Active Search of Connections

for Case Building and Combating Human Trafficking

Reihaneh Rabbany, David Bayani, Artur Dubrawski
Carnegie Mellon University
Pittsburgh, PA
{rrabbany,dbayani,awd}@cs.cmu.edu

ABSTRACT

How can we help an investigator to efficiently connect the dots and uncover the network of individuals involved in a criminal activity based on the evidence of their connections, such as visiting the same address, or transacting with the same bank account? We formulate this problem as Active Search of Connections, which finds target entities that share evidence of different types with a given lead, where their relevance to the case is queried interactively from the investigator. We present **RedThread**, an efficient solution for inferring *related* and *relevant* nodes while incorporating the user's feedback to guide the inference. Our experiments focus on case building for combating human trafficking, where the investigator follows leads to expose organized activities, i.e. different escort advertisements that are connected and possibly orchestrated. RedThread is a local algorithm and enables *online* case building when mining millions of ads posted in one of the largest classified advertising websites. The results of RedThread are *interpretable*, as they explain how the results are connected to the initial lead. We experimentally show that RedThread learns the importance of the different types and different pieces of evidence, while the former could be transferred between cases.

KEYWORDS

Active Learning, Graph Construction, Link Inference

ACM Reference Format:
Reihaneh Rabbany, David Bayani, Artur Dubrawski. 2018. Active Search of Connections: for Case Building and Combating Human Trafficking. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220103

1 INTRODUCTION

Studying how elements of data are connected to each other can help uncover salient patterns in data. The connections between data elements are often assumed to be observed, at least partially. However, in many real-world applications, these connections are not given a priori and need to be inferred from the available data. Extreme examples include the covert networks of terrorists, criminals,

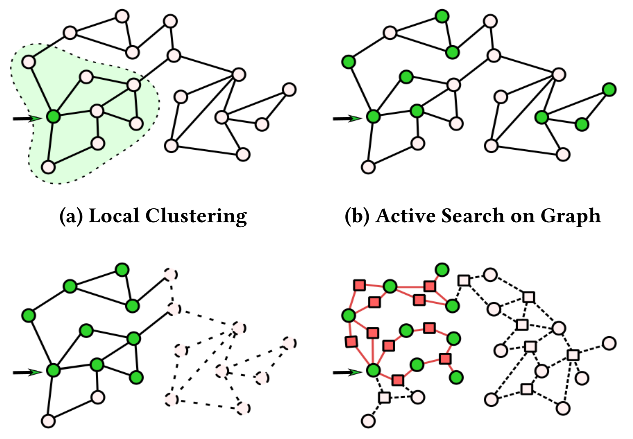


Figure 1: For a given seed query, a local clustering algorithm finds closely related nodes to the seed, and an active search algorithm finds the relevant nodes to the seed. RedThread finds related and relevant nodes, similar to the active exploration, while inferring the relations at the same time.

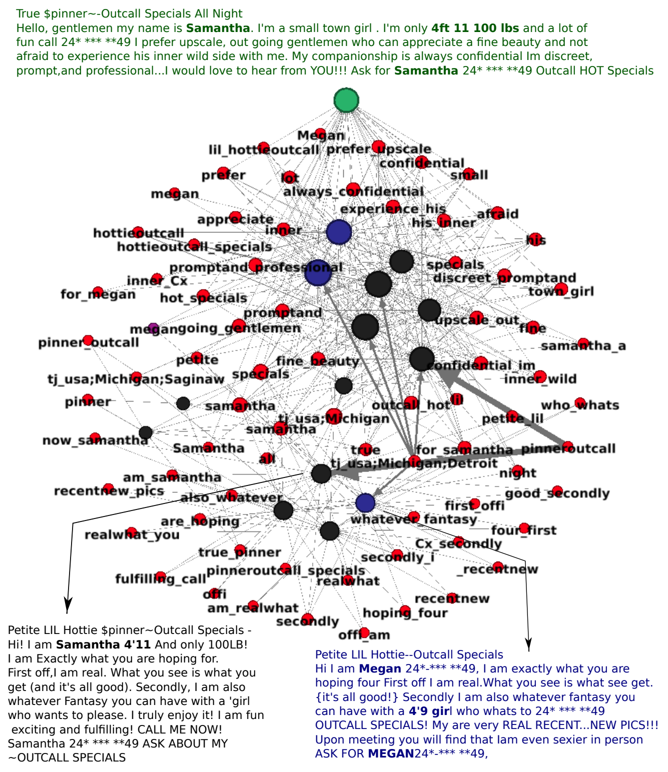
their accomplices and their victims, which are intentionally hidden. Although invaluable, the process of extracting such networks is usually difficult and laborious. Known as link charts in the law enforcement setting, the common practice is for an expert to manually link the entities of interest by examining the evidence of their association, e.g. shared addresses, related bank accounts, etc. [20, 45, 47]. This is often facilitated by visualization tools with built-in database lookups. The majority of research efforts is devoted to the analysis of these networks after they are extracted [7, 25, 44, 45], e.g. to detect influential members. Here, we present an interactive approach for more efficient extraction of these connections, using an active learning framework.

We introduce the problem of Active Search of Connections, i.e. *to infer the connections between entities from the evidence available in data, where the user is able to provide feedback and guide the network inference.* Active Search of Connections is closely related to local clustering, active search and active exploration on graphs. However, the solutions for those problems can not directly be applied in this setting. Local clustering on graphs [23, 39] (Fig. 1a) also commonly known as local community detection, finds a group of well-connected nodes to the seed query. In our setting, *not all the closely related nodes are relevant to the case.* Active search on graphs [12] (Fig. 1b) finds the maximum number of relevant nodes, i.e. nodes of the same target class as the seed query, by querying few labels from the user. This algorithm explores the entire graph assuming that relevant nodes are scattered, however, Active Search of Connections searches for nodes whose *connection can be explained to the case.* Our proposed RedThread solution for Active Search of Connections, also queries a limited number of labels from the user but to *locally* retrieve the maximum number of nodes *related* and *relevant* to the given seed. This is closer to active exploration [15, 28] (Fig. 1c) which also finds connected and relevant nodes. These methods, however, are built for when connections are observable, for example when crawling the web for relevant content, where hyperlinks are observed once reaching a page. Unlike the problems above, for Active Search of Connections (Fig. 1d) connections are not observed and are inferred from data.

Active Search of Connections is potentially useful in a wide range of domains, wherever **connections between entities are not given a priori**, in particular for mapping out the covert web of entities in fraud detection, counter-terrorism, or tracking online sales of illegal goods (weapons, drugs, etc.). We are particularly motivated by its application to support law enforcement **for case building in counter human trafficking** operations. This paper complements the recent efforts to develop data-driven techniques for tracking online human trafficking [3, 11, 17, 29, 34, 40, 42]. For case building and target identification in this domain, an initial lead is probed to find connected entities and eventually identify the person of interest. Currently, this task is performed by an expert investigator through manual exploration of the available data. RedThread makes this process semi-automatic and more efficient.

RedThread is designed to find **organized activities** in online escort ads, i.e. *ads marketing different potential victims which are linked by different types of evidence, e.g. phone numbers, catchphrases, misspelling and other text patterns, images with the same background, or other evidence of connection.* An example case built by RedThread is illustrated in Fig. 2. In this figure, the lead ad is the green node. From this lead, RedThread retrieved other relevant ads through the shared evidence, in this case mostly bigrams. Between these, there are ads for a different person who is being advertised by the same phone number. Phone numbers are used as oracle label in this case, i.e. RedThread learns to find persons being advertised by the same phone number. In practice, when the human investigator is available, the phone numbers are added to the evidence set instead of being put aside as training labels, which boosts the algorithm's effectiveness in finding advertisements posted by the same person.

We have applied RedThread to find similar cases in millions of escort ads posted on Backpage.com for cities across the U.S. and Canada posted between 2013 to 2017. We model this data as a k-partite graph, in which ads are connected to various types of evidence they share, e.g. phone numbers, urls, images, names, bigrams. The user's feedback is then used to guide the navigation through the graph when finding ads related to the given seed. RedThread learns the relative importance of each type of evidence (e.g. phone numbers would become stronger indicators than bigrams), as well as the relevance of specific pieces of evidence (e.g. some phone numbers are relevant to the current seed). The main intuition of RedThread is that the candidates for inclusion into the case graph are explored in order of how well-connected they are to the labeled nodes already explored, whereas the importances of these connections are updated incrementally based on the user's feedback.


True $pinner~-Outcall Specials All Night
Hello, gentlemen my name is **Samantha**. I'm a small town girl . I'm only **4ft 11 100 lbs** and a lot of fun call 24* *** **49 I prefer upscale, out going gentlemen who can appreciate a fine beauty and not afraid to experience his inner wild side with me. My companionship is always confidential Im discreet, prompt,and professional...I would love to hear from YOU!!! Ask for **Samantha** 24* *** **49 Outcall HOT Specials

Petite LIL Hottie $pinner~Outcall Specials -
Hi! I am **Samantha 4'11** And only 100LB!
I am Exactly what you are hoping for.
First off,I am real. What you see is what you get (and it's all good). Secondly, I am also whatever Fantasy you can have with a 'girl who wants to please. I truly enjoy it! I am fun exciting and fulfilling! CALL ME NOW!
Samantha 24* *** **49 ASK ABOUT MY ~OUTCALL SPECIALS

Petite LIL Hottie--Outcall Specials
Hi I am **Megan** 24*-*** **49, I am exactly what you are hoping four First off I am real.What you see is what see get. {it's all good!} Secondly I am also whatever fantasy you can have with a **4'9 gir**l who whats to 24* *** **49 OUTCALL SPECIALS! My are very REAL RECENT...NEW PICS!!! Upon meeting you will find that Iam even sexier in person ASK FOR **MEGAN**24*-*** **49,

Figure 2: Example Case Built by RedThread. The green ad (top) shows the seed. The black nodes are the discovered ads which are connected to the seed though shared evidence, plotted as red nodes. The text-box on the top left shows the content of the seed ad, and the bottom right text-box shows the content of a sample connected ad advertising a person with a different name and height.

In our experiments, we also use two publicly available datasets not related to escort ads to show RedThread's general applicability: (i) a music record dataset where RedThread retrieves records of the same artist, and (ii) a news dataset where RedThread finds memes originating from the same source. In all our datasets, RedThread achieves significant improvements over the baselines, including a random-walk which restarts given negative feedbacks. To summarize, **our main contributions are twofold**: first, we introduce the novel problem of Active Search of Connections motivated by its application in case building; second, we present RedThread method as an efficient, local and interpretable solution. For **reproducibility**, our source code is made publicly available at: https://github.com/rabbanyk/RedThread.

2 RELATED WORK

Related work in Section 2.1 overviews the **application-related** works on analyzing online escort advertisements, and Section 2.2 covers **theory-related** algorithms to Active Search of Connections.

2.1 Analyzing Online Escort Advertisements

Online classified advertising and social networking websites provide easy to use and low-risk platforms for traffickers which give them a sense of anonymity and enable wide geographic reach [21, 30, 37]. Millions of dollars are spent on online escort advertisements yearly. For example, Backpage.com, one of the main hubs, was estimated to have had a gross revenue of 120 million dollars (which contributed to more that 90% of its total revenue) from escort advertisements in 2014, according to court documents [32]. A majority of sex-trafficking victims (minors and adults who are being forced or coerced to provide sex commercially) are marketed online according to a recent survey [41]. The National Center for Missing and Exploited Children reports a 14-fold increase in reports of suspected child sex trafficking in 2014, which is correlated with the increased use of the online escort advertisements market [31].

Given the scale and importance of the problem, there have been multiple recent efforts to i) capture and extract information from the web mainly in form of knowledge graphs [17, 19, 40], e.g. to recognize location, name and age mentions in the ads, ii) discern patterns in the data [11, 27] e.g. frequency of ads around Super Bowl events, iii) train classifiers to flag suspicious ads given a small ground-truth of phone numbers associated with known traffickers [2, 3, 11, 29, 34, 42] based on textual features extracted from the content of the ads and the images associated with them, and iv) train classifiers to predict if two ads are related or not using strong link, i.e. phone numbers, as training labels, where features are extracted from content of the ads [29], and inferred bitcoin wallets information [34]. The latter link prediction methods are the most relevant to this paper, since we are also trying to find connections between the ads. RedThread is different in three ways from these methods. First, RedThread is a **local** and **scalable** method. It avoids computing pairwise similarities between the ads, which is unfeasible when facing millions of advertisements. Second, RedThread **explains** how ads are connected by the exact pieces of evidence they share, which is required for the produced link charts to serve the intended law enforcement usage. Third, RedThread is **learning interactively** from the investigator, to improve its performance. In our experiments, we use hard identifiers such as phone numbers as a proxy for a human expert for evaluation purposes since using an actual expert is not possible for experiments. In practice, the phone numbers will be part of the evidence set and the labels will be queried actively from the user. This adaptability is not available with the current classifier based techniques which also use the hard identifiers –phone numbers– as the training labels, e.g. [29, 34].

2.2 Active Search, Exploration and Clustering

Selective labeling in active learning enables analyzing data where labels are scarce. Here a given number of labels, determined by the query budget, can be queried from the user/oracle during the learning. In this setting, the task of recovering only the relevant portion of the data is known as active search [12, 13]; where the objective is to achieve the highest recall for a given class, instead of the overall classification accuracy. When targeting a specific class is the goal, e.g. for detecting fraud, drug discovery, or our case building problem, the active search technique can achieve better performance compared to the uncertainty sampling common in active classification [12, 13, 43, 43]. Active search methods formulate the problem as a sequential Bayesian decision theory problem and derive an optimal policy which is exponential and needs to be approximated using a fixed lookahead, using specific bounds to prune the search space, and narrowing the search spaces using a k-nearest-neighbor graph which allows only strategizing over similar states. In our case, even building the k-nearest neighbor graph is infeasible due to the scale of the data.

RedThread uses a similar learning framework but provides a **local and efficient** solution which draws inspiration from the developed optimal policy of these methods, i.e. sampling highly correlated regions [13].

Local clustering algorithms, a.k.a. community detection, are generally applied to when data is large scale. While global clustering algorithms partition a given graph, the local methods retrieve a cluster by expanding from a given seed node. Although the relevant entities in Active Search of Connections are assumed to be highly connected, the objective is different from the clustering algorithms, as we are interested in finding entities with positive labels which reflect the user's interest. There are several local graph clustering algorithms which also incorporate attributes for nodes [1, 8, 9, 24, 33, 39, 46]. These approaches consider attributes as an additional information source or metadata on the nodes, and develop unsupervised algorithms to cluster this heterogeneous data. RedThread however is proposing a semi-supervised or active paradigm, where we are using the labels on the nodes, provided by the expert user, as the true clustering. In this sense, RedThread is also closely related to active exploration [15] or selective harvesting [15, 28]. These methods however assume the graph is (partially) observable, which is not true in our case. RedThread **infers the connections** at the same time as finding the relevant nodes.

Finally, RedThread is also related to measuring proximity in graphs [38], to build the k nearest neighbor structure to search over [12], and various graph extraction techniques [10, 14, 26] and relational data knowledge base extraction methods [16, 35, 36]. These problem settings however differ from ours as RedThread focuses on a local active setting targeted at finding related entities. Another class of related works are entity resolution methods [4–6], which try to find references to the same underlying entity. RedThread is interested to find different entities (two or more possible victims) which are being advertised in the same fashion. In our experiments, we use a basic entity resolution technique to detect repetitive ads of a same person posted over time (with minor modifications), in order to both avoid querying the user with duplicate ads, and to trim the case being built.

3 PROBLEM DEFINITION

Consider n datapoints $\mathcal{D} = \{d_1, d_2 \dots d_n\}$ connected to k different types of evidence (e.g. phone number, image, bi-grams) which we refer to as modalities. Let evidence set

$$\mathcal{E} = \{\mathbf{X}_1, \mathbf{X}_2 \dots \mathbf{X}_k\} \quad (1)$$

denote the set of indicator matrices for these k modalities, i.e. $\mathbf{X}_m \in \mathbb{R}_+^{n \times c_m}$ for $m \in [1 \dots k]$; where c_m is the cardinality (number of unique pieces of evidence) of modality m (e.g. number of unique phone numbers). Each column of $\mathbf{X}_m$ shows a set of datapoints that share the corresponding evidence (e.g. datapoints that all share a particular phone number), and each row of $\mathbf{X}_m$ shows the pieces of evidence associated to the corresponding datapoint (all the phone numbers mentioned in a particular datapoint). **Shared evidence** across modalities for two datapoints i and j can be derived as:

$$s(i,j) = \cup_m \{u \in [1 \dots c_m] \mid [\mathbf{X}_m]_{iu} > 0 \wedge [\mathbf{X}_m]_{ju} > 0\}$$

Given the evidence set $\mathcal{E}$, Active Search of Connections finds datapoints of interest which are related to the given seed i through shared evidence; whereas being of interest is queried from the user or oracle. More precisely,

*Definition 3.1 (**Active Search of Connections**).* Given seed $i \in \mathcal{D}$, evidence set $\mathcal{E}$ (Eq. (1)), and a fixed query budget, $0 < b << n$, find the maximum number of related and relevant entities to i. Datapoint j is considered relevant and related to i iff $r(i,j) = 1$, i.e.

$$r(i,j) = 1 \iff i = j \vee \{\exists k\,,\, r(i,k) = 1 \wedge y_j = 1 \wedge s(j,k) \neq \emptyset\}$$

where we assume the unknown label $y_j \in \{-1, 1\}$ can be queried from the oracle for each $j \in \{1..n\}$.

We emphasize that in Active Search of Connections, **labels are local and depend on the given seed**, since they indicate whether other datapoints are related to the current case. This is different than the usual active search framework where the labels are global and positive labels might not be connected through the structure, for instance in active search on graphs [43] where the example task is to retrieve all the nips papers in citeseer, i.e. nodes are either positive or negative regardless of the seed node.

4 METHODOLOGY

We can consider each indicator matrix in the evidence set $\mathcal{E}$ represents the incidence matrix of a hypergraph, or the feature matrix of the datapoints, or the biadjacency matrix of a 2-partite graph. We will adopt the latter in the rest of the paper, from which all the modalities form a k-partite graph representation for the data. We refer to this as k-modal evidence (KME) graph. More specifically,

*Definition 4.1 (**k-modal evidence graph**).* Given evidence set $\mathcal{E}$ construct a heterogeneous graph with *vertex set*

$$V = \underbrace{\bigcup_{m=1}^{k} \{u_1^m, u_2^m, \dots u_{c_m}^m\}}_{V_{\mathcal{E}}\text{: evidence nodes}} \cup \underbrace{\{v_1, v_2 \dots v_n\}}_{V_{\mathcal{D}}\text{: datapoint nodes}}$$

where $\{u_1^m, u_2^m, \dots u_{c_m}^m\}$ are c_m nodes corresponding to the pieces of evidence of type m, i.e. the unique values of modality m (e.g. one node per each phone number); and $\{v_1, v_2 \dots v_n\}$ show the nodes corresponding to the n given datapoints (e.g. one node per each advertisement). *Edge set* of this representation is then defined as:

$$E = \{(v_i, u_j) \mid [\mathbf{X}_m]_{ij} > 0, v_i \in V_{\mathcal{D}}, u_j \in V_{\mathcal{E}}\}$$

which considers an edge between each data point and every evidence it is associated with, e.g. edges would be formed from the node corresponding to the i^{th} advertisement, v_i, to the pieces of evidence in that advertisement, phone numbers, images, etc.

Given this k-modal evidence graph and seed node of interest v_i, *the Active Search of Connections translates to finding nodes in*

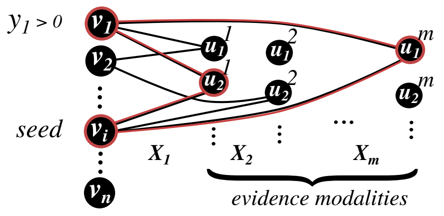


Figure 3: k-modal evidence graph: nodes consist of datapoints and different types of evidence associated with them; edges derive from the evidence indicator matrices, and labels are queried from the user.

$V_{\mathcal{D}}$ that are (tightly) connected to node v_i with even length paths (i.e. through shared evidence) and are relevant to the seed (i.e. have positive labels).

In the next section we propose RedThread which navigates through this graph to efficiently find these nodes while learning the importance of each modality and each piece of evidence from the labels (user's feedbacks) obtained while expanding. Before that, we start with describing the general learning framework and baselines.

4.1 Basic Baselines

The baseline algorithms as well as the RedThread have an iterative nature. The algorithms start from the given seed, and in each iteration picks another datapoint that it deems related to the seed, then queries the user to see if this is correct i.e. observes the label. Algorithm 1 outlines this framework.

Algorithm 1 Iterative Labeling ($seed$, $\mathbf{X}_1 \dots \mathbf{X}_k$, oracle, budget)

```
1: $b \leftarrow 0$                                          // initialize query counter
2: $\mathcal{L} \leftarrow \{seed : 1\}$                       // initialize labeled hash
3: while $b <$ budget do
4:     $j \leftarrow infer(\{\mathbf{X}_1 \dots \mathbf{X}_k\}, \mathcal{L})$       // pick a datapoint j
5:     if $j \notin \mathcal{L}$ then
6:         $\mathcal{L}[j] \leftarrow oracle(j)$               // query user if j is relevant
7:         $b \leftarrow b + 1$
8:     end if
9: end while
```

4.1.1 *Random (Rand).*

The most naive baseline picks a datapoint at random, while ignoring both the given evidence and user's feedbacks. In other words, the function $infer()$ in algorithm 1 simply chooses j from $\{1 \dots n\}$ uniformly at random.

To make use of the available evidence, the following algorithms construct the k-modal evidence graph of definition 4.1 (KME), and navigate through it. From here on we use i and v_i interchangeably to denote the i^{th} datapoint and its corresponding node in the graph.

4.1.2 *Random Walk (RW).*

This baseline expands from the seed by randomly walking through its neighbors, querying labels from the expert on each encounter of an unlabeled node. In more details, given seed node v, $\mathcal{N}(v) = \{u|(v,u) \in E\}$ gives the pieces of evidence connected to v, from which an evidence node u is selected uniformly at random to expand from, that is by randomly choosing the next node from the neighbors of u which are not previously labeled as negative, i.e. $\mathcal{N}(u) \setminus \mathcal{L}^-$, where $\mathcal{L}^- = \{j \in \mathcal{L} | y_j < 0\}$. This process repeats until budget is exhausted and restarts from the seed whenever it gets a negative feedback. In this way, the random walk is taking into account the feedbacks by only expanding on positives. We are not, however, learning what caused the algorithm to reach to a positive or negative instance. The aspect missing here is present in the heart of RedThread, which learns the importance of different pieces of evidence and types of evidence as it acquires labels. Before moving to the description of the learning mechanism, we discuss one more baseline which uses a weighting scheme to adjust the importance of different pieces of evidence.

4.1.3 Random Walk Adjusted by Inverse Degree (wRW). Considering rare pieces of evidence are more telling, here we pick the evidence nodes proportional to the inverse of their degree, i.e. probability of u being selected is

$$p_u \propto \frac{1}{d_u}$$

where d_u denotes the degree of evidence u, $d_u = |\mathcal{N}(u)|$. Since the random walk still picks the neighbors of the evidence nodes uniformly at random, the transition probability from node v_i to v_j through evidence u would be $1/d_u^2$. This is analogous to re-weighting the edges connected to the evidence nodes by their inverse degree, which RedThread also uses. This weighting scheme resembles the popular inverse document frequency commonly used in information retrieval.

4.2 RedThread

RedThread considers weights for different modalities (evidence types), assuming that some evidence are stronger than others, e.g. sharing a phone number is a stronger indicator of two ads being related than advertising persons with the same names. Moreover, given the current seed node, specific pieces of evidence become more and more relevant as they point to more positive instances, therefore the weight of evidence should also be adjusted as we get more labels. In the following, we first describe the parameters of RedThread, then explain how RedThread adaptively learns these parameters as labels are acquired, and applies them to infer the best node to query next from the user for labeling.

4.2.1 Weighing the Modalities. RedThread considers a weight for each partition of the KME graph to enforce the importance of its corresponding modality in the given evidence set, i.e.

$$\Theta = [\theta_1, \theta_2, \ldots \theta_k]$$

Here θ_m denotes the weight/importance of modality m. Correspondingly, RedThread considers the evidence flow coming from each partition separately, i.e. let s_m^j show the evidence support for v_j from modality m, then we consider a tie-strength vector for each expansion candidate v_j as:

$$\mathbf{s}^j = [s_1^j, s_2^j, \ldots s_k^j]$$

4.2.2 Computing the Evidence Flow. We can consider s_m^j simply as the number of (length two) paths that go through evidence nodes in partition m to reach v_j. To enforce the hypothesis that rare pieces of evidence are more important, RedThread further weighs down the pieces of evidence by their prevalence (the number of datapoints they are associated with) measured as their degree in the graph. In more detail, if node v_i and v_j are connected through evidence u, this evidence contributes to their tie strength by $1/d_u^2$, as each edge is down weighted by the degree. Moreover, the tie strength score of reaching v_j is summed from all the currently explored (labeled) positive nodes as

$$s_m^j = \sum_{v_i \in \mathcal{L}^+} \sum_{u \in \mathcal{N}_m(v_i) \cap \mathcal{N}_m(v_j)} \frac{1}{d_u^2} \tag{2}$$

where $\mathcal{L}^+ = \{j \in \mathcal{L} | y_j > 0\}$, and $\mathcal{N}_m(v) = \{u^m | (v, u^m) \in E\}$.

4.2.3 Inferring from the model and learning from feedback. To infer the next node, RedThread chooses the node with highest overall evidence support, as[1]:

$$j^* \leftarrow \arg\max_j \sum_m s_m^j \theta_m^j \tag{3}$$

Now let node v_j denote this last queried node, for which we observe the label y_j. RedThread adjusts the modality which most supported the selection of node j, i.e. it first determines the support modality,

$$m^* = \arg\max_m s_m^{j^*} \theta_m \tag{4}$$

Then it adjust the importance/credit of the modality m^* as:

$$\theta_m^* = \begin{cases} \delta\theta_m^* & y_{j^*} < 0 \\ (2-\delta)\theta_m^* & y_{j^*} > 0 \end{cases} \tag{5}$$

where $\delta \in (0, 1)$ is the learning rate. This penalizes or rewards the supporting modality based on the positive and negative labels. We note that this learning mechanism is inspired by the weighted majority voting and minimum regret learning.

4.2.4 Initialization. Prior information on the importance of different modalities (e.g. phone number more important than unigrams) could be easily incorporated into the model as the initial values for Θ. When no such information is available, Θ is initialized uniformly. In applications where cases are similar across different seeds, i.e. same evidence types are always important, e.g. phone numbers in our case building, the modality weights learned from one seed could be *transfered* to the next seed as initial parameters.

4.2.5 Re-Weighing the evidence. Given the observed negative labels, one can adjust the weights of different pieces of evidence, assuming some pieces of evidence (within or across different modalities) point to more relevant nodes. To enforce this, we re-weigh the pieces of evidence proportional to how many positive v.s. negative instances they point to. In more detail, instead of Eq. (2) we use:

$$s_m^j = \sum_{v_i \in \mathcal{L}} \sum_{u \in \mathcal{N}_m(v_i) \cap \mathcal{N}_m(v_j)} \frac{\delta(i)}{d_u^2} \tag{6}$$

where $\delta(j) = 1$ if $y_j > 0$ and $\delta(j) = -1$ when $y_j < 0$.

[1]We have experimented with a non-deterministic version that picks nodes with probability proportional to these scores, however the presented deterministic version achieves better performance in practice.

4.2.6 Computational Efficiency and Implementation. RedThread uses local computation similar to local clustering methods. It maintains a priority queue to keep track of the top candidate for expansions (i.e. for computing Eq. (3)). The size of this queue is limited to be much smaller than the actual size of the graph. We observe experimentally than changing the queue size does not affect the performance significantly. RedThread also keeps a shell structure, which maintains the tie strength of nodes surrounding those in the queue. In each iteration that a new label arrives, only the scores for the nodes in the shell are updated, and they might enter the queue based on their updated score.

4.3 Decompositional Baselines

To better evaluate the performance of the RedThread and the effect of different its components, we consider four more baselines descried as follow.

4.3.1 RedThread without Feedback (NF). This baseline uses the scoring scheme described in Section 4.2.2 to expands from the node with the current maximum score, using Eq. (3), but ignores the feedback from the user. The resulted algorithm is similar to a local clustering where expansion is purely based on the connectivity.

4.3.2 RedThread without Parameters (NP). This baseline consider feedback by only expanding from positive nodes, but does not have any parameter learning.

4.3.3 RedThread without Modality Parameters (NM). This baseline uses the feedback to re-weigh the pieces of evidence using the procedure described in Section 4.2.5. However the importance of different modalities is not adjusted by the feedback.

4.3.4 RedThread without Evidence Weights (NE). This baseline uses the feedback to learn the importance of different modalities but *skips* the re-weighing of the evidence (Section 4.2.5).

In the next section, we present a selected set of experiments to showcase the effectiveness of the proposed RedThread.

5 EXPERIMENTS

Here we first describe our datasets, then compare the performance of the RedThread with different baselines introduced in Section 4.

5.1 Datasets

Our data consists of advertisements scraped from escort section of Backpage.com for cities across the United States of America, Canada, and their territories from August 2013 to January 2017. For each advertisement, we have access to its unstructured text (title and body), attached images, date and location posted. Overall, we have about 40 million advertisements. We use a publicly available regular expression extractor, which is developed for the same data previously[11] to extract basic features from the advertisement text, which are: phone number, email, url, name. We also extract unigrams and bigrams used in both title, and body[2]. From these advertisements, we build **three datasets**, one restricted by location, and two by time. One of these time periods is chosen to include activities associated with a list of phone numbers reported to a victim advocacy groups [11]. Table 1 reports the basic summary statistics of these datasets.

		DMV	DJF	SUS
datapoints	advertisements	2,611,636	1,432,066	3,934,482
evidence types	title_unigram	80,006	81,833	133,682
	body_unigram	196,855	211,114	329,398
	title_bigram	546,847	542,437	920,051
	body_bigram	1,474,490	1,548,091	2,437,285
	date(day)	1,629	90	184
	image(hashcode)	1,156,719	588,774	2,338,285
	nickname	10,264	10,973	15,438
	location(city/state)	30	512	511
labels	url	13,557	16,316	25,814
	email	3,854	4,105	6,943
	phone numbers	189,210	215,447	378,403

Table 1: Statistics for Three HT Datasets, namely DMV (DC, Maryland, Virginia), DJF (December 2013, January 2014, February 2014), and SUS (July 2013 to December 2013 with contain suspicious ads). Each row gives the number of unique pieces of evidence for the corresponding type/modality across different datasets. Last three modalities are left out as oracle labels.

	Discogs		MemeTracker	
datapoints	master release	3,515,407	meme	1,531,687
evidence types	date	8082	date	31
	country	267	link	3,514,970
	track	13,078,458	unigram	761,599
	company	322,258	bigram	8,285,661
	genre	15		
	record label	526,441		
	catalog number	2,699,059		
	style	478		
labels	artist	1,181,652	url	140,316
	extra artist	2,076,945		

Table 2: Statistics for Two Public Datasets, namely Discogs and MemeTracker. The first row is corresponds to the main datapoints, and the last row is the modality treated as labels. Middle rows show different features/evidence types in these two datasets (note that we have smaller set of indicators for MemeTracker).

For the evaluation of the algorithms, and since labels are not available in these datasets, **we keep out of the graph some modalities that strongly indicate relations between ads to derive labels**. In more detail, we construct labels by connected components formed when only using url, email, and phone numbers. This means that two ads are assumed 'truly' related only if they share at least one of these hard identifiers[3]. Here, the RedThread will only use the first eight evidence types reported in Table 1 to infer relations. In practice however, RedThread will use all the 11 evidence types available as evidence and hence the performance reported here is a lower-bound on its true performance, since the strongest types of evidence are masked in our experiments to be used as labels.

We also include **two publicly available** datasets in our experiments. This further shows **the generality of RedThread when applied in different domains**. The *first* dataset is the Discogs from KONECT[18] which is collected from a large online music database, and provides information about different releases: date of

[2]Filtering those that appear in more than 10,000 advertisements, to trim out stop words and common phrases. We apply the same cutoff also to other evidence types, e.g. to filter out smiley images, or other common images.

[3]We further split these components when not reachable by the remaining evidence.

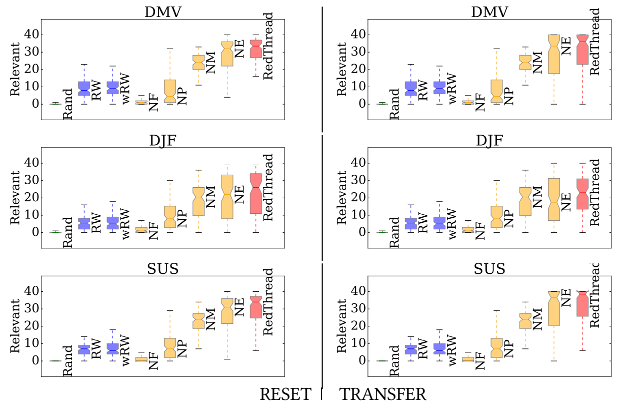


Figure 4: Overall comparison with baselines on the three human trafficking datasets of Table 1. In each boxplot, first three methods (green and blue) show the basic baselines, the next four (yellow) show RedThread without one of its components, and the last one (red) corresponds to the RedThread. In the left column, algorithms reset the parameters per each seed. On the right, weights are transfered to the subsequent seeds.

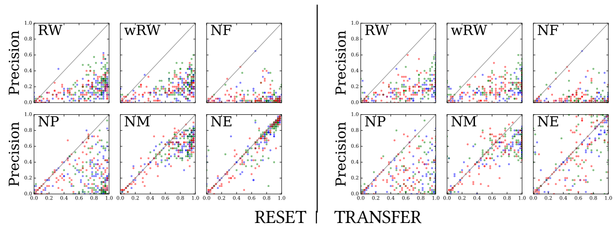


Figure 5: Per seed comparison with baselines. Similar to Fig. 4, we compare when the parameters reset per each seed (left) and when weights are transfered to the subsequent seeds (right). These scatter plots compare performance of RedThread with each algorithm per seed, i.e. *points above the diagonal are the seeds in which the corresponding baseline has higher accuracy than RedThread.* Seeds of the same dataset are marked with the same color, i.e. the green, blue and red colors represent seeds corresponding to the three human trafficking datasets of Table 1.

the release, artists (primary and extra) involved, record labels, track information, companies involved in the production of the release, etc. Full list of entities and their frequencies (number of unique values) is reported in Table 2. In this dataset we treat artists as true labels and consequently *RedThread is learning to find releases of the same artists*[4] as the given seed release based on their shared information. The *second* dataset is MemeTracker from SNAP[22]. This data contains frequent quotes and phrases used in the news. We take a subset of total datasets, i.e. over 1.5M memes posted between April 1th to May 1th 2009. Each meme has the posted date, and a set of links it points to, and we further extract unigram and bigrams from their content. For the true labels, we consider the domain of the url in which the meme is posted. Hence *RedThread is learning to find memes which are originated from the same domain as the given seed*, based on how they share terms, pointers, and temporal co-occurrence.

5.2 Performance Comparison with Baselines

We compare the general performance of RedThread against different baselines discussed in Section 4.1. In particular, Fig. 4, Fig. 5 and Fig. 6 compare the number of relevant entities found by different methods when query budget is fixed to 40. These figures show the distribution of number of relevant entities retrieved for 100 seeds [5].

In Fig. 4 we see that expanding only based on topology and ignoring the user's feedback (NF) is showing very low performance (on par with *Rand*). This would correspond to the performance of an unsupervised local clustering algorithm. Moreover, performance is only slightly better for the random walk variations which use the feedback to restart (*RW*and *wRW*) but ignore the importance of different modalities and don't remember the factors that resulted in reaching a negative. The variations of RedThread which only consider weights on evidence (NM) or modalities (NE) are also not doing as well as when incorporating both in RedThread. We further see that in general transferring weights improves the performance of RedThread in these datasets. The choice to transfer or reset depends on whether the modalities have the same importance across cases or not, which depends on the applications, For the two public datasets we observe a better performance when reseting between seeds, which is reported in Fig. 6. We can also see the precision per seed node in scatter plots of Fig. 5 [6]. Here, the x-axis shows the precision of RedThread and the y-axis shows the precision of the corresponding baseline. We can see that a significant majority of the seeds fall below the x=y line, i.e. in which case the baseline has lower precision compared to RedThread. In this figure, the three colors correspond to the three human trafficking datasets of Table 1.

5.2.1 Filtering Near Duplicates in Escort Advertisements. In the human trafficking datasets it is common to have a large number of near duplicates, i.e. the same advertisement posted repeatedly over time. We **filter out** these duplicates in the results reported in Fig. 4, i.e. we do not consider performance on nodes deemed to be near duplicates of the positives already found. The results reported by the algorithms are passed through a near duplicate detector post-processor and *are only handed to the user for labeling if it is not a near duplicate of the results already labeled.*

We consider two ads to be near duplicates if they share images, have a very similar text, and are advertising the same person(s). In more detail, the inferred advertisement j is compared pairwise with all of the ads the algorithm has returned thus far excluding those that were labeled negative, $i \in \mathcal{L}^-$, and is detected as a near duplicate if it duplicates any of them: j duplicates i, iff it uses the exact set of names as i, shares at least one image with i (when they both contain images), and at least 95% of their bigrams (used in

[4]Note that a release might have more than one artist, hence more than one label. Any two releases with overlapping sets of artists are considered related.

[5]The exact experiment settings for reported results are learning rate of 0.3 and queue size of 1000; however, similar trends are observed for other settings reported in the supplementary materials accompanying the code. To have enough relevant entities to search for, seeds are either selected randomly from ads connected to the largest label components (in Fig. 6), or large enough components (in Fig. 4) selected at random. A label component consists of all connected nodes of the same label which is the ground-truth; in other words, it is a maximal set of datapoints that are relevant and related to each others as defined by Definition 3.1.

[6]refer to Section 5.2.3 for a discussion on computing precision and other quality measures used.

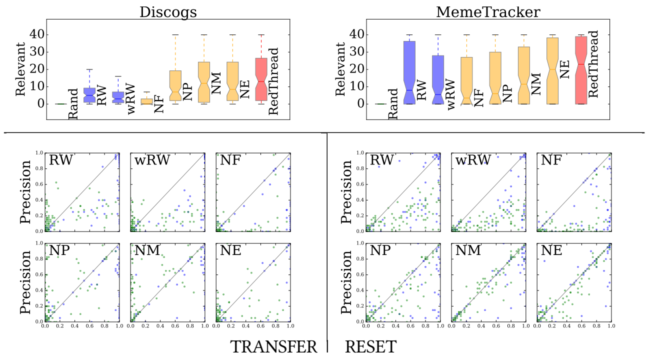


Figure 6: Comparison with baselines on the two public datasets of Table 2. The boxplots correspond to results when weights are reset between different seeds. The bottom scatter plots compare transfer (on the left) with reset (on the right). Similar to Fig. 5, *points above the diagonal are the seeds in which the corresponding baseline has higher accuracy than RedThread*; we can see that in these datasets reseting performs better as the lower triangle is denser in the plots on the right.

title or body) overlaps. A more sophisticated duplicate matcher or entity resolution method could be plugged in here, but this is out of the scope of current paper.

5.2.2 *Skipping High Confidence Queries.*

In the same manner as above, there are ads which the algorithms reach which are very similar to the currently positive set, but are not quite duplicates. Given the limited query budget, it would be better to skip querying the user with these high confidence matches. Note that we are not trying to select the queries for the most information gain, and are still mainly focused on maximizing the number or relevant entities found given the fixed query budget. This number would be boosted significantly if we skip querying those we are almost sure would be positive. Hence we consider a second post-processing procedure which filters queries to user. Similar to the near-duplicate detector, this process performs a pairwise comparison with all of the ads that the algorithm has returned thus far excluding members of $\mathcal{L}^-$, and skips querying a selected ad if it shares more than 90% of its quad-grams with one of the positive labeled ads. This procedure is applied to all the baselines and RedThread in similar fashion to the near duplicator discussed in Section 5.2.1.

Figure 7 shows a significant **boost** in the number of relevant entities found by different algorithm if we skip querying user with high-confidence matches[7]. For example, for most seeds in the DMV dataset, we are finding more relevant entities than the query budget as the median is well above 40. These results are on the same experimental setting as the Fig. 4 and are over the same set of 100 seeds. Again, a more sophisticated procedure could be trained to select the high confidence matches for RedThread, particularly by observing the raw scores the algorithm produces. The development of such matcher is in the future work, here however the current external procedure is preferred since we can apply it to the baselines in the same manner as to RedThread to have a fair comparison.

[7]Note that none of the methods use labels from the high confidence matches, and labels are only used to evaluate the performance, i.e. the number of positive founds for both high confidence matched and queried instances.

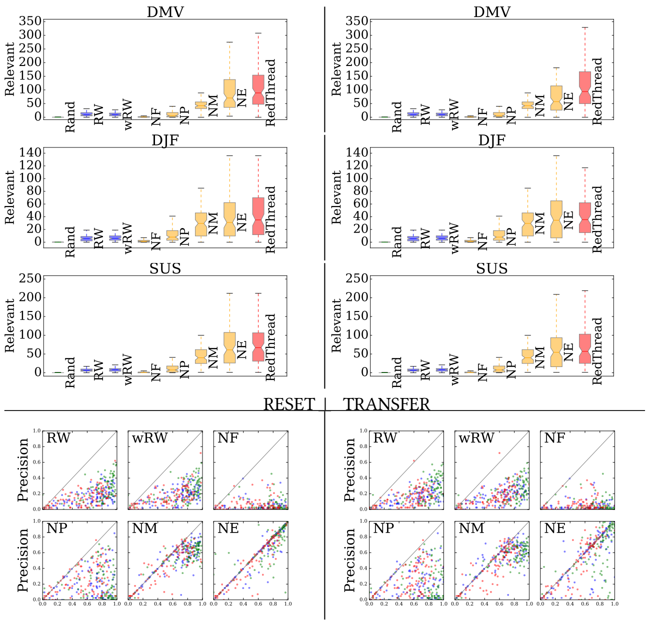


Figure 7: Comparison with baselines on three human trafficking datasets of Table 1 when skipping high confidence matches. We see the maximums are much higher compared to Fig. 4, using the same query budget of 40.

5.2.3 *Precision and Recall.*

When the query budget is fixed and we are not skipping the high confidence matches, i.e. in Fig. 4, to Fig. 6, precision is a scaled version of the number of relevant entities, i.e. divided by the total queried instances (40). However, when skipping the high confidence matches, it is important to look at the precision of the method since the total number of true positives and false positives are no longer constrained to be a constant. Table 3 reports the precision, recall and $F_{0.5}$ scores computed for the different algorithms. These results are averaged over the same set of 100 seed nodes. The top set of basic results correspond to the results of Fig. 4, whereas the bottom set show the change in precision and recall when skipping high confidence matches, i.e. Fig. 7. In Table 3 we see *a significant boost in the recall and f-measure* while the change in precision is negligible. Given this, skipping these high confidence matches seems to be the best strategy.

5.3 Constructing Graphs from Inferred Links

For a given set of positive labeled set, we can track and plot in which order and with what score the advertisements were inferred, to derive a graph of how advertisements are connected. Based on this, we can construct i) a heterogeneous graph with evidence and advertisement nodes, where length two paths show the ads that are connected and how; and ii) a multi-edge graph in which advertisement nodes are connected by edges of different type (evidence); or iii) a simple graph where advertisements are connected by aggregate edges of all the pieces of evidence they share. Figure 2 shows the case in the first form, for the relevant entities discovered for a seed that produced one of the largest clusters in the SUS dataset. From investigating the content of these connected ads, we observe

Method		DMV			DJF			SUS		
		Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
basic	RW	0.17 ± 0.12	0.04 ± 0.04	0.09 ± 0.07	0.23 ± 0.14	0.01 ± 0.01	0.05 ± 0.03	0.15 ± 0.12	0.03 ± 0.03	0.08 ± 0.07
	wRW	0.17 ± 0.11	0.04 ± 0.04	0.09 ± 0.07	0.24 ± 0.13	0.01 ± 0.01	0.05 ± 0.03	0.15 ± 0.11	0.03 ± 0.03	0.08 ± 0.07
	NF	0.04 ± 0.09	0.01 ± 0.02	0.02 ± 0.05	0.05 ± 0.09	0.00 ± 0.01	0.01 ± 0.02	0.06 ± 0.09	0.01 ± 0.02	0.04 ± 0.06
	NP	0.21 ± 0.20	0.05 ± 0.06	0.11 ± 0.12	0.21 ± 0.23	0.01 ± 0.01	0.04 ± 0.05	0.25 ± 0.21	0.06 ± 0.06	0.14 ± 0.13
	NM	0.55 ± 0.20	0.13 ± 0.09	0.30 ± 0.15	0.58 ± 0.18	0.03 ± 0.02	0.12 ± 0.06	0.45 ± 0.25	0.10 ± 0.07	0.25 ± 0.16
	NE	0.73 ± 0.33	0.17 ± 0.13	0.40 ± 0.23	0.70 ± 0.33	0.04 ± 0.02	0.14 ± 0.09	0.47 ± 0.34	0.10 ± 0.09	0.27 ± 0.20
	RedThread	**0.77** ± 0.32	0.18 ± 0.13	0.43 ± 0.22	0.75 ± 0.31	0.04 ± 0.02	0.15 ± 0.09	**0.55** ± 0.30	0.12 ± 0.08	0.30 ± 0.18
high confidence	RW	0.16 ± 0.10	0.04 ± 0.04	0.09 ± 0.07	0.24 ± 0.14	0.01 ± 0.01	0.05 ± 0.04	0.14 ± 0.11	0.03 ± 0.03	0.08 ± 0.07
	wRW	0.18 ± 0.11	0.05 ± 0.06	0.11 ± 0.08	0.24 ± 0.13	0.01 ± 0.01	0.05 ± 0.04	0.16 ± 0.12	0.04 ± 0.04	0.09 ± 0.08
	NF	0.04 ± 0.07	0.01 ± 0.02	0.02 ± 0.04	0.05 ± 0.08	0.00 ± 0.01	0.01 ± 0.02	0.06 ± 0.08	0.01 ± 0.02	0.03 ± 0.05
	NP	0.21 ± 0.19	0.07 ± 0.10	0.13 ± 0.14	0.20 ± 0.20	0.01 ± 0.02	0.05 ± 0.07	0.24 ± 0.18	0.07 ± 0.08	0.15 ± 0.14
	NM	0.53 ± 0.19	0.25 ± 0.20	0.39 ± 0.19	0.56 ± 0.16	0.05 ± 0.04	0.18 ± 0.10	0.43 ± 0.24	0.17 ± 0.16	0.31 ± 0.21
	NE	0.56 ± 0.29	0.35 ± 0.31	0.46 ± 0.29	0.62 ± 0.30	0.12 ± 0.15	0.28 ± 0.24	0.46 ± 0.30	0.20 ± 0.21	0.34 ± 0.25
	RedThread	0.63 ± 0.26	**0.41** ± 0.31	**0.52** ± 0.27	**0.76** ± 0.24	**0.16** ± 0.19	**0.37** ± 0.24	0.52 ± 0.26	**0.22** ± 0.18	**0.38** ± 0.23

Table 3: Performance of RedThread compared with baselines on three human trafficking datasets of Table 1. Average and standard deviation of precision, recall, and f-measure are reported over the 100 seeds.

that they are advertising two different persons (i.e. two different nicknames used with different heights) but are using many shared phrases (mostly bigrams in the body of the text which correspond). These correspond to ads related to two different possible victims), which are part of a bigger organized activity pattern. Note that these ads also share a hard identifier (phone number) which are treated as labels here and are not used as evidence when inferring the connections.

6 CONCLUSION AND FUTURE WORK

In this paper, we defined Active Search of Connections based on its use-case for spotting organized activities in escort advertisements. Active Search of Connections finds **related and relevant** datapoints to a given lead through their shared evidence. We presented RedThread as an efficient solution which searches **locally** by expanding from the lead and learns interactively by querying labels. RedThread considers a **heterogeneous** structure to account for different evidence types. We experimentally compared RedThread performance with different baselines on five different datasets, two of which are publicly available.[8] The code for RedThread is released publicly.

There are multiple lines of future work which we enumerate throughout the paper. This includes improving the post-processing procedures to further enhance recall of RedThread and baselines, and interactive visualization to present the results of RedThread in an accessible form to facilitate the collection of feedback from domain expert investigators. From a research perspective, we also envision the following future objectives: i) flagging suspicious seeds automatically, to generate leads for RedThread, using unsupervised pattern and anomaly detection methods. ii) incorporating additional databases as evidence modalities, e.g. bitcoin wallets, social media information, etc; iii) incorporating features on specific nodes in the inference, e.g. indicators on advertisement's textual component: third person style, authorship, etc; and iv) using graph databases to allow fast online storage and analysis of the data.

[8]Releasing collections of human trafficking data requires careful and through anonymization to prevent (re-)victimization of subjects, even if such data has been first published openly online. We are investigating privacy and technical concerns involved in that as a separate research project which might lead to a future release of such data.

ACKNOWLEDGMENTS

This work was supported in part by the Defense Advanced Research Projects Agency (FA8750-14-2-0244), National Institute of Justice (2013-IJ-CX-K007), and National Science Foundation (1320347). The authors would like to thank Marinus Analytics for providing insights on counter human trafficking practice, and the members of the CMU Auton lab for helpful discussions and support, in particular, Dr. Kyle Miller, Benedikt Boecking, and Chirag Nagpal.

REFERENCES

[1] Nir Ailon, Yudong Chen, and Huan Xu. 2015. Iterative and active graph clustering using trace norm minimization without cluster size constraints. *Journal of Machine Learning Research* 16 (2015), 455–490.

[2] Hamidreza Alvari, Paulo Shakarian, and J. E. Kelly Snyder. 2016. A non-parametric learning approach to identify online human trafficking. In *ISI*. IEEE, 133–138.

[3] Hamidreza Alvari, Paulo Shakarian, and J. E. Kelly Snyder. 2017. Semi-Supervised Learning for Detecting Human Trafficking. *CoRR* abs/1705.10786 (2017).

[4] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: a generic approach to entity resolution. *The VLDB JournalâĂŤThe International Journal on Very Large Data Bases* 18, 1 (2009), 255–276.

[5] Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 5.

[6] Mustafa Bilgic, Louis Licamele, Lise Getoor, and Ben Shneiderman. 2006. D-dupe: An interactive tool for entity resolution in social networks. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*. IEEE, 43–50.

[7] Hsinchun Chen, Wingyan Chung, Jialun Qin, Edna Reid, Marc Sageman, and Gabriel Weimann. 2008. Uncovering the dark Web: A case study of Jihad on the Web. *JASIST* 59, 8 (2008), 1347–1359.

[8] Hong Cheng and Jeffrey Xu Yu. 2012. Clustering Large Attributed Graph. *JIP* 20, 4 (2012), 806–813.

[9] Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. 2011. Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities. *TKDD* 5, 2 (2011), 12:1–12:33.

[10] Wei Dong, Moses Charikar, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*. ACM, 577–586.

[11] Artur Dubrawski, Kyle Miller, Matthew Barnes, Benedikt Boecking, and Emily Kennedy. 2015. Leveraging publicly available data to discern patterns of human-trafficking activity. *Journal of Human Trafficking* 1, 1 (2015), 65–85.

[12] Roman Garnett, Yamuna Krishnamurthy, Donghan Wang, Jeff Schneider, and Richard Mann. 2011. Bayesian optimal active search on graphs. In *Ninth Workshop on Mining and Learning with Graphs*.

[13] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff G. Schneider, and Richard P. Mann. 2012. Bayesian Optimal Active Search and Surveying. In *ICML*. icml.cc / Omnipress.

[14] Joseph J. Pfeiffer III, Sebastián Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2014. Attributed graph models: modeling network structure with correlated attributes. In *WWW*. ACM, 831–842.

[15] Joseph John Pfeiffer III, Jennifer Neville, and Paul N. Bennett. 2014. Active Exploration in Networks: Using Probabilistic Relationships for Learning and

Inference. In *CIKM*. ACM, 639–648.

[16] Hiroshi Kajino, Akihiro Kishimoto, Adi Botea, Elizabeth M. Daly, and Spyros Kotoulas. 2015. Active Learning for Multi-relational Data Construction. In *WWW*. ACM, 560–569.

[17] Mayank Kejriwal and Pedro Szekely. 2017. Information Extraction in Illicit Web Domains. In *WWW*. ACM, 997–1006.

[18] KONECT 2017. The Koblenz Network Collection. http://konect.uni-koblenz.de/. (2017).

[19] Renata A. Konrad, Andrew C. Trapp, Timothy M. Palmbach, and Jeffrey S. Blom. 2017. Overcoming human trafficking via operations research and analytics: Opportunities for methods, models, and applications. *European Journal of Operational Research* 259, 2 (2017), 733–745.

[20] Valdis E Krebs. 2002. Mapping networks of terrorist cells. *Connections* 24, 3 (2002), 43–52.

[21] Mark Latonero. 2011. Human trafficking online: The role of social networking sites and online classifieds. (2011).

[22] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (jun 2014).

[23] Jure Leskovec, Kevin J. Lang, and Michael W. Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *WWW*. ACM, 631–640.

[24] Jialu Liu, Chi Wang, Marina Danilevsky, and Jiawei Han. 2013. Large-Scale Spectral Clustering on Graphs. In *IJCAI*. IJCAI/AAAI, 1486–1492.

[25] Byron Marshall, Hsinchun Chen, and Siddharth Kaza. 2008. Using importance flooding to identify interesting networks of criminal activity. *JASIST* 59, 13 (2008), 2099–2114.

[26] Yutaka Matsuo, Junichiro Mori, Masahiro Hamasaki, Keisuke Ishida, Takuichi Nishimura, Hideaki Takeda, Kôiti Hasida, and Mitsuru Ishizuka. 2006. POLYPHONET: an advanced social network extraction system from the web. In *WWW*. ACM, 397–406.

[27] Kyle Miller, Emily Kennedy, and Artur Dubrawski. 2016. Do Public Events Affect Sex Trafficking Activity? *arXiv preprint arXiv:1602.05048* (2016).

[28] Fabricio Murai, Diogo Rennó, Bruno Ribeiro, Gisele L. Pappa, Don Towsley, and Krista Gile. 2018. Selective harvesting over networks. *Data Min. Knowl. Discov.* 32, 1 (2018), 187–217.

[29] Chirag Nagpal, Kyle Miller, Benedikt Boecking, and Artur Dubrawski. 2017. An Entity Resolution Approach to Isolate Instances of Human Trafficking Online. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. 77–84.

[30] NCMEC 2011. Human Trafficking Investigation Hearing. http://www.missingkids.com/Testimony/11-19-15. (2011).

[31] NCMEC 2014. NCMEC vs. Backpage. http://www.missingkids.com/en_US/documents/AMICUS_NCMEC_Backpage.pdf. (2014).

[32] California Department of Justice. 2016. Human Trafficking. https://oag.ca.gov/new-press-categories/human-trafficking. (2016).

[33] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. 2014. Focused clustering and outlier detection in large attributed graphs. In *KDD*. ACM, 1346–1355.

[34] Rebecca S. Portnoff, Danny Yuxing Huang, Periwinkle Doerfler, Sadia Afroz, and Damon McCoy. 2017. Backpage and Bitcoin: Uncovering Human Traffickers. In *KDD*. ACM, 1595–1604.

[35] Xiang Ren, Meng Jiang, Jingbo Shang, and Jiawei Han. 2017. Constructing Structured Information Networks from Massive Text Corpora. In *WWW (Companion Volume)*. ACM, 951–954.

[36] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. 2017. CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases. In *WWW*. ACM, 1015–1024.

[37] Renata Konrad, and Andrew C. Trapp 2017. Data Science Can Help Us Fight Human Trafficking. https://www.scientificamerican.com/article/data-science-can-help-us-fight-human-trafficking/. (2017). Originally on The Conversation US.

[38] Purnamrita Sarkar, Andrew W. Moore, and Amit Prakash. 2008. Fast incremental proximity search in large graphs. In *ICML (ACM International Conference Proceeding Series)*, Vol. 307. ACM, 896–903.

[39] Daniel A. Spielman and Shang-Hua Teng. 2013. A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26.

[40] Pedro A. Szekely, Craig A. Knoblock, Jason Slepicka, Andrew Philpot, Amandeep Singh, Chengye Yin, Dipsy Kapoor, Prem Natarajan, Daniel Marcu, Kevin Knight, David Stallard, Subessware S. Karunamoorthy, Rajagopal Bojanapalli, Steven Minton, Brian Amanatullah, Todd Hughes, Mike Tamayo, David Flynt, Rachel Artiss, Shih-Fu Chang, Tao Chen, Gerald Hiebel, and Lidia Ferreira. 2015. Building and Using a Knowledge Graph to Combat Human Trafficking. In *International Semantic Web Conference (2) (Lecture Notes in Computer Science)*, Vol. 9367. Springer, 205–221.

[41] Thorn. 2015. Report on the Use of Technology to Recruit, Groom and Sell Domestic Minor Sex Trafficking Victims. https://27l51l1qnwey246mkc1vzqg0-wpengine.netdna-ssl.com/wp-content/uploads/2015/02/Survivor_Survey_r5.pdf. (2015).

[42] Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating Human Trafficking with Deep Multimodal Models. *CoRR* abs/1705.02735 (2017).

[43] Xuezhi Wang, Roman Garnett, and Jeff G. Schneider. 2013. Active search on graphs. In *KDD*. ACM, 731–738.

[44] Jennifer Jie Xu and Hsinchun Chen. 2004. Fighting organized crimes: using shortest-path algorithms to identify associations in criminal networks. *Decision Support Systems* 38, 3 (2004), 473–487.

[45] Jennifer Jie Xu and Hsinchun Chen. 2005. Criminal network analysis and visualization. *Commun. ACM* 48, 6 (2005), 100–107.

[46] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph Clustering Based on Structural/Attribute Similarities. *PVLDB* 2, 1 (2009), 718–729.

[47] Yilu Zhou, Edna Reid, Jialun Qin, Hsinchun Chen, and Guanpi Lai. 2005. US Domestic Extremist Groups on the Web: Link and Content Analysis. *IEEE Intelligent Systems* 20, 5 (2005), 44–51.