Learning to Predict Peer Review Request Declination

Abstract

We study the problem of finding appropriate reviewers who are able to complete timely reviews and would not say "no" to the review invitation. The problem is a central issue in peer review, but has received little research attention. Different from most existing studies that focus on matching given reviewers with a list of papers, we focus on a more open question: given a paper, how successfully we can predict that an expert will accept or decline a review invitation. We formalize the problem as a ranking factor graph (RankFG) model to predict the reviewer acceptance for a given paper. We also develop an interactive learning algorithm for incrementally learning the ranking function. For empirical evaluation, we developed a Chrome Extension for reviewer recommendation and deployed it in the Google Chrome Web Store. Based on the feedback/track logs from hundreds of reviewer invitations, the proposed method demonstrates its superiority (+5-10% by MAP) over several state-of-the-art ranking algorithms.

1 Introduction

Peer review is an important part of scientific publishing [Rennie, 1999; Smith, 1997; Kassirer and Campion, 1994]. Indeed, many researchers consider peer review as part of their professional responsibility. Publishing groups also use the quality of peer reviews as an indicator of the success of a journal. However, the problem has not received much research attention. Peer review has long been criticized for being ineffective, slow and low-quality. One would expect that experts with sufficient knowledge would be the best to review a given paper. However, in practice, our preliminary statistics show that nearly 50% of the review invitations have been declined (Cf. § 4 for details). Many top experts are inclined to say "no" for various reasons. As a result, finding appropriate reviewers who are able to complete timely reviews still remains a challenge.

Quite a few studies have been conducted regarding this problem. In particular, several efforts have been made for automating the conference paper-reviewer assignment using methods such as mining the web [Haym *et al.*, 1999], latent semantic indexing [Dumais and Nielsen, 1992], probabilistic topic modeling [Karimzadehgan *et al.*, 2008; Mimno and McCallum, 2007], integer linear programming [Karimzadehgan and Zhai, 2009], minimum cost flow [Hartvigsen *et al.*, 1999; Tang *et al.*, 2012] and a hybrid approach employing domain knowledge and a matching model [Sun *et al.*, 2007]. Systems have been also developed to help proposal-reviewer and paper-reviewer assignment [Hettich and Pazzani, 2006; Conry *et al.*, 2009; Mauro *et al.*, 2005]. However, most of this research focuses on the following setting: given a list of reviewers and a list of papers, how to find an optimal matching (with some constraints) between reviewers and papers. In this paper, we study a more open question: given a new paper, how to find appropriate reviewers who will *agree to review* this paper?

The problem is non-trivial. A survey of five biomedical journals shows that the most important factors for a researcher to accept a review invitation include not just the relevance of the paper topic to the researcher's interest, but also the contribution of the paper to the subject area, and whether the researcher can learn something new from the paper [Tite and Schroter, 2007]. Other reasons for the researchers to decline a review invitation include having too many reviews at hand and tight deadline for completing the review. However, technically, it is still unclear how to design an approach to help deal with the problem.

In this paper, we formalize the problem as a ranking problem and propose a ranking factor graph (RankFG) model to predict the willingness of reviewers to review a given paper. RankFG is able to identify who is at a high risk of declining a review invitation. It can also interactively adjust the ranking function according to reviewers' responses. An efficient algorithm has been developed for learning and incrementally updating the ranking function. To empirically evaluate the proposed methodologies, we developed a Chrome Extension of reviewer recommendation and deployed it in the Google Chrome Web Store. More than 20 journal editors downloaded the extension and used it for reviewer recommendations. Based on their feedback/track logs, we made several interesting discoveries. First, the proposed method can clearly better predict the declination response, in comparison with several state-of-the-art ranking algorithms. Second and more interesting, expertise matching is not very important for predicting reviewer declination.

To summarize, the contributions of this paper include:

- Formalization of the learning to predict declination of peer review requests.
- Proposal of a ranking factor graph (RankFG) model to predict reviewers' responses, and an efficient algorithm to learn the ranking function.
- Empirical validation of the proposed method on a real-world data set, and implementation of a practical tool for reviewer recommendation.

2 Problem Definition

Given a paper, our goal is find researchers who are *qualified* and *willing to review* this paper.

We consider an academic social network $G = (V, E)$, where V is a set of $|V| = N$ researchers and $E \subseteq V \times V$ is a set of relationships between researchers. There can be various kinds of relationships, for example, collaborations, same-affiliation, and same-nationality. Let $\mathbf{x}_i$ denote researcher v_i's attributes, which could be one's research interests, research experience, or simply the number of his/her publications. We use $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ to denote the attributes of all researchers. Given this, we define our problem of review declination prediction.

Problem 1 ***Review Declination Prediction.*** *Let $G = (V, E, X)$ be an attribute augmented academic social network. Given a particular paper p, the goal is to find a predictive function such that we can suggest researchers of high relevance to paper p and of low risk of declining the review invitation, i.e.,*

$$f : (G, p) \rightarrow Y \tag{1}$$

where $Y = \{y_1, \cdots, y_N\}$ represents the prediction results for all researchers in the network G and $y_k \in \{0, 1\}$ is a binary score indicating whether reviewer $v_k \in V$ has a high risk of declining the invitation.

The predictive function f takes the network G as input, which means that we should consider the network information in designing the prediction function. We also demonstrate that the network information is very useful and can help improve the prediction accuracy.

3 Approach Framework

The common approaches for finding appropriate reviewers focus on estimating the relevance of a paper to potential reviewers' research interests. This can be done by different ways of calculating the similarity between researchers' interests and the paper content. The more important problem we want to solve is how to predict who will accept/decline the review invitation after retrieving some candidate reviewers.

At a high level, the proposed approach framework consists of three stages.

- **Candidate Generation.** First, given a submitted paper p and its keywords list K_p, we extract potential reviewer candidates through content similarity.
- **Declination Predication.** Second, we present a ranking factor graph (RankFG) model, which takes the selected candidates reviewers as input and predicts who has a high risk of declining the review.
- **Interactive Learning.** Third, having collected reviewers' responses (either agree or decline), the RankFG uses an interactive learning algorithm to update its parameter configuration in the predictive function incrementally.

3.1 Candidate Generation

Given a paper p in a peer review process, we first extract a number of keywords from the paper. The extraction is done using a keyword extraction tool [Zhang *et al.*, 2006]. The authors may also provide a set of keywords. In such case, we will combine the extracted keywords and the authors provided keywords together.

Then we use a language model to retrieve relevant researchers from an online publication database. Language model is one of the state-of-the-art approaches in information retrieval. It interprets the relevance between a document and a query word as a generative probability:

$$P(w|d) = \frac{N_d}{N_d + \lambda} \cdot \frac{tf(w, d)}{N_d} + (1 - \frac{N_d}{N_d + \lambda}) \cdot \frac{tf(w, \mathbf{D})}{N_{\mathbf{D}}} \tag{2}$$

where N_d is the number of word tokens in document d, $tf(w, d)$ is the word frequency (i.e., occurrence number) of word w in d, $N_{\mathbf{D}}$ is the number of word tokens in the entire collection, and $tf(w, \mathbf{D})$ is the word frequency of word w in the collection $\mathbf{D}$. λ is the Dirichlet smoothing factor and is commonly set according to the average document length in the collection [Zhai and Lafferty, 2001]. Further, the probability of the document model d generating a query q can be defined as $P(q|d) = \Pi_{w \in q} P(w|d)$. Adapted to our setting, we replace document d in Eq. 2 with a researcher's content information (e.g., all published papers by the researcher) and q with the extracted keywords from paper p.

3.2 Declination Prediction

After retrieving candidate reviewers, we propose a ranking factor graph (RankFG) model to predict how likely the reviewer is to accept or decline the review invitation. The graphical model RankFG consists of two layers of variables: observations and latent variables. In our problem, each observation represent a paper-reviewer pair $\{(p, v_i)\}$. Each observation is associated with a latent variable y_i to represent whether the reviewer will agree or decline to review this paper. Local factor functions are defined to capture the relationships between an observation and its corresponding latent variable. Moreover, observations might be correlated with each other. For example, candidate researchers with the same nationality may say "no" at the same time. Figure 1 shows an preliminary statistics on our collected real data (Cf. § 4 for details of the data sets). We see that, on average, the probability that a reviewer will decline the invitation almost doubles if another reviewer of the same nationality has declined the invitation. In RankFG, such kind of correlations can be defined as correlation factor functions among latent variables.

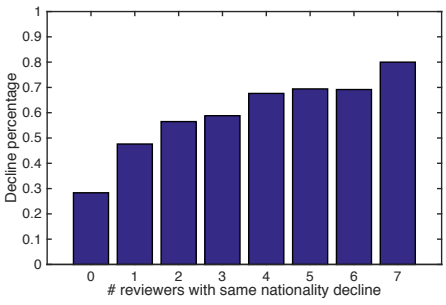

Figure 1: Declination probability conditioned on the number of reviewers of same nationality who have already declined.

More specifically, we have the following factor definitions in our problem.

Local factor function: Captures the characteristics of each paper-reviewer pair, including a relevance score between the paper and the reviewer, and any attributes associated with the reviewer. Defined as an exponential function

$$f(p, v_i, y_i) = \frac{1}{Z_a} \exp\left(\sum_k \alpha_k \psi_k(p, v_i, y_i)\right) \tag{3}$$

where $\psi_k(.)$ is the k^{th} feature function defined between p and v_i with respect to the value of y_i; α_k is the weight of the feature; Z_a is a normalization factor.

Correlation factor function: Captures the correlation between latent variables. Also defined as an exponential function

$$g(y_i, y_j) = \frac{1}{Z_b} \exp\left(\sum_l \beta_l \phi_l(y_i, y_j)\right) \tag{4}$$

where $\phi_i(.)$ is the l^{th} feature function defined between y_i and y_j; β_l is the weight of the feature; Z_b is a normalization factor.

By integrating the defined factor functions, and also following the Markov assumption [Kindermann *et al.*, 1980], we can define the following log-likelihood objective function:

$$\begin{aligned}\log P(Y|X, \theta) = &\sum_{y_i \in Y} \sum_k \alpha_k \psi_k(p, v_i, y_i) \\ &+ \sum_{v_i \sim v_j} \sum_l \beta_l \phi_l(y_i, y_j) - \log Z\end{aligned} \tag{5}$$

where $Z = Z_a Z_b$ is the normalization factor; $v_i \sim v_j$ indicates that there is a defined correlation between reviewer v_i and v_j; $\theta = (\{\alpha\}, \{\beta\})$ are parameters to estimate.

Feature Definitions

Now we introduce possible ways of defining the factor functions $\psi_k(p, v_i, y_i)$ and $\phi_l(y_i, y_j)$. In principle, the factor functions can be instantiated in different ways to reflect our prior knowledge or intuitions for different applications. It can be defined as either binary or a real-valued. In total, we define nine feature functions that can be divided into three categories: Basic statistics, Expertise matching, and Organization.

- **Basic statistics.** We define a set of statistics features for each potential reviewer, including their h-index, publication number, citation number, and length of research experience.
- **Expertise matching.** We calculate the similarity between the paper's keywords and the reviewer's research interests. Four features are defined in this category: the number of common interests, percentage of common interests, percentage of common interests among the paper's keywords, and percentage of common interets among the reviewer's interests.
- **Organization.** We define a binary feature to indicate whether the reviewer comes from academia or industry.

Two correlation factors are defined according to reviewers' attributes. The first one correlation factor represents whether two reviewers are of the same nationality, and the second correlation factor represents whether the two reviewers have the same affiliation.

Model learning

Training RankFG model involves finding a parameter configuration $\theta = (\{\alpha\}, \{\beta\})$ from a given training dataset, such that the log-likelihood objective function $L(\theta) = \log P(Y|X, \theta)$ can be maximized,

$$\theta^* = \arg\max_\theta \log P(Y|X, \theta) \tag{6}$$

This optimization problem can be solved using a gradient ascent algorithm (or a Newton-Raphson method). The gradient of each parameter θ wrt $L(\theta)$ is

$$\begin{aligned}\frac{\partial L(\theta)}{\partial \theta_i} &= \sum_j f(\mathbf{x}_i, \mathbf{y}_i) - \frac{Z(\theta)'}{Z(\theta)} \\ &= \mathbb{E}[\phi(\mathbf{x}_i, \mathbf{y}_i)] - \mathbb{E}_{P(\mathbf{x}_i, \mathbf{y}_i)}[\phi(\mathbf{x}_i, \mathbf{y}_i)]\end{aligned} \tag{7}$$

In this equation, we use ϕ to indicate both ϕ and ψ. The first term $\mathbb{E}[\phi(\mathbf{x}_i, \mathbf{y}_i)]$ in Eq. 7 is easy to calculate. However, the second term is unmanageable to estimate because of the difficulty in determining the marginal probability $P(\mathbf{x}_i, \mathbf{y}_i)$, as the graph structure may contain cycles. There are several methods to approximately solve the problem. In our work, we choose the sum-product algorithm [Kschischang *et al.*, 2001] (also known as Loop Belief Propagation (LBP) [Yedidia *et al.*, 2000]). To perform the sum-product algorithm, we first derive a factor graph from the original graph G. Factor graph model represents the factorization of a function, which means the likelihood $P(Y|X)$ in our framework. We can flexibly add factor nodes for feature functions. Then we apply the sum-product algorithm to the factor graph to compute the approximate marginal distributions. The algorithm operates message passing according to the sum-product update rule: the message sent from variable v to factor f (or g) is the product of the local function at v received from all factors other than f, and the message sent from factor f to variable v is the sum of all factor functions associated with v, i.e.,

$$\mu_{v \to f}(x_v) = \prod_{f^* \in N(v) \setminus \{f\}} \mu_{f^* \to v}(x_v) \tag{8}$$

$$\mu_{f \to v}(x_v) = \sum_{\sim x_v} f(\mathbf{x}_f) \prod_{v^* \in N(f) \setminus \{v\}} \mu_{v^* \to f}(v^*) \tag{9}$$

Algorithm 1 Learning algorithm for RankFG.

Input: Query papers $Q = \{p\}$ with corresponding keywords $\{K_p\}$, $G = (V, E, X)$, and the learning rate η;
Output: learned parameters θ;

```
θ ← 0;
repeat
    for p ∈ Q and K_p do
        L ← initialization list;
        Factor graph FG ← BuildFactorGraph(L);
        repeat
            for v_i ∈ L do
                Update the messages of v_i by Eq. 8 and 9;
            end for
        until all messages μ do not change;
        for θ_i ∈ θ do
            Calculate gradient ∇_i according to Eq. 7;
            Update θ^new = θ^old + η · ∇_i;
        end for
    end for
until converge;
```

where $N(v)$ are neighborhood factors of variable v and $N(f)$ are neighborhood variables of factor v; $\mathbf{x}_f$ is the set of arguments of $f(.)$; $\sum_{\sim x}$ means the summation over all variables expect x. Algorithm 1 describes the learning process of the RankFG model. In each iteration, messages are updated sequentially in a certain order. We randomly select a node as the root and perform a breadth-first search on the factor graph to construct a tree. We update the messages from the leaves to the root, then from the root to the leaves. The updating process continues until the convergence or the number of iterations is large enough. Based on the received messages from factors, we can calculate the marginal probabilities for each variable. Then we compute the gradient according to Eq. 7 and update the parameters by $\theta^{new} = \theta^{old} + \eta \cdot \nabla_i$, with the learning step η.

Declination Prediction

Given the observed value $\mathbf{x}$ and the learned parameters θ, the declination prediction is to find the most likely configuration of Y_p for a given paper p. This can be obtained by:

$$Y_p = \arg\max_{Y_p} P(Y_p|X, \theta) \tag{10}$$

For inference, we use the max-sum algorithm to find the values of Y_p that maximize the likelihood. This max-sum algorithm is similar to the sum-product algorithm, except for calculating the message according to max instead of sum in message passing functions Eqs. 8 and 9.

3.3 Interactive Learning

In our framework, interactive learning refers to update the ranking model base on reviewer responses. The idea is to record whether the invited reviewers accept or reject the review invitation, and then incrementally adjust parameters in the ranking model according to this feedback. The algorithm supports both online interactive update and offline complete update.

The technical challenge of interactive learning is how to update the learned RankFG model efficiently and effectively. We use an algorithm to incrementally update the parameters, which mainly solves the problem of how to calculate the gradient. According to Eq. 7, for the first term, it is easy to obtain an incremental estimation, i.e.,

$$\mathbb{E}^{new}[.] = \frac{N}{N+1}\mathbb{E}^{old}[.] + \frac{1}{N+1}\sum_k \theta_k \phi_k(\mathbf{x}_{N+1}, \mathbf{y}_{N+1})$$

where $\{\theta_k \phi_k(\mathbf{x}_{N+1}, \mathbf{y}_{N+1})\}$ denotes factor functions defined for the new learning instance with the reviewer's feedback ($y_{N+1} = 1$ for agree or 0 for decline). For the second term, it is again unmanageable to compute the marginal probabilities. Since it is very time-consuming to perform global message passing on the complete factor graph, we can approximate it by performing a local message passing. A similar idea has been previously used in [Wu *et al.*, 2013]. Specifically, we first add new factor nodes (variable and factor nodes) to the factor graph built before. Then we perform local message passing which starts from the new variable node y_{N+1} and terminates within l steps. More precisely, we take the new variable node y_{N+1} as root node, begin by calculating messages $\mu_{y_{N+1} \to f}$, and then send messages to all of its neighborhood factor nodes. The messages are propagated according to a function similar to Eqs. 8 and 9 and terminates when the path length exceeds l. After forward passing, we then perform a backward messages passing, which finally propagates all messages back to the root node y_{N+1}. In this way, based on the messages sent between variables and factors, we can calculate an approximate value of the marginal probabilities of the newly added factors/variables. Accordingly, we can estimate the second term of Eq. 7, and can further estimate the gradient.

4 Experimental Results

To empirically evaluate the proposed methodologies, we developed an Chrome Extension for reviewer recommendation and deployed it in the Google Chrome Web Store. More than 20 journal editors downloaded the app and used it for reviewer recommendations.

4.1 Experimental Setup

Datasets

We evaluate our method from two different perspectives: Relevance and Response. In Relevance, we focus on evaluating how the proposed method can find experts who are able to review the given paper, and in Response, we focus on evaluating how the proposed method can decrease reviewers' declination rate in comparison with state-of-the-art methods.

Relevance: this data set includes 86 papers and 2048 candidate reviewers. As it is often difficult to evaluate the relevance between a paper and a reviewer, we collected 86 papers, and for each paper we generated 30 reviewers as candidates (Cf. § 3.1). Finally we used human judgments as ground truth. The dataset contains 36 papers submitted to IEEE TKDE, IEEE TBD, ACM TKDD, ACM TIST, and SCIS, and 50 papers published on ACM KDD, ICML, ACL,

SIGGRAPH, and SIGCOMM. We asked two faculty members and three PhD students from the author's lab to make paper-reviewer relevance judgments. We simply considered the relevance as binary: relevance and irrelevance.

Response: this data set was collected from the Chrome extension. It includes 83 papers submitted to a number of different journals and 367 reviewers' responses (agree or decline). Among these responses, 186 are "agree", while the rest are viewed as "decline" (including "decline", "unavailable" and "no response").

On both data sets, we randomly selected 60% as training data and the other 40% as test data. We perform each experiment 10 times (including partition of the training and test data) and report the average score.

Comparison Methods

We compare the following methods for reviewer recommendation:

Content Similarity (Content): We calculate similarity between paper and reviewer based on the paper's keywords and the reviewer's research interests. The similarity score $Sim(p, r)$ between the query paper p and reviewer r is defined based on Jaccard similarity coefficient:

$$Sim(p, r) = \frac{K_p \cap I_r}{K_p \cup I_r} \tag{11}$$

where K_p is the set of paper p's keywords and I_r is the set of reviewer r's research interests (also represented as keywords). The reviewer recommendation is then made based on the rank of similarity score.

SVM-Rank: The Content Similarity method does not use training data. We considered a learning to rank approach which uses the same training data as our RankFG. As for learning model, we used the implementation of SVM-light.

RankFG: The proposed method, which trains a RankFG model to make reviewer recommendations.

Evaluation Measures

In all the evaluations, we consider the problem as a ranking problem and evaluate different methods with following performance metrics: Precision for the top-ranked N results (P@N), Mean Average Precision (MAP), and R-prec [Buckley and Voorhees, 2004; Craswell *et al.*, 2005]. In the evaluations, we only consider those researchers collected in our data sets.

All the experiment codes are implemented in C++ and Python, and the evaluation are performed on an x86-64 machine with 2.70GHz Intel Core i5 CPU and 8GB RAM. The operation system is OS X 10.11.2.

4.2 Performance Analysis

We compare the performance of all methods on the two data sets. Table 1 lists the performance of comparison methods for ranking candidate reviewers in the Relevance data set and Table 2 lists the performance of declination prediction in the Response data set.

We find that, in the Relevance data set, three comparison methods have similar performances. This is reasonable, as the relevance between paper and reviewer mostly depends on content similarity. Other features and correlations contribute little to the relevance score. On the other hand, in the Response data set, SVM-Rank performs better than Content Similarity (3.1% in terms of MAP). The proposed RankFG achieves +10.6% improvement over Content and +5.9% over SVM-Rank. RankFG leverages the correlation between reviewers, and thus improves the performance. The result also confirms the discovery reported in the survey [Tite and Schroter, 2007] that whether a reviewer agrees to review a paper depends not only on the relevance between the paper content and the reviewer's interests, but also on other factors.

Table 1: Results of reviewer ranking in Relevance (%).

Method	P@5	P@10	P@15	P@20	MAP	R-prec
Content	64.1	**57.6**	48.1	42.8	**66.8**	58.4
SVM-Rank	**64.7**	**57.6**	**48.5**	**43.4**	66.0	56.6
RankFG	60.0	57.1	47.9	43.3	65.1	**58.8**

Table 2: Results of declination prediction in Response (%).

Method	P@1	P@3	P@5	MAP	R-prec
Content	55.9	56.9	59.1	70.1	56.6
SVM-Rank	52.9	61.8	64.4	73.2	59.7
RankFG	**61.8**	**64.7**	**65.0**	**77.5**	**66.6**

Factor Contribution Analysis

We now analyze how different factors can help find reviewers. We mainly consider three factors as features: basic statistics (S), expertise matching (E), and organization (O). Here we examine the contribution of different factors by removing each factor and testing the RankFG performance. Figure 2 shows the factor contribution analysis result, using MAP score as the evaluation metric. In particular, RankFG-S represents that removing basic statistical features from our model, RankFG-E represents removing expertise matching features, and RankFG-O means removing organization features.

From Figure 2, we can see that in the Relevance data set, the performance mainly depends on expertise matching. In other word, the similarity between a paper's keywords and a reviewer's interests. Removing statistical or organization features does not significantly affect the performance. In the Response data set, the situation is very different. When we remove expertise matching features, the MAP score does not decrease, but rather increases 1.1%. It might be because in our collected data, all the reviewers are selected by journal editors, which means that all these reviewers can be considered relevant to the paper. Besides, it can be seen that the MAP score drops significantly when ignoring statistical or organization features. This confirms that our model works well when combining these features together.

Convergence Analysis

We conduct an experiment to analyze the convergent property of the RankFG model. Figure 3 shows the convergence performance of the RankFG learning algorithm. In the Relevance data set, the MAP score keeps increasing along with the number of iterations, and finally converges after 1800 itera-

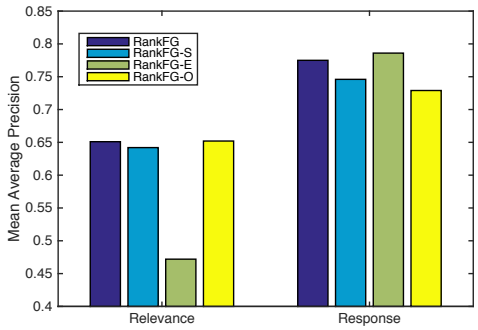

Figure 2: Factor contribution analysis: RankFG-S stands for ignoring statistics; Rank-E stands for ignoring expertise matching; Rank-O stands for ignoring organization.

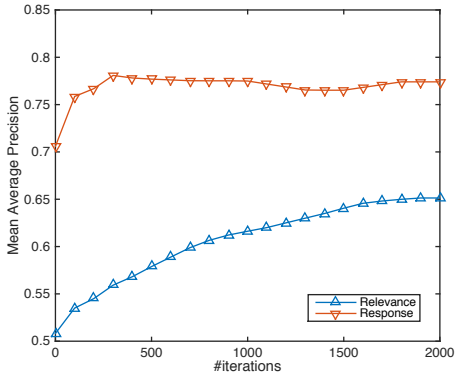

Figure 3: Convergence analysis of RankFG model.

tions. In the Response data set, the RankFG quickly reaches a relatively stable result (after only 200 iterations), though does not really converge.

Training/Test Ratio Analysis

We provide further analysis on the effect of the training ratio on our RankFG model. Figure 4 shows the experiment results when varying the percentage of training data. In the Response data set, we see a rising trend as the percentage of training data increases. This indicates the positive effect of training data size on the recommendation performance of our model.

5 Related Work

The research of matching papers with reviewers can be traced back to 20 years ago. In general, existing methods for expertise matching mainly fall into two categories: probabilistic models and optimization models.

Probabilistic models try to improve the matching accuracy between experts and papers based on different probabilistic models, such as keyword matching [Haym *et al.*, 1999], latent semantic indexing [Dumais and Nielsen, 1992], and probabilistic topic modeling [Karimzadehgan *et al.*, 2008; Mimno and McCallum, 2007]. These works are similar to expert finding problems, which always use language models or topic models.

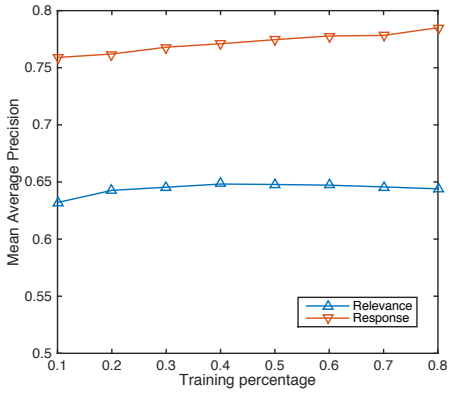

Figure 4: Training/test ratio analysis of RankFG model.

The optimization model considers the constraints in conference paper-reviewer assignment tasks. These models concentrate on solving the optimization problem of constructing panels between a list of reviewers and a list of papers, for example integer linear programming [Karimzadehgan and Zhai, 2009] and minimum cost flow [Hartvigsen *et al.*, 1999; Tang *et al.*, 2012].

Recently, a few systems have also been developed to make reviewer recommendation [Yang *et al.*, 2009], or help with conference reviewer assignment [Hettich and Pazzani, 2006; Conry *et al.*, 2009; Mauro *et al.*, 2005; Benferhat and Lang, 2001]. Wu *et al.* [2013] presents a patent partner recommendation framework in enterprise social networks, which also uses a ranking factor graph model.

In this paper, our general goal is to recommend appropriate reviewers to a given paper and predict the willingness of the reviewers to accept or decline a review invitation.

6 Conclusions

In this paper, we study a novel problem of learning to perform declination prediction in peer review requests. We formalize the problem as a ranking problem, and propose a ranking factor graph (RankFG) model to predict how likely a candidate reviewer would be to accept (or decline) a review invitation. RankFG leverages correlation between reviewers to improve prediction accuracy. We also develop an efficient algorithm to interactively train the RankFG model. It is often difficult to find a ground truth data to evaluate such task. To fairly evaluate the proposed methodologies, we developed a Chrome Extension of reviewer recommendation and deployed it in the Google Chrome Web Store. More than 20 editors downloaded the extension and used it for reviewer recommendations. Based on the feedback/track logs, we compare the proposed method with several state-of-the-art methods. Experiments show that our method can significantly improve (5-10%) the accuracy of declination prediction.

References

[Benferhat and Lang, 2001] Salem Benferhat and Jérôme Lang. Conference paper assignment. *Int. J. Intell. Syst.*, 16(10):1183–1192, 2001.

[Buckley and Voorhees, 2004] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proc. of SIGIR'04*, pages 25–32, 2004.

[Conry *et al.*, 2009] Don Conry, Yehuda Koren, and Naren Ramakrishnan. Recommender systems for the conference paper assignment problem. In *RecSys'09*, pages 357–360, 2009.

[Craswell *et al.*, 2005] Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the trec-2005 enterprise track. In *TREC 2005 Conference Notebook*, pages 199–205, 2005.

[Dumais and Nielsen, 1992] Susan T. Dumais and Jakob Nielsen. Automating the assignment of submitted manuscripts to reviewers. In *SIGIR'92*, pages 233–244, 1992.

[Hartvigsen *et al.*, 1999] David Hartvigsen, Jerry C. Wei, and Richard Czuchlewski. The conference paper-reviewer assignment problem. *Decision Sciences*, 30(3):865–876, 1999.

[Haym *et al.*, 1999] Chumki Basu Haym, Haym Hirsh, William W. Cohen, and Craig Nevill-manning. Recommending papers by mining the web. In *IJCAI'99*, pages 1–11, 1999.

[Hettich and Pazzani, 2006] Seth Hettich and Michael J. Pazzani. Mining for proposal reviewers: lessons learned at the national science foundation. In *KDD'06*, pages 862–871, 2006.

[Karimzadehgan and Zhai, 2009] Maryam Karimzadehgan and ChengXiang Zhai. Constrained multi-aspect expertise matching for committee review assignment. In *CIKM'09*, pages 1697–1700, 2009.

[Karimzadehgan *et al.*, 2008] Maryam Karimzadehgan, ChengXiang Zhai, and Geneva Belford. Multi-aspect expertise matching for review assignment. In *CIKM'08*, pages 1113–1122, 2008.

[Kassirer and Campion, 1994] Jerome P Kassirer and Edward W Campion. Peer review: crude and understudied, but indispensable. *Jama*, 272(2):96–97, 1994.

[Kindermann *et al.*, 1980] Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, 1980.

[Kschischang *et al.*, 2001] Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[Mauro *et al.*, 2005] Nicola Di Mauro, Teresa Maria Altomare Basile, and Stefano Ferilli. Grape: An expert review assignment component for scientific conference management systems. In *IEA/AIE'05*, pages 789–798, 2005.

[Mimno and McCallum, 2007] David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *SIGKDD'07*, pages 500–509, 2007.

[Rennie, 1999] Drummond Rennie. Editorial peer review: its development and rationale. *Peer review in health sciences. London: BMJ Books*, pages 3–13, 1999.

[Smith, 1997] Richard Smith. Peer review: reform or revolution? *BMJ: British Medical Journal*, 315(7111):759, 1997.

[Sun *et al.*, 2007] Yong-Hong Sun, Jian Ma, Zhi-Ping Fan, and Jun Wang. A hybrid knowledge and model approach for reviewer assignment. In *HICSS'07*, pages 47–47, 2007.

[Tang *et al.*, 2012] Wenbin Tang, Jie Tang, Tao Lei, Chenhao Tan, Bo Gao, and Tian Li. On optimization of expertise matching with various constraints. *Neurocomputing*, 76(1):71–83, 2012.

[Tite and Schroter, 2007] Leanne Tite and Sara Schroter. Why do peer reviewers decline to review? a survey. *Journal of epidemiology and community health*, 61(1):9–12, 2007.

[Wu *et al.*, 2013] Sen Wu, Jimeng Sun, and Jie Tang. Patent partner recommendation in enterprise social networks. In *WSDM'13*, pages 43–52, 2013.

[Yang *et al.*, 2009] Kai-Hsiang Yang, Tai-Liang Kuo, Hahn-Ming Lee, and Jan-Ming Ho. A reviewer recommendation system based on collaborative intelligence. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 564–567. IET, 2009.

[Yedidia *et al.*, 2000] Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. Generalized belief propagation. In *NIPS*, volume 13, pages 689–695, 2000.

[Zhai and Lafferty, 2001] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR'01*, pages 334–342, 2001.

[Zhang *et al.*, 2006] Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. Keyword extraction using support vector machine. In *WAIM'06*, pages 85–96, 2006.