Using Machine Learning to Predict Links and Improve Steiner Tree Solutions to Team Formation Problems

Peter Keane[1], Faisal Ghaffar[2] and David Malone[1]

[1] Maynooth University
peter.keane.2014@mumail.ie, David.Malone@mu.ie

[2] Innovation Exchange, IBM Ireland
faisalgh@ie.ibm.com

Abstract. The team formation problem has existed for many years in various guises. One important problem in the team formation problem is to produce small teams that have a required set of skills. We propose a framework that incorporates machine learning to predict unobserved links between collaborators, alongside improved Steiner tree problems to form small teams to cover given tasks. Our framework not only considers size of the team but also how likely are team members going to collaborate with each other. The results show that this model consistently returns smaller collaborative teams.

Keywords: team formation, link prediction, Steiner tree

1 Introduction

Online team formation and patent collaboration within enterprises are vast and widely studied fields [13, 6, 15]. Team formation is the problem of identifying a set of individuals with skills that are required by a task for its completion. One can image a setting where a stream of incoming tasks and a system in automatic fashion is finding out a number of individuals who can complete those tasks from a pool of interconnected community. Online social networks (e.g., Facebook, Linkedin, Enterprise networking tools etc.) in our personal and professional lives provide us opportunity to connect with each other and work together on common tasks. At the same time it is quite challenging to find a team of experts. The problem has been considered as NP-hard [6]. However, there have been approximate solutions to the problem and many of these rely on social or collaboration networks among individuals. The proposed solutions leverage techniques from graph theory, utilising topological features of social and business networks, combined with graph theory to examine the existing structures of teams, interactions, and collaborations within the enterprise [16, 12, 8, 11]. Others works have been depended on machine learning, to see what individual's characteristics likely to contribute in forming a link between colleagues and suggest people with these features collaborate [1, 7].

Most of the online team formation work [6, 13] focus on reducing the communicative cost among potential team members in a network. The *communicative cost* in a network is considered as a measure of how effectively team members can collaborate with each other [6]. There is a correlation between team size and cost, both communicative and financial [4, 9]. The problem that we focus on in this paper is producing small, ideally minimal teams, that have the required skills to complete a task. In producing a small team, we aim to keep costs down.

In this paper, we propose a framework which incorporates both individual's attributes as well as topological features from individual's network into machine learning link prediction task to improve the enhanced Steiner algorithm for team formation proposed in [6]. Our aim is to use machine learning to produce an augmented graph, containing both real and predicted links, before feeding it into the Enhanced Steiner Tree Algorithm [6], in order to return a minimal (weight) team which covers the necessary skill set. In doing so, we are aiming to combine previous work both on link prediction using machine learning [1] and work carried out using minimal spanning tree solutions [6]. We hope that by combining the two approaches we will overcome possible shortcomings of same. The Enhanced Steiner Tree algorithm provides good solutions to the team formation problem, but one possible shortcoming we point to is that it may sometimes neglect to consider isolated nodes or nodes that sit on unconnected components of the network.

We propose that, sometimes, it may be better to recommend collaboration between two inventors who aren't closely connected. There may be a valuable inventor who has all the required skills, and has parameters that are conducive to a good collaborative relationship with at least one member of the potential team. It may be more prudent to recommend this person to the team instead of bringing in many more inventors to cover the skills that this one inventor has. This will help in terms of finding small teams, and keep cost down.

To evaluate our scheme, we will show how we implemented it on a collaboration graph consisting of of IBM Patent inventors. Our data is drawn from the US Patent Office (USPTO) data set. We evaluate the scheme in terms of the team sizes produced.

The paper is structured as follows. In Section 2 we introduce notation and discuss other work carried out in the area of team formation. Section 3 will give a high level outline of the model. A detailed explanation will be provided in Section 4 along with the method for evaluating our model. Finally, Sections 5 will show our results. The conclusions are presented in Section 6.

2 Preliminaries

2.1 Mathematical Definitions and Notations

Some definitions will be laid out in this section. Full details of these definitions can be found in "Modern Graph Theory [2]" by Bollobás.

For the purpose of this paper, $G = (V, E)$, will denote a weighted graph, with a node/vertex set V and edge set E. In our case, the graph will be a collaboration graph where the nodes will represent patent inventors and the edges collaborations. The weights, w_e, are assigned to each edge and represent the strength of the collaborations, where the higher the weight, the less strong the collaboration is. We will need to work with an induced subgraph of G based on the skills required for the task. This is a subgraph $G'(V', E')$ where $V' \subset V$ and $E' \subset E$ are the edges with both nodes in V'.

Steiner Tree problems, named after Jakob Steiner, are a combinatorial optimisation problem. They typically marry the problems of finding minimal spanning trees with the shortest path problem. For example, in [5] it is used to find "*a shortest network which spans a given set of points*."

As input to the machine learning part of our scheme, we will use a number of simple graph theory quantities, such as clustering coefficient [14].

2.2 Related Work

We are motivated by forming small collaborative teams. In [9] the authors observe that smaller teams lead to lower communicative and co-ordination costs. They also explicitly state that *"Most of the software development cost is related to the programmers salaries"*. To this end, it seems prudent to recommend the smallest possible teams to cover necessary skills required for given tasks.

One of the primary algorithms we will rely on in this paper is the Enhanced Steiner Tree algorithm[6]. We also use their problem definition for the *Team Formation Problem*. That is to say, given a task T, individuals V with skills, and a Graph G representing the network between those individuals, we aim to return a minimal team which contains individuals that not only cover the skills required for T, but can also work effectively together.

We will also draw on work on bonding and bridging measurements as social capital [10] and collaborative ratio measurement [15]. These metrics are as inputs to our machine learning scheme in Section 4.3.

3 Proposed Framework

The steps involved in our proposed model are outlined in the high level block diagram shown in Figure 1.

The scheme begins with *Raw Data*, where all of the raw patent data downloaded and summarised. We identified the number of inventors, to assess our graph size, and also how many different patent classes there, as these will represent our skills and determine the induced subgraphs.

At the *Pre-Processing* stage, we selected the subset of patents that we were interested in working with, the IBM patents, using the assignee information.

We next generate the *Skills & Inventor Network*. We consider the the distribution of skills, (i.e. patent classes) across inventors, and created a cross-domain graph where each node represents an inventor, and an edge between them represents collaboration between inventors from different domains. We can also extract a subgraph where certain inventors have expertise in certain skills.

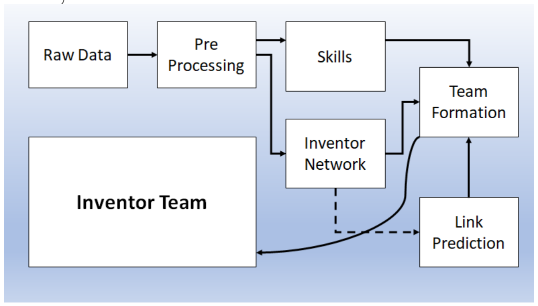

Fig. 1: Block Diagram of Proposed Model

Link Prediction is the stage where machine learning is used to predict links based on certain parameters. At this point, the original inventor network is augmented with the false positives from the machine learning model.

Finally, the *Enhanced Steiner Tree* [6] is the algorithm that we use to form a small team which possesses the necessary skills to cover the skill requirement.

4 Scheme Implementation

In this section, we will lay out the steps involved in producing the graphs used for analysis, and the CSVs used for machine learning purposes.

4.1 The USPTO Database

Data was downloaded in TSV format from from the USPTO PatentsView website[3], specifically the *patent, ipcr, patent_inventor* tables and tables relating to company assignments.

Using the company assignee data for IBM, all the IBM patents were extracted. They were cross referenced with the other other tables and Python's pandas package was used to merge the relevant data to produce a final working table which consisted of the following headings: *patent_id, inventor_id, full_class_id*. The full *full_class_id* used is the International Patent Classification[4].

4.2 Graph Creation

The main graph is our collaboration graph $G = (V, E)$, where V contains the inventors of IBM patents, and E contains any collaboration between the inventors on a patent. The extracted patent collaboration graph has 39,199 nodes and 158,279 edges with average degree of 8 and clustering coefficient of 0.4665. The weights assigned to each edge correspond to the strength of the collaboration and it is calculated as inverse of the number of co-inventions to indicate the cost between two inventors.

[3] http://www.patentsview.org/download/
[4] https://www.wipo.int/classifications/ipc/en/

```
Result: A Graph G(V,E) where V is a patent holder, and E is collaboration
        between patent holders
Initialisation: Group the dataframe by unique patent ids, resulting in a
  Group with a title of patent id, and within the group are all of the inventor ids
  associated with that patent;
for Each Group do
    if Patent ID Group Has Only One Unique Inventor then
        Add node with Inventor ID;
    else
        for Each Pair of Inventors do
            if Pair is Not Connected then
                Create nodes for inventors and add edge of weight 1 between
                  Inventors;
            else
                Add 1 to current edge weight;
Invert all weight values;
```

Algorithm 1: Creating Collaboration Network

This graph was created from the table generated by Algorithm 1. Python's pandas package was used for grouping the data. As noted above, patent classes are used to represent the skills required on a team. A very small scale example of such a graph is shown in 2, where the weight is the cost of traversing the edge.

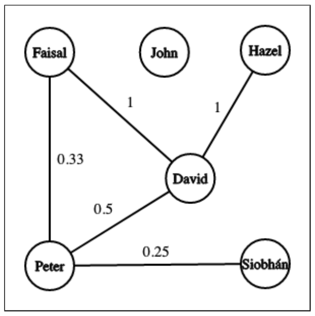

Fig. 2: How a Graph Created by Algorithm 1 Would Look

The initial graph has 39,199 nodes and 158,279 edges. As such it would be computationally expensive to run many measurements on it. Consequently, the graph was reduced by selecting specific patent class ids and inducing the subgraph on all nodes/inventors holding patents in those classes.

One thing to note is that our graph includes all patents from the relevant database, which runs from 1976-2011. Some links are temporal, and a future user can just limit the graph they create to currently active inventors by using a date of last patents as a cutoff point.

4.3 Machine learning model training

From analysis of the collaboration graph G another dataset was created for the purpose of training a machine learning model. The aim is to produce a set of features that might predict if two inventors might collaborate, even if that is not reflected in the patent data set.

From here the machine learning dataset was created with the following headings: *vi, vq, bonding_vi, bridging_vi, bonding_vq, bridging_vq, vi_collab_ratio, vq_collab_ratio, vi_cluster, vq_cluster, vi_patents, vq_patents, common_neighbors, expert_jaccard, resource_allocation, connected.* We will explain each of these in turn. For each pair of inventors, vi and vq refer to each inventor of the pair. Some of the measurements relate to each inventor individually, and these are denoted by having vi or vq in their name.

The bonding and bridging capital [10] of each inventor was calculated by getting the communities of the graph, using the best partition function in networkx's community module, and seeing how many of the neighbors of each inventor are in the same community. The bonding capital is given by:

$$\text{Bonding Capital} = \frac{|\text{Neighbors in same community}|}{|\text{Total number of neighbors}|}. \tag{1}$$

Conversely, the bridging capital is given by:

$$\text{Bridging Capital} = \frac{|\text{Neighbors } \textbf{not} \text{ in same community}|}{|\text{total number of neighbors}|}. \tag{2}$$

The collaborative ratio [15] of each inventor is given by:

$$\text{Collaborative Ratio} = \frac{|\text{Collaborative patents of inventor}|}{|\text{Patents of inventor}|}, \tag{3}$$

where a *collaborative patent* is any patent which has more than one inventor.

The values vi_cluster and vq_cluster are simply the clustering co-efficient of each inventor. Likewise, vi_patents and vq_patents are simply the total number of patents held by each inventor and common_neighbors is self explanatory.

The Jaccard Index of two sets is given by:

$$J(A,B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid}. \tag{4}$$

To find the expert_jaccard between two inventors, we take the Jaccard Index of the set of patent classes in which each inventor held expertise. We define expertise by an inventor being in the top quartile of inventors in a patent class, as measured by number of patents held in that class.

Resource allocation is the *resource allocation index* of the pair of nodes as given by Python's networkx module. Finally, the connected value is 1 if there is an edge between vi and vq in the collaborator graph, and 0 if there is no edge.

We now have a dataset for a machine learning algorithm which will attempt to predict if two nodes are connected based on the features above.

4.4 Machine Learning Model

We now use Python's xgboost module to create a model, with the *connected* column providing the binary classification target for prediction.

As there is no link between most pairs of inventors, the connection column exhibits substantial skew that might impact predictions. To counteract this, SMOTE oversampling [3] was employed to ensure a more accurate model was trained. Following the training of this model, a second graph split between random domains was created as before, and a dataset of the same format was created. The model was tested on this new, unseen, dataset and the predicted *connected* column was compared against the actual one. From this, the accuracy of the model was measured.

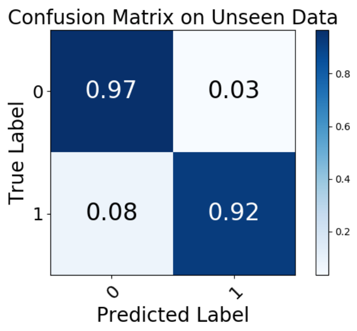

Fig. 3: Confusion Matrix of Machine Learning Model on Unseen Data

Figure 3 shows that the model provides good accuracy across all classifications. The model is predicting ones, or there being an edge between inventors, with a 92% accuracy. Conversely, it is incorrectly predicting that there is a zero, or no edge between inventors, 8% of the time. The main area of interest for us is the top right quadrant. Our model predicts that there is no edge between inventors, 97% of the time, and predicting incorrectly that there is an edge between inventors just 3% of the time. This 3% of *false positives* will be what we use to create additional links on our graph.

4.5 Creating New Links

Having built a model to predict links, we are now ready to introduce the graph that is augmented with extra edges that are predicted by that model. This is to overcome a limitation of using the Enhanced Steiner Tree Algorithm [6] directly on the collaboration graph, in that it will not consider collaborations that sit in unconnected components.

As such, we propose that artificial links are created on the graph between pairs of inventors for which our machine learning model incorrectly predicted

```
Result: An Augmented Graph with Machine Learning False Positive Edges
        added
Initialisation: Gather all pairs of nodes between which there was an edge
  predicted by the machine learning model but no edge in the collaboration
  graph;
for Each Pair of Nodes do
  | Add edge between them;
```

Algorithm 2: Creating Machine Learning Augmented Graph

there was a link, or the 3% as seen in the top right quadrant of Figure 3. The logic here being that the machine learning model predicted, based on the parameters discussed in Section 4.3, that these two inventors had attributes that were considered to be conducive to a collaborative relationship. This procedure is shown in Algorithm 2

The graph was augmented by creating an edge between every pair of inventors for which the model predicted an edge, but there didn't already exist such an edge. We also need to provide a weight for the edge. This was was set as $1 - P$ where P was the probability of an edge existing as predicted by the model. This weight is applied to all edges, including those included in the original collaboration graph.

This provides us with our augmented graph on which we will run the Enhanced Steiner Tree algorithm to identify a team.

4.6 Testing

Given the relation between team size and cost, the main criterion for testing was the team size necessary for the given skills.

This raised the obvious point that, having added in edges predicted by the machine learning model, the cardinality of the team would almost always be smaller, as the graph is now better connected. In fact, even if we added edges at random, we would expect teams to be smaller. We take advantage of this intuition, and use teams generated from a randomly augmented graph as a comparison. To generate this randomly augmented graph, we add edges at random between pairs of nodes until the number of edges of this randomly augmented graph equals the number of edges of the machine learning augmented graph.

In the evaluation we will compare the average team size required for different skillsets (i.e. sets of required skills). Since there are many skillsets, we compare the average team size as a function of the number of skills.

Specifically, a task T is a set of patent classes that could be interpreted as the skillset required for the task at hand. For this task, the the Enhanced Steiner Tree algorithm [6] can be run on (1) the original graph, (2) the machine learning augmented graph, and (3) the randomly augmented graph.

Task	G	G Random-Augmented	G ML-Augmented
T_1	13.89	7.97	**6.88**
T_2	9.69	7.92	**6.41**
T_3	7.86	5.36	**4.56**
T_4	5.76	5.69	**3.83**

Table 1: Average Cardinality of Teams for Task of size 10

5 Results

This evaluation was initially performed by selecting 10 patent classes at random. The algorithm was run 100 times, and the average team size recorded. This was repeated for four different random tasks, T, of size 10.

Table 1 shows the results for these four initial tests performed on differing tasks T, all of size 10. Our augmented graph returns a smaller team than the original collaboration graph, as expected, but also consistently returns a smaller team than the graph with randomly created links, even though the random graph and augmented graph both have the same number of edges.

Following that, a random T was selected as before, but with a size of 8. The same tests were again run, in this case 50 times, and the average number of members in each team recorded. T was increased by 1 random element after each iteration until the size of T was 24 patent classes.

The results displayed in Figure 4 show the outcome of the tests for differing numbers of patent classes in task T. The plot shows the number of elements in a task T on the x-axis, and the average number team members returned having run the Enhanced Steiner Tree algorithm 50 times on all three graphs.

The average cardinality of the team returned for the augmented graph is lower than that of the original collaboration graph, intuitively and as we discussed in Section 4.6. More interestingly, the average cardinality of the team returned when the Enhanced Steiner Tree algorithm is run on the machine learning augmented graph is consistently lower than the graph which was augmented with random links, even though the random graph has the same number of edges as the machine learning augmented graph.

This result is replicated for nine other random tasks T as shown in Figure 5. In this table, each graph follows the same legend and has the same axes as the graph shown in Figure 4, although some of the y axis scales differ. This shows that the machine learning augmented graph consistently returns a smaller team, than that returned from the randomly augmented graph and the regular graph, which covers the skillset across differing tasks of differing sizes.

While the team size is an important factor, one might also want to consider the communicative cost. One crude way to represent this would be the sum of weights of the links used by the team. The results of this are shown in Figure 6. Interestingly, though the average cardinality of the team suggested by the machine learning graph is lower, the sum of weights tends to be higher at larger

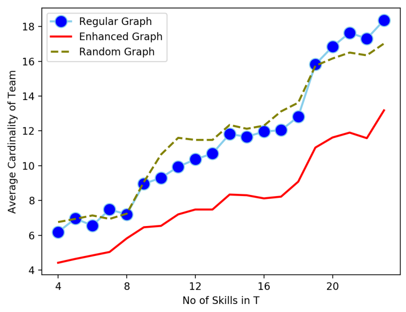

Fig. 4: Average Cardinality of Team V Number of Skills in task, T

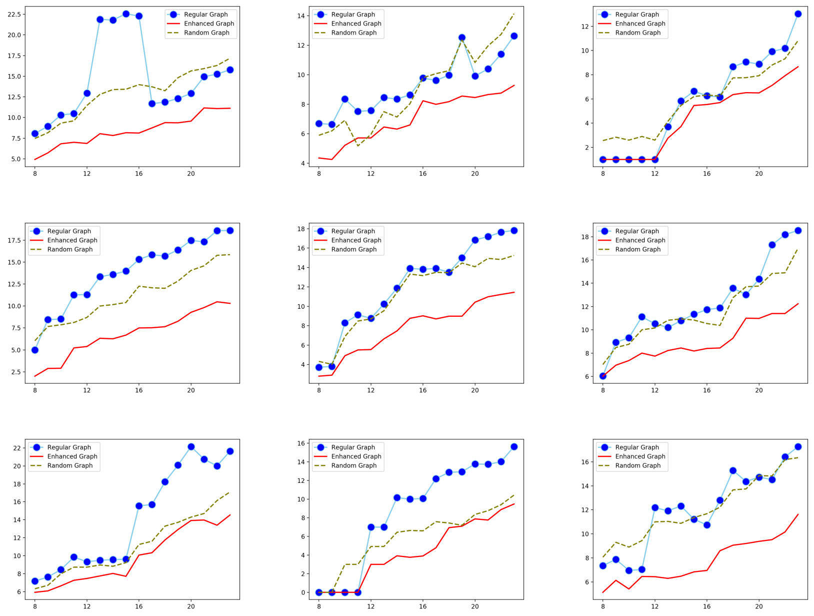

Fig. 5: Average Cardinality of Teams for Differing Task Sizes

numbers of skills. The physical or financial communicative cost is, of course, determined by the circumstances. Consequently, it may be interesting to explore other, more meaningful measurements of the communicative cost.

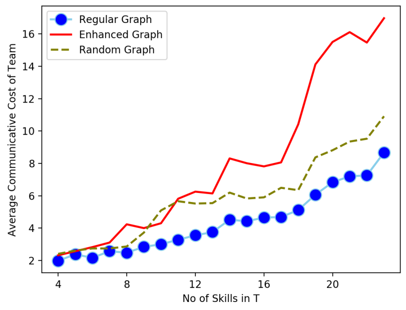

Fig. 6: Communicative Cost based on Machine Learning Probabilities

6 Conclusion

We aimed to provide a method to find small teams that cover necessary skillsets using a combination of graph-based techniques and machine learning. We built upon the Enhanced Steiner Tree algorithm, which provides a good solution to the team formation problem, but can sometimes neglect to include inventors which may not be well connected in the network. We use machine learning to predict links to connect these inventors based on potential relationships.

Our results show that when we add these predicted links we consistently get a smaller team which covers the skills required for a given task T, compared both to the original collaboration graph and a randomly augmented graph. This suggests our augmented graph produces compact teams with good collaborative potential.

A substantial area for future research is the communicative cost of the teams. There may be a more direct method available to users for measuring potential communicative cost. This could be traded off against the cost of additional team members in an expanded model. Another area to be considered is how to measure the efficacy of a team, pre-existing or newly formed by this method. This could give a whole new dimension on the team formation problem.

Future work may also consist of including the temporal aspects of the links in the team finding algorithm, instead of the graph fed into the algorithm. We may also explore using bipartite networks, where one set of nodes is the expertise, and the other is the inventors. It is worth noting, that the enhance line of the Enhanced Steiner Tree algorithm does create additional nodes based on skills, and connects them to inventors who possess those skills, but it is not a true bipartite graph, as the inventors are still connected directly.

Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

References

1. Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.
2. Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.
3. Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
4. Narasimhaiah Gorla and Yan Wah Lam. Who should work with whom?: building effective software project teams. *Communications of the ACM*, 47(6):79–82, 2004.
5. FK Hwang, DS Richards, and P Winter. *The Steiner Tree Problem*, volume 53. Elsevier, 1992.
6. Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 467–476. ACM, 2009.
7. Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.
8. Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.
9. Parag C Pendharkar and James A Rodger. The relationship between software development team size and software development cost. *Communications of the ACM*, 52(1):141–144, 2009.
10. Rajesh Sharma, Kevin McAreavey, Jun Hong, and Faisal Ghaffar. Individual-level social capital in weighted and attributed social networks. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1032–1037. IEEE, 2018.
11. Gabriel Spadon, Andre CPLF de Carvalho, Jose F Rodrigues-Jr, and Luiz GA Alves. Reconstructing commuters network using machine learning and urban indicators. *Scientific reports*, 9(1):1–13, 2019.
12. Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1285–1293. ACM, 2012.
13. Xinyu Wang, Zhou Zhao, and Wilfred Ng. A comparative study of team formation in social networks. In *International conference on database systems for advanced applications*, pages 389–404. Springer, 2015.
14. Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440, 1998.
15. Sen Wu, Jimeng Sun, and Jie Tang. Patent partner recommendation in enterprise social networks. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 43–52. ACM, 2013.
16. Jiawei Zhang, Yuanhua Lv, and Philip Yu. Enterprise social link recommendation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 841–850. ACM, 2015.