lot among different individuals. For example, users with common preferences could have more similar users than the users with unique tastes. There are aggregation-based strategies [36] to make the feature number identical for all the samples when extending the feature space. However, such strategies would inevitably lose meaningful information otherwise introduce some amount of noise.

We propose to employ SRL for the challenging task of implementing a hybrid recommendation system. Specifically, we consider the formulation of Relational Dependency Networks (RDNs) [37], which are approximate graphical models that are inferred using the machinery of Gibbs sampling. Fig. 2 shows a template model of RDNs. As can be seen, other than the attributes of the target user A and target job B, it also captures the dependencies between the target predicate *Match*(*A*, *B*) and attributes from the similar user D and previous applied job C. The interpretation of the learned model will be explained in more detail in Section 4.

As an approximation of Bayesian Networks, Dependency Networks (DNs) make the assumption that the joint distributions can be approximated as the product of individual conditional probability distributions and that these conditional probability distributions are independent from each other. Since it prevents the need for acyclicity checking, the structure and conditional probability of each node can be optimized separately by certain local search strategy. RDNs extend DNs to relational data and are considered as one of the most successful SRL models that have been applied to real-world problems. Hence, we propose to construct a hybrid recommendation system by learning an RDN using a state-of-the-art learning approach–Relational Functional Gradient Boosting (RFGB) which has been proven to be one of the most efficient relational learning approaches [8].

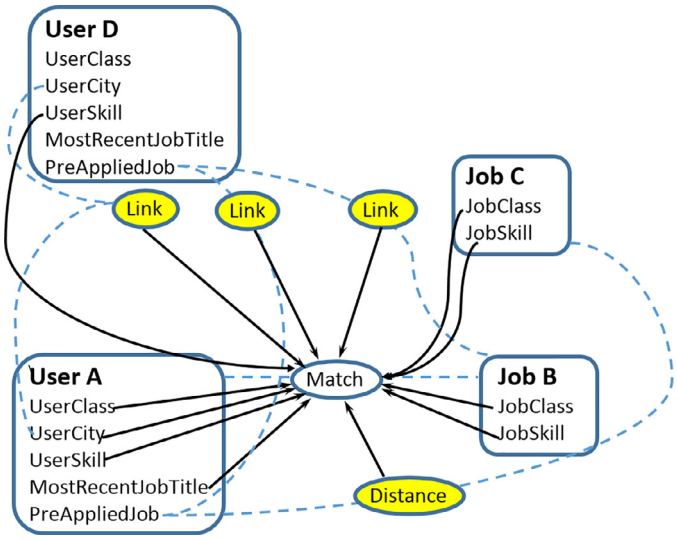

Fig. 2. Template Model of a Sample RDN. The target is *Match*(*UserA, JobB*) while the related objects are User D (introduced by the link nodes) and previous applied Job C. Note that D and C are first-order variables which could have multiple groundings for different target user–job pairs.

The following subsections will first introduce the basic concept of RFGB, then cover the way we incorporate domain knowledge to the cost matrix so the proposed hybrid recommendation system can improve the confidence of recommended jobs.

3.1. Relational functional gradient boosting

When fitting a probabilistic model $P(y|x)$, standard gradient ascent approaches start with initial parameters θ_0 and iteratively add the gradient (Δ_i) of an objective function with respect to θ_i. Friedman [38] proposed an alternate approach where the objective function is represented using a regression function ψ over the examples $\mathbf{x}$, and the gradients are derived w.r.t. $\psi(x)$. Similar to parametric gradient descent, after n iterations of functional gradient descent, the function parameter can be given as the sum over all the gradient terms $\psi_n(x) = \psi_0(x) + \Delta_1(x) + \cdots + \Delta_n(x)$.

Each gradient term (Δ_m) is a set of training examples and regression values given by the gradient w.r.t $\psi_m(x_i)$, i.e., $< x_i, \Delta_m(x_i) = \frac{\partial LL(\mathbf{x})}{\partial \psi_m(x_i)} >$. To generalize from these regression examples, a regression function $\hat{\psi}_m$ (generally regression tree) is learned to fit to the gradients. The final model $\psi_m = \psi_0 + \hat{\psi}_1 + \cdots + \hat{\psi}_m$ is a sum over these regression trees. Functional gradient ascent is also known as functional gradient boosting (FGB) due to this sequential nature of learning models.

FGB has been applied to relational models [8,39–41] because of its ability to learn structures and parameters of these models simultaneously. Gradients are computed for every grounding/instantiation of the target first-order predicate. In our case, the grounding *Match(John, Software Engineer)* of the target predicate *Match(User, Job)* could be one example. Relational regression trees [42] are learned to fit the ψ function over the relational regression examples. Since the regression function $\psi : X \rightarrow (-\infty, \infty)$ is unbounded, a sigmoid function over ψ is commonly used to represent conditional probability distributions. Thus the RFGB log-likelihood function is:

$$LL = \sum_i \log P(y_i = \hat{y}_i; \mathbf{X}_i) = \sum_i \log \frac{1}{1 + \exp(-\hat{y}_i \cdot \psi(y_i = \hat{y}_i; \mathbf{X}_i))}$$

where y_i corresponds to a target grounding (a grounded instance of the target predicate) of example i with parents $\mathbf{X}_i$. In our case, the target predicate is *Match(User, Job)*, and the parents $\mathbf{X}_i$ would be the attributes of the target user and target job, and the jobs previously applied to by the target user and similar users sharing the same preferences. $\hat{y}_i$ is the true label for a user–job pair which is 1 for a positive matching pair and 0 for a negative matching pair. The key assumption is that the conditional probability of a target grounding y_i, given all the other predicates, is modeled as a sigmoid function.

The gradient w.r.t. $\psi(y_i = \hat{y}_i; \mathbf{X}_i)$ is

$$\frac{\partial LL(\mathbf{x})}{\partial \psi(y_i = \hat{y}_i; \mathbf{X}_i)} = I(\hat{y}_i = Match) - P(y_i = Match; \mathbf{X}_i) \quad (1)$$

which is the difference between the true observation (I is the indicator function) and the current predicted probability of the match being true. Note the indicator function, I returns 1 for positives and 0 for negatives. Hence the positive gradient terms for positive examples push the regression values closer to ∞ and thereby the probabilities closer to 1, whereas for negative examples, the regression values are pushed closer to $-\infty$ and the probabilities closer to 0.

3.2. Cost sensitive learning with RFGB

Following the work of Yang et al. [9], we propose to construct a hybrid job recommendation system by learning a cost-sensitive RDN.

As shown in Eq. (1), the magnitude (absolute value) of the gradient in RFGB only depends on how well the current model fits the example. If it fits well, the probability of the positive example given the current model would be close to 1 (0 for negative examples), and the gradient that will be assigned to such examples as the training weight would approach 0. If it does not, the predicted probability of the example would be far from the true label and hence cause the boosting algorithm to attach a high weight to that example. As a result, this method treats both false positive and false negative examples in the same way. Since most of the relational data suffers from class imbalance, where negative instances compose a much larger part of the training samples compared with positive instances, the negative outliers would easily dominate the classification boundary after a few iterations. So, Yang et al. [9] proposed a cost-sensitive relational learning approach which is able to address these issues and model the target task more faithfully. This is achieved by adding a term to the objective function that penalizes false positives and false negatives differently. They defined the cost function as:

$$c(\hat{y}_i, y_i) = \alpha I(\hat{y}_i = 1 \wedge y_i = 0) + \beta I(\hat{y}_i = 0 \wedge y_i = 1),$$

where $\hat{y}_i$ is the true label of the ith instance and y_i is the predicted label. $I(\hat{y}_i = 1 \wedge y_i = 0)$ is 1 for false negatives (in our case, the matching user–job pair that is predicted as mis-matching) and $I(\hat{y}_i = 0 \wedge y_i = 1)$ is 1 for false positives (in our case, the mismatching user–job pair that is classified as matching). This cost function was hence being introduced into the normalization term of the objective function as:

$$\log J = \sum_i \left[\psi(y_i; \mathbf{X}_i) - \log \sum_{y_i'} \exp\left(\psi(y_i'; \mathbf{X}_i) + c(\hat{y}_i, y_i')\right) \right]$$

Thus, in addition to simple log-likelihood of the examples, the algorithm also takes into account these additional costs.

Then, the gradient of the objective function w.r.t $\psi(y_i = 1; \mathbf{X}_i)$ can be calculated by:

$$\Delta = I(\hat{y}_i = Match) - \lambda P(y_i = Match; \mathbf{X}_i). \quad (2)$$

Table 1
Cost Matrix.

		Actual Class	
		True	False
Predicted Class	True	0	$I(\hat{y}_i = 1) - \frac{P(y_i=1:\mathbf{X}_i)}{P(y'=1:\mathbf{X}_i)+P(y'=0:\mathbf{X}_i)\cdot e^{-\beta}}$
	False	$I(\hat{y}_i = 1) - \frac{P(y_i=1:\mathbf{X}_i)}{P(y'=1:\mathbf{X}_i)+P(y'=0:\mathbf{X}_i)\cdot e^{\alpha}}$	0

where $\lambda =$

$$\begin{cases} For\ \ matching\ \ User-Job\ \ pairs: \\ \dfrac{1}{P(y'=Match;\mathbf{X}_i)+P(y'=MisMatch;\mathbf{X}_i)\cdot e^{\alpha}} \\ For\ \ mis-matching\ \ User-Job\ \ pairs: \\ \dfrac{e^{\beta}}{P(y'=Match;\mathbf{X}_i)\cdot e^{\beta}+P(y'=MisMatch;\mathbf{X}_i)} \end{cases} \quad (3)$$

As shown above, the cost function $c(\hat{y}_i, y_i)$ is controlled by α when a potentially matching job being ruled out by the recommender, while being controlled by β when a mis-matching job being recommended. The cost matrix of our approach can be formally defined as (Table 1).

As the cost matrix shows, the influence of false negative and false positive examples on the final learned model can be directly controlled by tuning the parameters α and β respectively.

Now, consider the special requirement on the cost matrix in most job recommendation systems, which is that we would rather miss certain candidate jobs which to some extent match the target user than send out recommendations that are not appropriate to the target user. In other words, we prefer high precision as long as the recall maintains above such a reasonable value that the system would not return zero recommendation for the target user.

Since α is the parameter controlling the weights of false negative examples, we simply assign it as 0 which makes $\lambda = 1 / \sum_{y'} [P(y'; \mathbf{X}_i)] = 1$ for misclassified positive examples. As a result, the gradient of the positive examples is the same as it was in the original RFGB settings.

For the false positive examples, we use a harsher penalty on them, so that the algorithm would put more effort into classifying them correctly in the next iteration. According to Eq. (3), when it is a negative example ($\hat{y}_i = 0$), we have

$$\lambda = \frac{1}{P(y'=Match;\mathbf{X}_i)+P(y'=MisMatch;\mathbf{X}_i)\cdot e^{-\beta}}.$$

As $\beta \to \infty$, $e^{-\beta} \to 0$, hence $\lambda \to 1 / P(y_i = Match; \mathbf{X}_i)$, so

$$\Delta = 0 - \lambda P(y_i = Match; \mathbf{X}_i) \to -1$$

This means that the gradient is pushed closer to its maximum magnitude $|-1|$, no matter how close the predicted probability is to the true label. On the other hand, when $\beta \to -\infty$, $\lambda \to 0$, hence $\Delta \to 0$, which means that the gradients are pushed closer to their minimum value of 0. *In summary, if $\alpha < 0$ ($\beta < 0$), the algorithm is more tolerant of misclassified positive (negative) examples. Alternately, if $\alpha > 0$ ($\beta > 0$), the algorithm penalizes misclassified positive (negative) examples even more than standard RFGB.* So, in our experiment, we set $\beta > 0$, which amounts to putting a large cost on the false positive examples.

The proposed approach is shown in Fig. 3. We iterate through M steps and in each iteration, we generate examples based on the cost-sensitive gradients. We learn a relational regression tree to fit the examples using FitRelRegressionTree which is added to the current model. We limit our trees to have maximum L leaves and greedily pick the best node to expand. For generating the regression examples (GenSoftMEgs function), we iterate through all the examples (N in the algorithm). For each example, we calculate the probability of the example being true based on the current model. We then calculate the parameter λ for positive and negative examples respectively based on the set values of parameters α and β, which are then used to calculate gradients based on Eq. (2). The example and its gradient are added to the set of regression examples, S.

Table 2
Feature Space.

Variable Name	SkillID	ClassID	Distance
Num of Instances	8534	1867	4
Variable Name	CityName	CompanyID	JobTitle
Num of Instances	22,137	1,154,623	823,733

In job recommendation systems, the major goal is typically not to have misclassified negative examples (false positive). As a result, we need to eliminate the noise/outliers in negative examples as much as possible. Most algorithms generate negative examples by randomly drawing objects from two related variables, and the pair that is not known as positively-related based on the given facts is assumed to be a negative pair. However, in our case, if we randomly draw instances from *User* and *Job*, and assume it is a negative example if that grounded user never applied to that grounded job, it could introduce a lot of noise into the data since not applying does not necessarily indicate the job not matching the user. For example, it could simply be due to the fact that the job has never been seen by the user. Hence, instead of generating negative instances following a "closed-world assumption", as most of the relational approaches did, we instead generated the negative examples by extracting the jobs that were sent to the user as recommendations but were not applied to by the user. In this way, we can guarantee that this User–Job pair is indeed not matching.

4. Experiments

We extracted 4 months of user job application history and active job posting records and evaluated our proposed model on that data. Our goal was to investigate whether our proposed model can efficiently construct a hybrid recommendation system with cost-sensitive requirements by explicitly addressing the following questions:

- **(Q1)** How does combining collaborative filtering improve the performance compared with content-based filtering alone?
- **(Q2)** Can the proposed cost-sensitive SRL learning approach reduce false positive prediction without sacrificing too much on other evaluation measurements?

To answer these questions, we extracted 9 attributes from user resumes as well as job postings, which are defined as first-order predicates: *JobSkill(JobID, SkillID), UserSkill(UserID, SkillID), JobClass (JobID, ClassID), UserClass(UserID, ClassID), PreAppliedJob(UserID, JobID), UserJobDis(UserID, JobID, Distance),*

UserCity(UserID, CityName), MostRecentCompany(UserID, CompanyID), MostRecentJobTitle(UserID, JobTitle).

There are 707,820 total job postings in our sample set, and the number of possible instances the first order variables can take is shown in the Table 2.

function SoftRFGB($Data$)
  for $1 \leq m \leq M$ **do** ▷ Iterate through M gradient steps
    $S :=$GenSoftMEgs($Data; F_{m-1}$) ▷ Generate examples
    $\Delta_m :=$FitRelRegressTree(S) ▷ Relational Tree learner
    $F_m := F_{m-1} + \Delta_m$ ▷ Update model
  end for
end function
function GenSoftMEgs($Data, F$) ▷ Example generator
  $S := \emptyset$
  for $1 \leq i \leq N$ **do** ▷ Iterate over all examples
    Compute $P(y_i = 1|\mathbf{x}_i)$ ▷ Probability of the example being true
    if $\hat{y}_i = 1$ **then**
      $\lambda = 1$ ▷ Compute parameter λ for positive examples
    else
      $\lambda = 1/[P(y_i = 1|\mathbf{x}_i) + P(y_i = 0|\mathbf{x}_i) \cdot e^{-\beta}]$
      ▷ Compute parameter λ for negative examples
    end if
    $\Delta(y_i; \mathbf{x}_i) := I(\hat{y}_i = 1) - \lambda P(y_i = 1|\mathbf{x}_i)$ ▷ Cost of current example
    $S := S \cup [(y_i), \Delta(y_i; \mathbf{x}_i))]$ ▷ Update relational regression examples
  end for
return S ▷ Return regression examples
end function
function FitRelRegressionTree(S) ▷ Relational tree Learner
  Tree := createTree($P(X)$)
  Beam := {root(Tree)}
  while $numLeaves(Tree) \leq L$ **do**
    Node := popBack(Beam) ▷ Node w/ worst score
    C := createChildren(Node) ▷ Create children
    BN := popFront(Sort(C, S)) ▷ Node w/ best score
    addNode(Tree, Node, BN) ▷ Replace Node with BN
    insert(Beam, BN.left, BN.left.score) ▷ Insert branch
    insert(Beam, BN.right, BN.right.score)
  end while
return Tree
end function

Fig. 3. Cost-sensitive RFGB.

Table 3
Domains.

JobTitle	Training			Test		
	pos	neg	facts	pos	neg	facts
Retail Sales Consultant	224	6973	13,340,875	53	1055	8,786,938
Case Manager	387	35,348	13,537,324	87	5804	8,815,216
District Manager	358	16,014	13,396,635	87	3521	8,798,522

Table 4
Results.

Job Title	Approach	FPR	FNR	Precision	Recall	Accuracy	AUC-ROC
Retail	Content-based Filtering (CF)	0.537	0.321	0.060	0.679	0.473	0.628
Sales	Cost-sensitive CF ($\alpha 0\beta 2$)	**0.040**	0.868	0.143	0.132	**0.921**	0.649
Consultant	Hybrid Recommender (HR)	0.516	**0**	0.089	**1.0**	0.509	**0.776**
	Cost-sensitive HR ($\alpha 0\beta 2$)	0.045	0.906	0.096	0.094	0.914	0.755
	Cost-sensitive HR ($\alpha 0\beta 1$)	0.113	0.623	**0.144**	0.377	0.863	0.772
Case	Content-based Filtering (CF)	0.220	0.184	0.053	0.816	0.781	0.861
Manager	Cost-sensitive CF ($\alpha 0\beta 2$)	0.084	0.609	0.066	0.391	0.909	0.847
	Hybrid Recommender (HR)	0.239	**0**	0.059	**1.0**	0.765	**0.911**
	Cost-sensitive HR ($\alpha 0\beta 2$)	**0.037**	0.736	**0.096**	0.264	**0.952**	**0.911**
District	Content-based Filtering (CF)	0.427	0.195	0.045	0.805	0.579	0.746
Manager	Cost-sensitive CF ($\alpha 0\beta 2$)	0.017	0.920	**0.104**	0.080	0.961	0.745
	Hybrid Recommender (HR)	0.439	**0**	0.053	**1.0**	0.572	0.817
	Cost-sensitive HR ($\alpha 0\beta 2$)	**0.013**	0.977	0.042	0.023	**0.964**	0.812
	Cost-sensitive HR ($\alpha 0\beta 1$)	0.068	0.678	**0.104**	0.322	0.917	**0.825**

Information on the *JobClass* and *UserClass* are extracted based upon the work of Javed et al. [43]. The other features related to users are *UserSkill, UserCity, MostRecentCompany* and *MostRecentJobTitle* which are either extracted from the user's resume or the user's profile document, whereas the job feature *JobSkill* represents a desired skill extracted from the job posting. Predicate *UserJobDis* indicates the distance between the user (first argument) and the job (second argument), which is calculated based on the user and job locations extracted from respective documents. The *UserJobDis* feature is discretized into 4 classes (1: <15 mile; 2: [15 miles, 30 miles); 3: [30 miles, 60 miles]; 4: >60 miles). The predicate *PreAppliedJob* defines the previous applied jobs and serves as an independent predicate which indicates whether the target user is in a cold start scenario, as well as act as a bridge which introduces into the searching space the attributes of other jobs related to the target user during the learning process.

To incorporate collaborative filtering, we use three additional first-order predicates: *CommSkill(UserID1, UserID2), CommClass (UserID1, UserID2)* and *CommCity(UserID1, UserID2)* which are induced from the given groundings of the predicates *UserSkill, UserClass* and *UserCity* and also serve as bridges which introduce features of other users who share the similar background with the target user.

The performance of our model is evaluated in 3 different user classes, each of which has its data scale description shown in Table 3.

As Natarajan et al. discovered in their work [8], limiting the number of leaves in each tree and learning a set of small trees can improve the learning efficiency as well as prevent overfitting compared with learning a single complex tree. To choose the appropriate number of trees, we sampled a smaller tuning set of just 100 examples. Our tuning set results showed that choosing the number of trees beyond 20 did not improve performance. However, between 10 and 20 trees the performance had significant improvement. This is similar to the original observation of Natarajan et al. [8] and hence, we followed their discoveries and set the number of iterations M as 20 and maximum number of leaves L as 8. The results are shown in Table 4. For each of these user classes, we experimented with our proposed model using first-order predicates for the content-based filtering alone (denoted as CF in Table 4), as well as the first-order predicates for both content-based filtering and collaborative filtering (denoted as HR in Table 4). We also tried different settings of the parameters α and β (denoted in the parentheses following the algorithm names in Table 4)for both scenarios in order to evaluate their effectiveness on reducing the false positive prediction.

As the table shows, although the two approaches show similar performance on False Positive Rate, Precision, and Accuracy, the hybrid recommendation system improves a lot on the False Negative Rate, Recall and AUC-ROC compared with content-based filtering alone, especially on the Recall (reached 1.0 for all three of the user classes). So, question **(Q1)** can be answered affirmatively. The hybrid recommendation system improves upon the performance of content-based filtering alone, by taking into consideration the information of similar users who have the same expertise or location as the target user.

The first column of Table 4 shows the False Positive Rate which we want to reduce. As the numbers shown, the cost-sensitive approach greatly decreases the FPR compared with prior research which does not consider the domain preferences on the cost matrix. It also significantly improves the accuracy at the same time. Hence, question **(Q2)** can also be answered affirmatively. Note that, although it seems that recall has been considerably sacrificed, our goal here is not to capture all the matching jobs for the target user, but instead to increase the confidence on the recommendations we are giving to our users. Since we may have hundreds of millions of candidate jobs in the data pool, we can usually guarantee that we will have a sufficient number of recommendations even with relatively low recall. Moreover, our proposed system can satisfy various requirements on the trade-off of precision and recall for different practical consideration by tuning the parameters α and β. If one does not want the recall too low, in order to guarantee the quantity of recommendations, one can simply decrease the value of β; if one does not want the precision too low, in order to improve the customer satisfaction, one can just increase the value of β.

Fig. 4 shows a sample regression tree learned from *Case Manager* users. The left most path can be interpreted as: if we have the information on record about the *UserCity, UserSkill* and

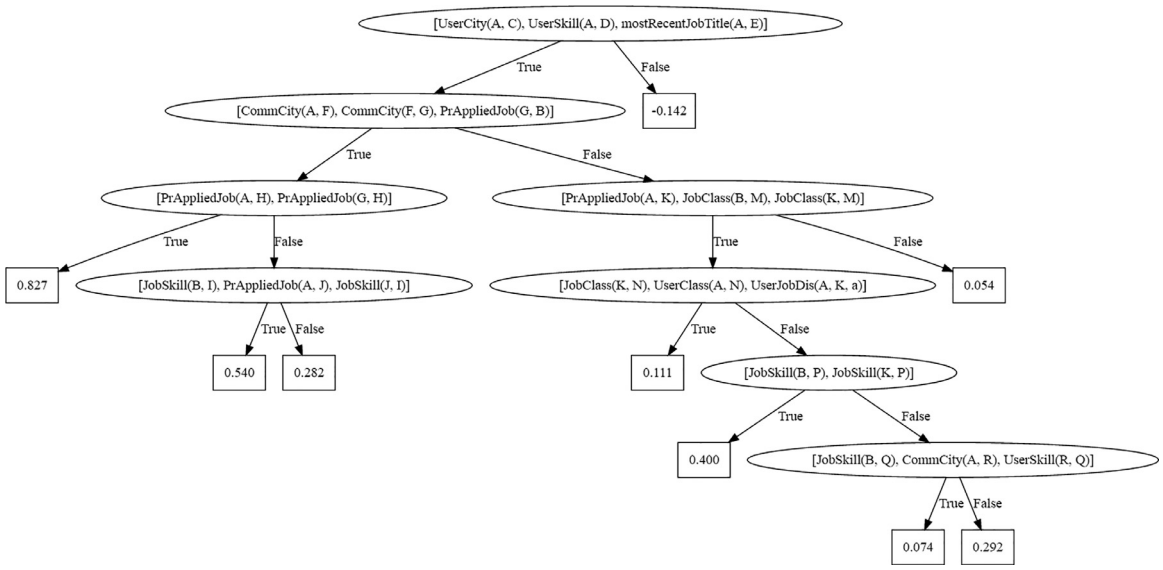

Fig. 4. Sample Regression Tree.

MostRecentJobTitle of the target user A (in other words, the target user does not suffer from the cold start problem), and there is another user G who lives in the same city and applied for the same job as user A before, and if user G applied for the target job B, then the probability for the target user A to apply to the target job B (in other words the probability for the user–job pair to be matching) is $P(Match(A, B) = 1) = \frac{1}{1+e^{-0.827}} \approx 0.696$.

It is worth mentioning that we also tried to experiment with Markov Logic Networks on the same data with *Alchemy2* [44]. However, it failed after continuously running for three months due to the large scale of our data. This underscores one of the major contributions of this research in applying statistical relational learning to building a hybrid recommendation model in a real-world large-scale job recommendation domain.

5. Conclusion

We proposed an efficient statistical relational learning approach to construct a hybrid job recommendation system which can also satisfy the unique cost requirements regarding precision and recall in this specific domain. The experiment results show the ability of our model to reduce the rate of inappropriate job recommendations. Our contribution includes: (i). we are the first to apply statistical relational learning models to a real-world large-scale job recommendation system; (ii). our proposed model has not only been proved to be the most efficient SRL learning approach, but also demonstrated its ability to further reduce false positive predictions; (iii). the experiment results reveal a promising direction for future hybrid recommendation systems– with proper utilization of first-order predicates, an SRL-model-based hybrid recommendation system can not only prevent the necessity for exhaustive feature engineering or pre-clustering, but can also provide a robust way to solve the cold-start problem.

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12–32.
[2] K. AlJadda, M. Korayem, T. Grainger, C. Russell, Crowd sourced query augmentation through semantic discovery of domain-specific jargon, in: 2014 IEEE International Conference on Big Data, IEEE, 2014, pp. 808–815.
[3] C. Basu, H. Hirsh, W. Cohen, Recommendation as classification: using social and content-based information in recommendation, in: Fifteenth National Conference on Artificial Intelligence, AAAI Press, 1998, pp. 714–720.
[4] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.
[5] J. Basilico, T. Hofmann, Unifying collaborative and content-based filtering, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004.
[6] L. Getoor, B. Taskar, Introduction to statistical relational learning, Adaptive Computation and Machine Learning, MIT Press, 2007.
[7] M. Pazzani, D. Billsus, Learning and revising user profiles: the identification ofinteresting web sites, Mach. Learn. 27 (3) (1997) 313–331.
[8] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, J. Shavlik, Gradient-based boosting for statistical relational learning: the relational dependency network case, Mach. Learn. 86 (1) (2012) 25–56.
[9] S. Yang, T. Khot, K. Kersting, G. Kunapuli, K. Hauser, S. Natarajan, Learning from imbalanced data in relational domains: A soft margin approach, in: 2014 IEEE International Conference on Data Mining, ICDM 2014, 2014, pp. 1085–1090.
[10] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.
[11] G. Salton, Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[12] J. Rocchio, Relevance Feedback in Information Retrieval, in: The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall Inc., 1971, pp. 313–323.
[13] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. in Artif. Intell. 2009 (2009) 4:2.
[14] N. Rao, H.-F. Yu, P.K. Ravikumar, I.S. Dhillon, Collaborative Filtering with Graph Information: Consistency and Scalable Methods, in: Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2107–2115.
[15] W. Wang, G. Zhang, J. Lu, Member contribution-based group recommender system, Decis. Support. Syst. 87 (2016) 80–93.
[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: An Open Architecture for Collaborative Filtering of Netnews, ACM Press, 1994, pp. 175–186.
[17] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 791–798.
[18] L. Si, R. Jin, Flexible mixture model for collaborative filtering., in: ICML, AAAI Press, 2003, pp. 704–711.
[19] N. Sahoo, P.V. Singh, T. Mukhopadhyay, A hidden Markov model for collaborative filtering, Manag. Inf. Syst. Q. 36 (4) (2012) 1329–1356.
[20] L. Getoor, M. Sahami, Using probabilistic relational models for collaborative filtering, Workshop on Web Usage Analysis and User Profiling (WEBKDD'99), 1999.
[21] J. Newton, R. Greiner, Hierarchical probabilistic relational models for collaborative filtering, in: Workshop on Statistical Relational Learning, 21st International Conference on Machine Learning, 2004.
[22] Y. Gao, H. Qi, J. Liu, D. Liu, A recommendation algorithm combining user grade-based collaborative filtering and probabilistic relational models, in: Fourth International Conference on Fuzzy Systems and Knowledge Discovery, vol.1, IEEE, 2007, pp. 67–71.

[23] Z. Huang, D.D. Zeng, H. Chen, A unified recommendation framework based on probabilistic relational models, Available at SSRN 906513 (2005).
[24] Q. Shambour, J. Lu, A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services, Int. J. Intell. Syst. 26 (9) (2011) 814–843.
[25] L.M. De Campos, J.M. Fernández-Luna, J.F. Huete, M.A. Rueda-Morales, Combining content-based and collaborative recommendations: a hybrid approach based on bayesian networks, Int. J. Approximate Reasoning 51 (7) (2010) 785–799.
[26] M. Balabanović, Y. Shoham, Fab: content-based, collaborative recommendation, Commun. ACM 40 (3) (1997) 66–72.
[27] X. Guo, H. Jerbi, M.P. O'Mahony, An analysis framework for content-based job recommendation., in: 22nd International Conference on Case-Based Reasoning (ICCBR), 2014.
[28] N.D. Almalis, G.A. Tsihrintzis, N. Karagiannis, A.D. Strati, Fodra — a new content-based job recommendation algorithm for job seeking and recruiting, in: Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on, 2015, pp. 1–7.
[29] O.M. Salazar, J.C. Jaramillo, D.A. Ovalle, J.A. Guzmán, A Case-Based Multi-Agent and Recommendation Environment to Improve the E-Recruitment Process, Springer International Publishing, Cham, pp. 389–397.
[30] E. Malherbe, M. Diaby, M. Cataldi, E. Viennet, M.A. Aufaure, Field selection for job categorization and recommendation to social network users, in: Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on, 2014, pp. 588–595.
[31] M. Diaby, E. Viennet, T. Launay, Toward the next generation of recruitment tools: an online social network-based job recommender system, in: Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on, 2013, pp. 821–828.
[32] W. Hong, S. Zheng, H. Wang, Dynamic user profile-based job recommender system, in: Computer Science Education (ICCSE), 2013 8th International Conference on, 2013, pp. 1499–1503.
[33] Y. Lu, S. El Helou, D. Gillet, A recommender system for job seeking and recruiting website, in: Proceedings of the 22Nd International Conference on World Wide Web, in: WWW '13 Companion, ACM, New York, NY, USA, 2013, pp. 963–966.
[34] A. Pacuk, P. Sankowski, K. Wegrzycki, A. Witkowski, P. Wygocki, Recsys challenge 2016: job recommendations based on preselection of offers and gradient boosting, CoRR abs/1612.00959(2016).
[35] J. Hoxha, A. Rettinger, First-order probabilistic model for hybrid recommendations, in: 12th International Conference on Machine Learning and Applications, ICMLA 2013, 2013, pp. 133–139.
[36] C. Perlich, F. Provost, Aggregation-based feature invention and relational concept classes, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '03, ACM, 2003, pp. 167–176.
[37] J. Neville, D. Jensen, Relational dependency networks, J. Mach. Learn. Res. 8 (2007) 653–692.
[38] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (2001) 1189–1232.
[39] A. Karwath, K. Kersting, N. Landwehr, Boosting relational sequence alignments, ICDM, 2008.
[40] R. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, NIPS, 2000.
[41] S. Natarajan, S. Joshi, P. Tadepalli, K. Kristian, J. Shavlik, Imitation learning in relational domains: a functional-gradient boosting approach, IJCAI, 2011.
[42] H. Blockeel, L.D. Raedt, Top-down induction of first-order logical decision trees, Artif. Intell. 101 (1998) 285–297.
[43] F. Javed, Q. Luo, M. McNair, F. Jacob, M. Zhao, T.S. Kang, Carotene: a job title classification system for the online recruitment domain, in: IEEE First International Conference on Big Data Computing Service and Applications, 2015, pp. 286–293.
[44] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, P. Domingos, The Alchemy System for Statistical Relational AI, Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, WA., 2009.