Client Network: An Interactive Model for Predicting New Clients

Massimiliano Mattetti[1], Akihiro Kishimoto[1], Adi Botea[1], Elizabeth Daly[1], Inge Vejsbjerg[1], Bei Chen[1], and Öznur Alkan[1]

[1] IBM Research - Ireland
massimiliano.mattetti@ibm.com
[2] {akihirok,adibotea,elizabeth.daly,ingevejs,oalkan2,beichen2}@ie.ibm.com

Abstract. Understanding prospective clients becomes increasingly important as companies aim to enlarge their market bases. Traditional approaches typically treat each client in isolation, either studying its interactions or similarities with existing clients. We propose the Client Network, which considers the entire client ecosystem to predict the success of sale pitches for targeted clients by complex network analysis. It combines a novel ranking algorithm with data visualization and navigation. Based on historical interaction data between companies and clients, the Client Network leverages organizational connectivity to locate the optimal paths to prospective clients. The user interface supports exploring the client ecosystem and performing sales-essential tasks. Our experiments and user interviews demonstrate the effectiveness of the Client Network and its success in supporting sellers' day-to-day tasks.

Keywords: Link prediction; Graph visualization; User Interfaces

1 Introduction

Identifying new potential clients is an important and time-consuming task for many organizations. Sellers have to prospect for new clients and figure out which companies might be more interested in becoming a client. Sellers may also have to follow up on marketing responses where potential clients may have previously expressed an interest in a product by downloading a white paper or asking for additional information as a follow up to a marketing campaign. With limited time and large numbers of follow-ups to pursue, sellers may need assistance in prioritizing which leads represent likely conversions. Typical approaches to client prospecting use the notion of similarity to identify clients who have shared attributes to the organization's existing clients. Prioritization methods tend to filter out low likelihood interactions or identify spam-like behavior to focus on interactions that express the most interest. These solutions evaluate the organization or interactions in isolation from the existing client relationships. Knowledge and trust of a brand is something that is built up over time with clients and can play an important role in a client's decision to purchase from a company in the future. This influence does not need to be limited to the existing clients, given that the business world is a small world where key professionals move between organizations. When a person takes on a new role in an organization they do not just start afresh, they also bring with

them their knowledge, experiences and relationships. This can tell us something about the companies that they are moving between: it could be that either they are in a similar industry space or have a similar corporate culture. We can harness the fact that they may bring with them knowledge and experience of products and services used in their prior organization. If we can find these companies, they can represent valuable prospective clients for an organization to approach.

In this paper we explore the research question of how to leverage latent information in the relationship graph to predict prospective clients and also surface the relationships to sellers, to provide tangible explanations and give context for the predictions. We present our solution, Client Network, which uses organizational connectivity to build a network with different types of links with different meanings and different weights. It combines a novel ranking algorithm with data visualization to allow the sellers to understand the client ecosystem better. We tuned the algorithm by analyzing our connectivity with existing clients assuming the client link is missing in order to be able to use this ranking to predict the utility of the prospective client.

Our contributions in this paper are as follows: A scalable network algorithm that supports heterogenous network relationships to rank prospective clients in order to allow sellers to focus on leads with the highest probability of success; and an interactive visualisation tool which supports exploration and provides context supporting the predictions allowing them to interpret the ranking and better inform the seller.

2 Background

Given a network of companies connected together via multiple relationships, the problem of recommending prospective clients can be formulated as predicting whether the link *client* appears in the network or not. Such problem is a variant of the standard *link prediction problem* which aims to predict the formation of any type of relationships in a network. It has extensively been studied in various fields, such as web science [1, 36], healthcare and biology [17, 5, 4, 12], and recommender systems [3, 23, 32, 20, 15].

Formally, the link prediction task can be formulated as follows [20]. Given a network $G(V,E)$, let edge $e = (u,v) \in E$ represent an interaction between nodes $u,v \in V$ at time $t(e)$. The multiple interactions between u and v are recorded as parallel edges. For $t \leq t'$, let $G[t,t']$ denote the subgraph of G restricted to the edges between t and t'. For *training interval* $[t_0,t_0']$, the link prediction task is to predict a list of edges occur in the network $G[t_1,t_1']$, where $t_1 > t_0'$.

Various techniques exist for link prediction which differs in model complexity, prediction performance, scalability, and generalization ability (see [30, 22, 14] for an overview). Topology-based link prediction methods [30] are unsupervised approaches that leverage the *latent information* contained in the network topology to assign a score to each pair of nodes. Algorithms in this category are generally divided into *local* and *global* similarity-based approaches [22]. Local approaches, such as Common Neighbors, the Jaccard Coefficient and the Adamic-Adar Coefficient [2], use node neighbourhood-related structural information to compute similarity among nodes. They are simple and fast to compute [22]. However, their performance in non-small-networks [20], where links can form between nodes at distances greater than two, is poor. Global ap-

proaches, on the other hand, use the whole network topological information to score each link. They have strong predictive power but at higher computational cost. Path-based algorithms, such as the Katz index [19], and algorithms based on random walks, such as PageRank [24], SimRank [16], Hitting Time [20] and PropFlow [21], fall under the umbrella of global topology-based approaches.

A wide range of real-world systems can be modelled as *heterogeneous information networks* (HIN), such as human social activities, communication and computer systems, and biological networks. Formally a HIN is defined as a network of multiple types of nodes (e.g., authors, conferences and papers) and multiple types of edges (e.g., co-author, author-write-paper and paper-published-in-conference) [27].

Initial approaches to the link prediction problem in HIN utilized the same algorithms used in the homogeneous ones, with no changes. These algorithms were not designed to take into account the dependency patterns across types that exist in heterogeneous networks. Such approaches treat all relationships equally or separately study homogeneous projections of the networks, completely ignoring information about the different topology or the different formation mechanism that each type of edge may have. Later approaches introduced new extensions to classical link prediction algorithms [9, 33] which improved their performance in HIN. More recently, a meta-structure known as a *meta path* [28] has been proposed in order to account for the semantic of the different types of relationships. Furthermore, a new category of algorithms has arisen which leverages the concept of meta path [26, 25].

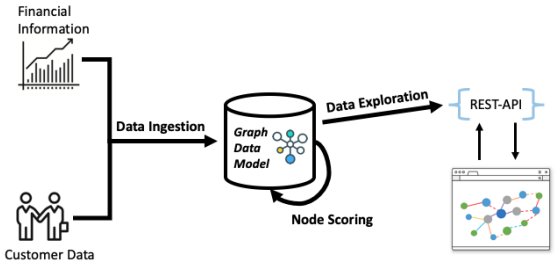

Fig. 1. Application Workflow

3 The Proposed Solution: Client Network

Our solution, Client Network, aims at supporting sellers to identify prospective clients. It leverages past interactions that took place within an ecosystem of people and organizations to build a network connecting the seller's company to any other company in the ecosystem. It employs a novel ranking algorithm for computing the likelihood of turning a company into a new client and enables the seller to explore the network with an interactive UI. Hereafter we refer to the seller's company with the term *root*.

Data Model. The Client Network can work with any type of network providing it has the following characteristics: i) a subset of nodes represent companies; ii) one of the nodes of type company is marked as *root*; iii) a relationship of type *client* exists between the root node and other nodes of type company.

Thus, the Client Network is a generic solution, with no additional assumptions about the types of nodes and the types of relations, for example.

However, for clarity, we use details from an actual network in the presentation. This is also used as input to our system in the evaluation. It is a network that combines information about the clients of our organization with financial information about banks, corporations, investment managers and more. Being confidential data, we are not able to publish the dataset used in the evaluation. The network contains two types of nodes, company and person. The latter is for professionals with decisional power in their companies. Each job role of a professional, either current or former, is represented as an edge between the corresponding person node and company node.

Job roles are divided into two types: *board member* and *executive*. We further attach to job-role edges a label such as *current* or *former*. For instance, given a company and its current CTO, the corresponding company and person nodes are connected with an edge labelled as *current executive*. Likewise, the node of a professional that has served in the company's board in the past is connected to the company's node through an edge labelled as *former board member*.

In addition to the connections capturing the organizational people flow, the network contains *business to business* (B2B) relationships, such as *sponsor*, *subsidiary* and *investor*, which can be in different states, such as *pending*, *cancelled* and *prior*.

Despite being a B2B relationship, the *client* relationship differs from the others because it does not have a state. In other words, a company is either a current client of the root company or not. Former clients[3] are equivalent to non-client companies from the seller's point of view. Note that a client relationship always involves the root, being an edge between the root and the client at hand. It is worth mentioning that there is a significant imbalance in the distribution of relationships and nodes. For example, of the 11.5 million nodes in the network, two-thirds are organizations whereas the remaining third are professionals. Furthermore, within the organizations there is a ratio of 1 to 14 between the client and non-client nodes, respectively.

Finally, multiple relationships can exist between two entities, that is, the Client Network allows multiple edges between a pair of nodes.

Solution Overview. The Client Network includes two main components: the *Analyzer* and the *Interactive Visualizer*.

The Analyzer extracts firmographic data and biographical information about professionals from relational data sources, converts the data into a graph data model and finally loads it into a database. Once the structure of the network is ready, our NORA algorithm, described in Section 3, assigns a score to each node in the network. These scores are then stored as attributes of the node objects in the database.

The Interactive Visualizer allows users to interact with the network. Possible interactions include exploring the neighborhood of a node, retrieving the subgraph con-

[3] In this work, a former client is a company that has not purchased any product/service produced/provided by the root company in the last 5 years.

necting a node to the root node, and ranking a list of companies, given as input, using NORA. These functionalities are backed by a set of REST APIs which have been designed to support any sales application in exploring the network data, therefore allowing an easier integration of our model with existing sellers' tools.

The client ecosystem is a very dynamic environment. A new client acquisition, an individual who moves to a different company and a company that invests in another company are all examples of events that contribute to the evolution of the network. The Client Network requires an up-to-date snapshot of the network in order to provide a high prediction accuracy. Hence, a full import of the data, as well as a new scoring of the nodes, have to be performed periodically. Figure 1 describes the full workflow of our solution, where data ingestion and node scoring tasks are performed by the Analyzer and data exploration is enabled with the Interactive Visualizer.

User interface. An API and UI have been developed to support navigating through the complex network of companies and individuals. Our aim was to facilitate the exploration of the network context while maintaining a task-oriented focus. Users can discover prospective clients with connections to the root company, explore each client's relationships and receive a measure of the success chances of a sales pitch to that client. We envisioned that the Client Network would help sellers in several important tasks, such as: following up on a list of leads from marketing; exploring the connections of a recently acquired client; and finding out more about a social media interaction.

Initial discussions with domain experts revealed that one concern for the root organization was the amount of time that the sellers spend prospecting for *whitespace*[4] companies using their existing tools like LinkedIn. Using the Client Network will allow sellers to save time when deciding which marketing leads they should prioritize, as well as to make discoveries which could help them shape their approach to the client and increase the likelihood of a successful sales opportunity outcome. The seller can get a feeling for how the individual company fits into a larger and connected client ecosystem relating back to their company.

The user interface is composed of a general landing and information area, a search interface to find companies, company ranking list views and a view to explore a company and its connections in more detail. The user interface consists of a React.js web application which communicates with a Java API that supplies it with the data to power the search and the visualizations. The UI is built responsively, so that it will respond to screen size and show appropriate styling and information depending on the device screen size. The network graph is visualized using the vis.js library [7] and it uses a force-directed layout, with controls for zooming and resetting the graph.

The search functionality allows the user to search for a company name. On searching for a company, the user is presented with a list of results which shows company features such as the company name, ranking, probability of success, status, location, and year founded, allowing the user to disambiguate between similarly named companies. Once the targeted company is identified, the user can click through to access a detailed view of that company, or they can add them to a ranking list.

Sellers receive lists of prospective clients from the marketing team or their managers. Deciding which of these clients is a likely prospect and should be approached as

[4] Term used internally by sellers to refer to prospective clients.

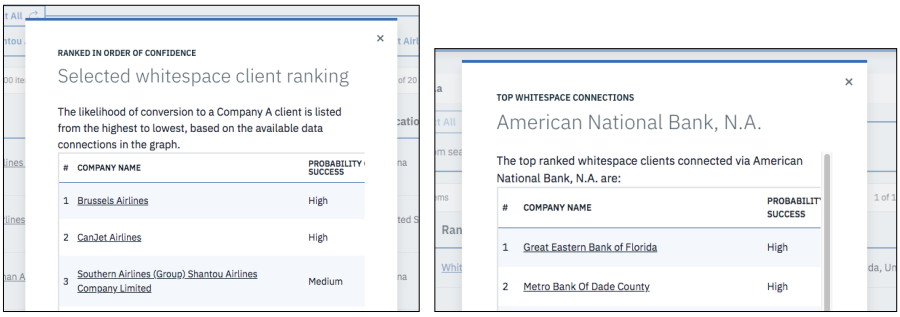

Fig. 2. Left: Company ranking. Right: Top Whitespace Companies

a priority can be time-consuming. The seller can input a list of companies and the Client Network will return the list ranked by the confidence level of turning a company into a new client. The seller searches for each company in turn and adds them to the list. The system processes this list, returning a modal window with the list of companies. The list is ranked, showing companies assumed to have a higher chance of success of becoming a future client first, as presented in Figure 2 left.

Depending on factors such as the product they are selling or the country they are targeting, some sellers may receive fewer leads than others. In the case of a seller struggling to find prospective clients, a good starting point would be to explore the connections of a recently acquired client. An existing client can be used as a starting-off-point to prospect for whitespace clients that are connected to it. Thus, the Client Network turns a newly acquired client into an opportunity for approaching whitespace clients which were not taken into consideration previously due to a low connectivity to the root company. The seller searches for the company and can further investigate the "Whitespace connections". A ranked list of whitespace clients that are connected to the newly acquired client is returned to the seller (Figure 2 right) allowing them to quickly identify companies in that ecosystem that have a high chance of conversion to a client.

In the third scenario, the seller can find information about the positioning of the company within the client ecosystem and use company descriptions and professional biographies to find out more. The seller can select a company to see a detailed view of that company and its associated subgraph, which visualizes how it links back to the root company, as shown in Figure 3. The seller can further see details about companies and professionals that are connected by a series of relationships. Each relationship is labelled with its type. Current relationships are displayed with a solid line and previous relationships with a dashed line. The size of the node is dictated by its connectivity with the root company and is related to the size of the other nodes in the subgraph, that is to say it is scaled based on the scores of the other nodes in the current subgraph.

An important feature is the interaction design to allow the sellers to make sense of a large graph. It has been suggested that producing a good visualization gets harder as the graph gets larger with difficulties in the syntactic (e.g., avoiding occlusion), and semantic (e.g., highlighting the important features of the underlying network) areas [31]. With this in mind, the decision was taken to limit the initial view to a subgraph so that

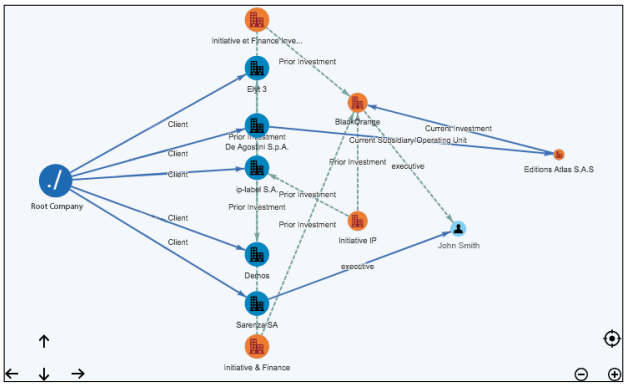

Fig. 3. A detailed view of the company BlackOrange and its connections

the user did not suffer from information overload and that initial view of the graph is not as overwhelming as it would be if the subgraph was too large for the window in which it is presented. Not every link between the root company and the target company is visualized, but just a selection of the shortest paths. The user can explore the graph in any direction that they find interesting by requesting the visualization of any further connections that the selected node in the subgraph may have. Information about the target company and each other node in the subgraph is available below the diagram in the form of short biographies of professionals and company descriptions.

A good visualization should emphasize the readability and promote the understanding of the underlying relationships. Our aim with this visualization is to afford users the opportunity to understand the intrinsic information contained in the structure of the network, enabling them to better understand the data that is available about the companies and professionals. The external image of the graph becomes a kind of *information store* [35], allowing the sellers to use it as a space for visual problem solving.

Ranking Mechanism Marketing campaigns allow companies to capture the interest of prospective clients. This interest is converted into leads passed to the sales division. The seller who is supposed to follow up on these leads has to go through a careful skimming phase which involves prospecting and prioritizing the companies in the list. Our domain experts estimate that a seller spends over 30% of their time on these activities.

To support the seller in taking a decision about which companies should have the highest priority and which ones may not be worth pursuing, the Client Network assigns a score to each company in the network. These scores are computed using our Node Relevance Algorithm (NORA), which computes a *flow value* of each node in a graph. Then, nodes can be ranked (ordered decreasingly) based on the scores. The flow value of a node measures the strength of the relation to the source node (seller's company). See details about the computation of flow values later in this section.

Intuitively, the sellers should approach a company that has a close connection to their company. Additionally, a company that has multiple connections to the sellers' company should be considered as a strong candidate. NORA's flow values attempt to

Algorithm 1: NORA ranking technique

input : graph $G = (V, E)$, start node s
output: Scores of all nodes, to be used for ranking
1 $(\Delta, E') \leftarrow$ AnalyzeNetwork(G, s)
2 **for** $v \in V$ *in the order given by* Δ **do**
3 $F(v) = \gamma \times \Sigma_{p \in P} \frac{F(p)}{\text{od}(p)}$, where P is the set of all parent nodes in $G' = (V, E')$ and $\gamma < 1$ is a discount factor.
4 **return** F

satisfy these criteria by considering the subset of the global topological structure of the Client Network.

Algorithm 1 shows the main steps of NORA in pseudocode. At line 1, method `AnalyzeNetwork` has a three-fold purpose: compute an ordering of the nodes Δ, and a filtered set of edges $E' \subseteq E$. Secondly, edges in E' become directed edges, as explained later in this section. Finally, E' allows NORA to ensure the convergence of flow values with low computational complexity. We will introduce two approaches of this step, leading to two variations of our algorithm, called NORA-D and NORA-T.

The next step of NORA (lines 2–3 in Algorithm 1) is to compute a function $F : V \rightarrow \Re_+$, called the flow-value function. A lookup table is a simple and easy implementation for F. Nodes are parsed in the order given by Δ. It remains to describe the actual formula to compute the flow value of a given node v (line 3 in the pseudocode). Given a node p, let $\text{od}(p)$ be the number of outgoing directed edges from p, in graph G'. For a given node v, let P be the set of all parent nodes in G'. The flow of v is defined as: $F(v) = \gamma \times \Sigma_{p \in P} \frac{F(p)}{\text{od}(p)}$, where $\gamma < 1$ is a discount factor. Intuitively, a parent node p will distribute its flow value among its children in G'. In the formula above, the value is shared equally among the children nodes of p. A child node accumulates flow values from all its parent nodes in G'. The discount factor γ implements the intuition that, if a node v is further away from s, then its flow value should be smaller.

Next we present NORA-D and NORA-T, our algorithmic versions that differ in the way they implement step 1 in Algorithm 1.

NORA-D. In this variant of our algorithm, step 1, corresponding to `AnalyzeNetwork`, performs a one-to-many shortest-path search to produce its output. We discuss two possible implementations of shortest-path search, one based on Dijkstra's algorithm and one based on breadth-first search.

With no assumptions made about whether the edges have a uniform cost or not, the first implementation runs Dijkstra's algorithm [10] from the source node s. The Dijkstra algorithm takes as input a graph, such as $G' = (V, E')$, and a node $s \in V$. It computes the distance (i.e., the cost of a minimum-cost path) from v to any other node in the graph. As such, Dijkstra's algorithm is a one-to-many distance computation technique.

A standard implementation of Dijkstra's algorithm returns, for each node in $v \in V$, the cost of an optimal path from s to v. We have slightly modified the Dijkstra implementation to return additional information beside the optimal costs. Specifically, we mark all edges in E with the property that they belong to an optimal path from s to some

Algorithm 2: `Extended Dijkstra`

input : graph $G = (V, E)$, start node s
output: Ordering of nodes Δ, in increasing order of the distance from s; Edges that belong to shortest paths from s, in the data structure E'

```
1  g(s) ← 0
2  for v ∈ V, v ≠ s do
3      g(v) ← ∞
4  P ← {s}
5  while P ≠ ∅ do
6      n ← pop(P)
7      append n to Δ
8      for (n,c) ∈ E do
9          if g(c) > g(n) + cost(n,c) then
10             g(c) ← g(n) + cost(n,c)
11             insert c into P, unless this has been done before
12             MarkedIncomingEdges(c) ← ∅
13         (n,c) → MarkedIncomingEdges(c)
14 E' ← ∪_{v∈V,v≠s} MarkedIncomingEdges(v)
15 return Δ, E'
```

node v. More formally, let $\text{Opt}(a,b)$ be the set of all optimal paths from a node a to a node b. An edge e is marked iff $\exists v \in V, \exists \pi \in \text{Opt}(s,v)$ such that $e \in \pi$. Marked edges are added into the subset E'. We further assign a direction to the edges in E', from the node with a smaller cost to the node with the larger cost. As such, the graph $G' = (V, E')$ is a directed acyclic graph (DAG).

Algorithm 2 shows the Dijkstra's algorithm in pseudocode, together with our extension that marks all edges that we want to keep after filtering. Dijkstra's algorithm uses a priority list P populated with elements $n \in V$, ordered on the cost $g(n)$. The cost $g(n)$, also called the g value of the node n, is the cost from the starting node to the current node n. Popping a node from P returns and removes from P an element with a smallest g value. The algorithm expands each node once, in an increasing order of the g value. Expanding a node n updates the g value of each successor c, if reaching c through n is a shortest path from s to c among all paths from s to c explored so far. At an expansion step, we also insert successor nodes into the priority list, unless a given successor node has been inserted into P at a previous time. Our extensions, which keep track of Δ and E', are shown at lines 7, 12, 13, 14 and 15.

On a uniform-cost graph, we can replace Dijkstra's algorithm with breadth-first search. This reduces the complexity of method `AnalyzeNetwork`, as discussed later in this section.

NORA-T. In constructing a DAG based on Dijkstra's algorithm, NORA-D only takes into account the edges that generate shortest paths. On the other hand, in NORA-T, method `AnalyzeNetwork` transforms the original graph into a DAG $G' = (V, E)$ where the edges in E' do not necessarily have to be created from edges that belong to

Algorithm 3: `AnalyzeNetwork (NORA-D)`

input : graph $G = (V, E)$, start node s
output: node ordering Δ; set of filtered edges E'
1 Run Extended Dijkstra and take Δ and E'
2 **return** Δ, E'

Algorithm 4: `AnalyzeNetwork (NORA-T)`

input : graph $G = (V, E)$, start node s
output: node ordering Δ; set of filtered edges E'
1 Generate a DAG $G' = (V, E')$ from G
2 Perform topological sort and take Δ
3 **return** Δ, E'

shortest paths in G. In this way, NORA-T attempts to consider more connections that may impact the companies.

NORA-T performs a topological sorting and obtains a node ordering Δ (see Algorithm 4). If G' has an edge (n, m), the topological ordering will place n before m [8]. In line 2 of Algorithm 1, the topological node ordering Δ ensures that a node n accumulates the values propagated to n via all paths in G'.

The first step of NORA-T is to assign a direction to each edge in the graph, direction which corresponds to the one given by traversing the graph in a breadth-first manner, starting with source node s. The second step of NORA-T is to convert the directed cyclic graph resulting from the fist step into a DAG. Eliminating a *minimum* number of edges from a directed cyclic graph to obtain a DAG is known as the feedback arc set problem, which is NP-hard [18]. However, approximation techniques that run in polynomial time exist. We take such an approximation approach, based on Eades et al.'s algorithm [11]. More specifically, NORA-T computes a minimum feedback arc set using the Eades et al.'s algorithm [11] and then builds the final DAG G' by removing from G all the edges in the set.

Either depth-first or breadth-first search can be used for the topological sort. We employ depth-first search, as in [29, 8].

Worst-case Computational Complexity. When using Dijkstra's algorithm, the complexity of NORA-D is $O(|E| + |V| \log |V|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges. Dijkstra's algorithm is the step with the largest complexity. When the graph edges have uniform costs, replacing Dijkstra's algorithm with breadth-first search reduces NORA-D's complexity to $O(|E| + |V|)$. The complexity of topological sort and assigning direction to the edges is $O(|E| + |V|)$. The algorithm of Eades *et al.* calculates a feedback arc set in $O(|E|)$ time. NORA-T [11], therefore, calculates flow values in $O(|E| + |V|)$ time. Recall that, after applying NORA to compute flows, nodes need to be ranked. The ranking (sorting) complexity is within $O(|V| log(|V|))$. However, this step is specific to the problem, rather than being specific to an algorithm such as NORA. Regardless of the algorithm used to compute relevance scores for the nodes, nodes would have to be ranked based on those scores.

4 Evaluation Framework

Our evaluation goes into two main directions. Firstly, we evaluate the algorithm by formulating it as a solution to link prediction. Secondly, we gathered feedback from user interviews to evaluate the impact of our work from the perspective of real users.

The Client Network can support sellers in identifying prospective clients by highlighting the non-client companies which occupy the top positions in the ranking computed by the ranking algorithm. A human expert is involved in selecting the candidates, and contacting the selected candidates afterwards. A human expert will not have time to process the entire list, and they typically focus on a subset at the top of the list. Therefore, the *Precision at K* ($P@K$) is a key metric for evaluating the performance of the ranking algorithm. Specifically, we used Precision at 10, 50, 100 and 1000 for measuring the quality of the recommendations.

Moreover, for the sake of completeness, we added to the comparison metrics that are widely adopted in the link prediction literature [34], such as the Area Under the Receiver Operating Characteristic Curve (AUROC), the Area Under the Precision-Recall Curve (AUPR) and the Top $|P|$ Predictive Rate ($\text{TPR}_{|P|}$)[5].

The client relationship partitions the organization nodes into two distinct classes, *client* and *non-client*. These two are unevenly represented in the network. The imbalance ratio is 1 to 14 between client nodes and non-client nodes. Furthermore, about 50% of the client nodes are only connected to the root node. This type of node does not provide any information that can be leveraged by the link prediction algorithms, especially those that rely on the topological structure of the network. Hence, only the remaining half of client nodes together with the non-client ones are used in our experiments.

We compared NORA with two random-walk based methods, Rooted PageRank and PropFlow. Rooted PageRank (RPR) [20] is a modified version of PageRank. The rank of a node corresponds to the probability that the node will be reached through a random walk from the source. A parameter α specifies how likely the algorithm is to visit the node's neighbors rather than starting over. PropFlow [21] is related to Rooted PageRank, but it is more localized. The rank of a node corresponds to the probability that a restricted random walk starting at the source node ends at the target node in no more than l steps. Unlike RPR, PropFlow does not require walk restarts or convergence but simply employs a modified breadth-first search restricted to depth l.

We used a 10-fold cross-validation stratified edge holdout scheme to measure the performance of the three algorithms in predicting links of type client. A formal description of the evaluation methodology is presented in Algorithm 5. At line 1, method `SampleFolds` randomly samples nodes from the client and non-client classes. Its implementation ensures that the distribution of the two classes is preserved in each fold.

Method `ComputeMetrics` (line 10 in Algorithm 5) calculates the Precision at 10, 50, 100, 1000 as well as the AUROC, AUPR and top $|P|$ predictive rate by labelling the client and non-client nodes in the ith fold as positive and negative instances, respectively. These values are computed for each algorithm j and fold i and stored into the multidimensional vector `Metrics`.

[5] Where P is the set of positive instances in the prediction results [34].

Algorithm 5: Evaluation Framework Workflow

Data: graph $G = (V, E)$, root node r, set of link prediction algorithms AS
Result: average values of AUROC, AUPR, top $|P|$ predictive rate and Precision at 10, 50, 100, 1000 for each algorithm in AS

```
Folds ← SampleFolds(V)
for each fold_i ∈ Folds do
    DC ← ∅
    for each node ∈ fold_i do
        if node is a client then
            insert node into DC
            RemoveClientLink(G, r, node)
    for each alg_j ∈ AS do
        Scores ← ComputeScores(G, r, alg_j)
        Metrics[j][i] ← ComputeMetrics(Scores, fold_i)
    for each node ∈ DC do
        AddClientLink(G, r, node)
for each row_j ∈ Metrics do
    ComputeAverageMetrics(row_j)
```

Results. All evaluated algorithms are implemented in Python using the NetworkX module [13]. An *undirected multigraph* represents the structure of the network. Further details on the configuration parameters used in the experiments are available in Table 1.

Table 2 contains the averages of the metrics for each algorithm. NORA-D achieves the highest scores for the Precision at 10 and 50 while NORA-T places the highest number of disconnected clients on the top 100 and 1000 of its ranking. Rooted PageRank offers the highest performance on the AUROC, AUPR and $\text{TPR}_{|P|}$ but it is also the worst in term of Precision when considering up to position 1000 of the ranking. PropFlow never excels in any of the metrics although it is never the worst.

PropFlow has a computational complexity of $O(|V| + |E|)$, where $|V|$ is the number of nodes and $|E|$ the number of edges in the graph. So do both NORA variants, since the graph has edges with uniform cost, as explained in complexity analysis section. Rooted PageRank is the algorithm with the highest time complexity in the group since each $iteration$ of the algorithm runs in linear time $O(|V| + |E|)$.

In summary, NORA has a good time complexity, and it performs well for the Precision at K metrics. It is worth mentioning again that these metrics are very important in our application since the list of companies that the Client Network recommends the sellers to approach is extracted from the top of the ranking.

[6] We use the term "weight" for indicating the flow capacity associated to each edge.
[7] The values are averages over the 10 folds. In bold the highest value for each metric.

Table 1. Configurations used in the experiments

Algorithm	Parameters
Rooted PageRank	α = 0.15; Error tolerance = $1.0e-6$ Maximum number of iterations = 100
PropFlow	Depth = 10
NORA (both variants)	Discount factor = 0.95
All	Edge cost = 1.0; Edge weight[6]= 1.0

Table 2. Client Prediction Results[7]

Algorithm	$P@10$	$P@50$	$P@100$	$P@1000$	$\text{TPR}_{\lvert P\rvert}$	AUROC	AUPR
Rooted PageRank	0.39	0.55	0.578	0.589	**0.338**	**0.917**	**0.292**
PropFlow	0.81	0.73	0.715	0.628	0.313	0.833	0.26
NORA-D	**0.85**	**0.746**	0.722	0.617	0.265	0.816	0.214
NORA-T	0.82	0.734	**0.728**	**0.634**	0.284	0.823	0.232

5 User interviews

Our aim in these interviews was to understand which features of the application were considered the most and least useful by the users and which features could achieve a time saving when approaching prospective clients. We aimed to understand if the visualization and the ranking would aid the user in their daily tasks.

Method. We demonstrated the system in the sales department in our organization, after which we recruited 5 volunteers whose job roles involve evaluating whitespace clients. They were asked to evaluate the usability and usefulness of the Client Network in relation to their daily tasks. After initial group demonstrations, they tried out the functionality. We followed up with semi-structured interviews. A questionnaire with open-ended questions was developed which allowed some scope for exploring some of the responses in depth. We found this approach useful as the sellers have different roles in the organization and therefore were using the Client Network in different ways.

Results. The sellers were enthusiastic about the functionality of the Client Network, but they also pointed out a need for a deeper integration with the Sales Customer Relationship Management (CRM) tool they use before they can really make use of its time-saving potential. There were positive responses to the company search coverage, indicating that they could find results related to the companies they were searching for, but there were some omissions when they searched for public sector clients.

Sellers showed an interest in the application's display of information about the movement of people and subsidiaries. This information is difficult, if not impossible, to uncover in their existing software toolset, and is useful for them to know before they contact the company. Several sellers thought that the company description is very useful information to display because *"...it's important to know a bit about the client when you are calling the client..."*, and that this information could be useful as a conversational "hook" when cold calling *"...when you call 20 people a day you obviously cannot research every individual in a very detailed way."*

The majority of sellers reported that the visualization was the most important part of the application to them from an exploratory point of view. A number of key requirements have been identified for successful network exploration tools, including high quality layout algorithms, data filtering, clustering, statistics and annotation [6]. The findings of the user interviews suggest that we should reconsider our approach to annotating the data, and that showing the visualizations as the only interface to the data should be reconsidered if we are considering a task-based rather than exploratory use, to afford users the option of exploring visually or in a more structured manner.

There were very positive comments about the time-saving aspect of showing the company and subsidiary information, and the fact that the data was more up to date than the existing system that the seller would look in for this type of information. UI improvements that were suggested were to add a legend to the visualization to help interpret the different type of connections and to improve the graph search filters.

There was a strong belief from the majority of the sellers that the prototype could be even more useful if it was expanded to display historical product information when showing information about linked companies, for example, if there was a linkage between nodes on the graph and the sales CRM that is used in the company so that you could see which products the connected clients had purchased previously *"... there is a lot of functionality that if they are expanded in the right way, can actually help us a lot, and integrations are the very zero point to start..."*. Another case which was suggested for the graph is to highlight the companies that have business relationships, as this is helpful when the sellers are trying to target business-to-business products.

6 Conclusion and Future work

In this paper, we formalized the problem of identifying prospective clients in sales and proposed an innovative solution, Client Network. It utilizes a novel ranking algorithm for predicting new clients and provides an interactive interface for allowing the exploration of the client ecosystem. While typical approaches study client interactions or explore clients' similarity to existing clients based on market segmentation, our approach leverages the whole structure of the client ecosystem. This ecosystem can be represented as a heterogeneous network and, in such a context, the problem of identifying prospective clients can be formalized as an instance of the link prediction problem. We presented NORA, a novel ranking algorithm and compared its performance with two well known algorithms in the link prediction field. The experiments demonstrate that our technique achieves higher precision than Rooted PageRank and PropFlow. Our approach can be considered highly suitable for the use cases covered by the Client Network. In user interviews sellers expressed their appreciation about how the Client Network can help them in prioritizing leads and how the information it provides is complementary to that available from the tools they use to work with. As part of future work, we will iterate the design of the UI based on the user feedback, add new features based on new use cases and explore more novel ways to display the subgraph. Furthermore, we aim to further improve the quality of the recommendations and re-evaluate the performance of the algorithms with new experiments.

References

1. Adafre, S.F., de Rijke, M.: Discovering missing links in wikipedia. In: Proceedings of the 3rd International Workshop on Link Discovery. pp. 90–97. LinkKDD '05, ACM (2005). https://doi.org/10.1145/1134271.1134284
2. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social Networks **25**(3), 211–230 (2003)
3. Aiello, L.M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., Menczer, F.: Friendship prediction and homophily in social media. ACM Trans. Web **6**(2), 9:1–9:33 (Jun 2012)
4. Airoldi, E.M., Blei, D.M., Xing, E.P., Fienberg, S.E.: Mixed membership stochastic block models for relational data, with applications to protein-protein interactions. In: Proceedings of International Biometric Society-ENAR Annual Meetings (2006)
5. Almansoori, W., Gao, S., Jarada, T.N., ElSheikh, A.M., Murshed, A.N., Jida, J., Alhajj, R., Rokne, J.G.: Link prediction and classification in social networks and its application in healthcare and systems biology. NetMAHIB **1**(1-2), 27–36 (2012)
6. Bastian, M., Heymann, S., Jacomy, M., et al.: Gephi: an open source software for exploring and manipulating networks. Icwsm **8**(2009), 361–362 (2009)
7. B.V., A.: Vis.js (2018), http://visjs.org/
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, 3 edn. (2009)
9. Davis, D., Lichtenwalter, R., Chawla, N.V.: Multi-relational link prediction in heterogeneous information networks. In: Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining. pp. 281–288. ASONAM '11 (2011)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (Dec 1959). https://doi.org/10.1007/BF01386390, http://dx.doi.org/10.1007/BF01386390
11. Eades, P., Lin, X., Smyth, W.F.: A fast and effective heuristic for the feedback arc set problem. Information Processing Letters **47**, 319–323 (1993)
12. Freschi, V.: A graph-based semi-supervised algorithm for protein function prediction from interaction maps. In: LION. Lecture Notes in Computer Science, vol. 5851, pp. 249–258. Springer (2009)
13. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference. pp. 11–15. Pasadena, CA (2008)
14. Hasan, M.A., Zaki, M.J.: A survey of link prediction in social networks. In: Social Network Data Analytics, pp. 243–275 (2011)
15. Huang, Z., Li, X., Chen, H.: Link prediction approach to collaborative filtering. In: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 141–142. JCDL '05, ACM, New York, NY, USA (2005)
16. Jeh, G., Widom, J.: Simrank: A measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 538–543. KDD '02, ACM (2002)
17. Johnson, R.A., Yang, Y., Aguiar, E., Rider, A., Chawla, N.V.: Alive: A multi-relational link prediction environment for the healthcare domain. In: Proceedings of the 2012 Pacific-Asia Conference on Emerging Trends in Knowledge Discovery and Data Mining. pp. 36–46. PAKDD'12, Springer-Verlag, Berlin, Heidelberg (2013)
18. Karp, R.M.: Reducibility among combinatorial problems. In: a symposium on the Complexity of Computer Computations. pp. 85–103 (1972)
19. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (March 1953), http://ideas.repec.org/a/spr/psycho/v18y1953i1p39-43.html

20. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007). https://doi.org/10.1002/asi.20591, http://dx.doi.org/10.1002/asi.20591
21. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 243–252. KDD '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1835804.1835837, http://doi.acm.org/10.1145/1835804.1835837
22. Martínez, V., Berzal, F., Cubero, J.C.: A survey of link prediction in complex networks. ACM Comput. Surv. **49**(4), 69:1–69:33 (Dec 2016). https://doi.org/10.1145/3012704, http://doi.acm.org/10.1145/3012704
23. Mori, J., Kajikawa, Y., Kashima, H., Sakata, I.: Machine learning approach for finding business partners and building reciprocal relationships. Expert Systems with Applications **39**(12), 10402 – 10407 (2012). https://doi.org/https://doi.org/10.1016/j.eswa.2012.01.202, http://www.sciencedirect.com/science/article/pii/S0957417412002308
24. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford University (1999)
25. Shakibian, H., Moghadam Charkari, N.: Mutual information model for link prediction in heterogeneous complex networks. Scientific Reports **7**, 44981 (03 2017), http://dx.doi.org/10.1038/srep44981
26. Shi, C., Li, Y., Yu, P.S., Wu, B.: Constrained-meta-path-based ranking in heterogeneous information network. Knowl. Inf. Syst. **49**(2), 719–747 (2016), http://dblp.uni-trier.de/db/journals/kais/kais49.html#ShiLYW16
27. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P.S.: A survey of heterogeneous information network analysis. IEEE Trans. on Knowl. and Data Eng. **29**(1), 17–37 (Jan 2017). https://doi.org/10.1109/TKDE.2016.2598561, https://doi.org/10.1109/TKDE.2016.2598561
28. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. PVLDB **4**(11), 992–1003 (2011), http://dblp.uni-trier.de/db/journals/pvldb/pvldb4.html#SunHYYW11
29. Tarjan, R.E.: Edge-disjoint spanning trees and depth-first search. Acta Informatica **6**, 171–185 (1976)
30. Wang, P., Xu, B., Wu, Y., Zhou, X.: Link prediction in social networks: the state-of-the-art. CoRR **abs/1411.5118** (2014), http://arxiv.org/abs/1411.5118
31. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information visualization **1**(2), 103–110 (2002)
32. Wu, S., Sun, J., Tang, J.: Patent partner recommendation in enterprise social networks. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. pp. 43–52. WSDM '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2433396.2433404, http://doi.acm.org/10.1145/2433396.2433404
33. Yang, Y., Chawla, N., Sun, Y., Hani, J.: Predicting links in multi-relational and heterogeneous networks. In: Proceedings of the 2012 IEEE 12th International Conference on Data Mining. pp. 755–764. ICDM '12, IEEE Computer Society, Washington, DC, USA (2012). https://doi.org/10.1109/ICDM.2012.144, http://dx.doi.org/10.1109/ICDM.2012.144
34. Yang, Y., Lichtenwalter, R.N., Chawla, N.V.: Evaluating link prediction methods. CoRR **abs/1505.04094** (2015)
35. Zhang, J., Norman, D.A.: Representations in distributed cognitive tasks. Cognitive science **18**(1), 87–122 (1994)
36. Zhu, J., Hong, J., Hughes, J.G.: Using markov models for web site link prediction. In: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia. pp. 169–170. HYPERTEXT '02, ACM, New York, NY, USA (2002). https://doi.org/10.1145/513338.513381, http://doi.acm.org/10.1145/513338.513381