Discovering Context for Real-World Events

La découverte du contexte de faits réels

Arjun Satish[1], Ramesh Jain[2], Amarnath Gupta[3]

[1] Confluent Inc, Palo Alto, CA, arjun.satish@gmail.com
[2] University of California, Irvine, CA, jain@ics.uci.edu
[3] SDSC, University of California, San Deigo, gupta@sdsc.edu

ABSTRACT. Specifying the search space is an important step in designing multimedia annotation systems. With the large amount of data available from sensors and web services, context-aware approaches for pruning search spaces are becoming increasingly common. In these approaches, the search space is limited by the contextual information obtained from a fixed set of sources. For example, a system for tagging faces in photos might rely on a static list of candidates obtained from the photo owner's Facebook profile. These contextual sources can become extremely large, which leads to lower accuracy in the annotation problem.

We present **Context Discovery Algorithm**, a technique to ***progressively*** discover the most relevant search space from a dynamic set of context sources. This allows us to reap the benefits of context, while keeping the size of the search space within limits.

As a concrete application for our approach, we present a simple photo management application, which tags faces of people in a user's personal photos. We empirically study the role of our framework in the face tagging application to tag photos taken at real-world events, such as conferences, weddings or social gatherings. Our results show that the availability of event context, and its dynamic discovery, can produce 97.5% smaller search spaces with at least 93% correct tags.

RÉSUMÉ. Spécifier un espace de recherche représente une étape importante dans la conception de systèmes d'annotation multi-média. Avec une grande quantité de données provenant de capteurs et de services Web, les approches sensible au contexte deviennent de plus en plus usuelles pour élaguer les espaces de recherche. Dans ces approches, l'espace de recherche est limité par les informations contextuelles qui sont obtenues à partir d'un ensemble donné de sources. Par exemple, un système pour marquer les visages dans des photos pourrait reposer sur une liste statique de candidats obtenus à partir de photos de personnes sur leur profil FaceBook. Ces sources contextuelles peuvent devenir très volumineuses, ce qui peut conduire à une précision plus faible dans le problème des annotations.

Nous présentons un nouvel algorithme de découverte du contexte, une technique pour découvrir progressivement l'espace de recherche le plus pertinent pour un ensemble dynamique de sources contextuelles. Ceci nous permet de recueillir les bénéfices du contexte tout en gardant la taille de l'espace de recherche à une taille raisonnable. Comme concrète application de notre approche, nous présentons une application simple de management de photos, où les visages de personnes sont marqués à partir de photos privées d'un utilisateur. Nous étudions empiriquement le rôle de notre cadre de travail dans l'application de marquage de visages pour marquer des photos prises lors d'événements sociaux comme des conférences, des mariages ou de rassemblements sociaux. Nos résultats montrent que la disponibilité du contexte des événements et sa découverte dynamique peut produire des espaces de recherche plus petits de 97.5% avec au moins 93% de marquages corrects.

KEYWORDS. Context-based reasoning, Search Space Pruning.
MOTS CLÉS. Raisonnement basé sur le contexte, élagage d'un espace de recherche, reconnaissance de visages.

1. Introduction

With the popularity of global social networks and proliferation of mobile phones, information about people, their social connections and day-to-day activities are becoming available at a very large scale. The web provides an open platform for documenting many real world events such as conferences, weather events and sports games. With such context sources, multimedia annotation algorithms [15, 19, 21] are being designed where the search space of tags is obtained from one or more sources (figure 1.1(b)). These approaches rely on a single *type* of context. For example, using social network information from Facebook to solve the face recognition problem. We refer to such a direct dependency between the search space and a data source as *static linking*. Although these systems are meritorious in their own right, they suffer from the following drawbacks: they do not employ multiple

sources, and therefore the *relations* between them. By realizing that these sources are interconnected in their own way, we are able to treat the entire source topology as a network. Our intuition in this work is to navigate this network to progressively discover the search space for a given media annotation problem. Figure 1.1(c) shows how context discovery can provide substantially smaller search spaces for a set of images, which contain a large number of correct tags. A small search space with large number of true positives provides the ideal ground for an annotation algorithm to exhibit superior performance.

We present the CueNet framework, which provides access to multiple data sources containing event, social, and geographical information through a unified query interface to extract information from them. CueNet encapsulates our *Context Discovery Algorithm*, which utilizes the query interface to discover the most relevant search space for a media annotation problem. To facilitate a hands-on discussion, we show the use of context discovery in a real world application: face tagging in personal photos. As a case study, we will attempt to tag photos taken at conference events, weddings and social gatherings (birthday parties, for example) by different users. These photos could contain friends, colleagues, relatives or friends-of-friends or newly found acquaintances (who are not yet connected to the user through any social network). Real world event photos are particularly interesting because no single source can provide all the necessary information. It emphasizes the need to utilize multiple sources in a meaningful way.

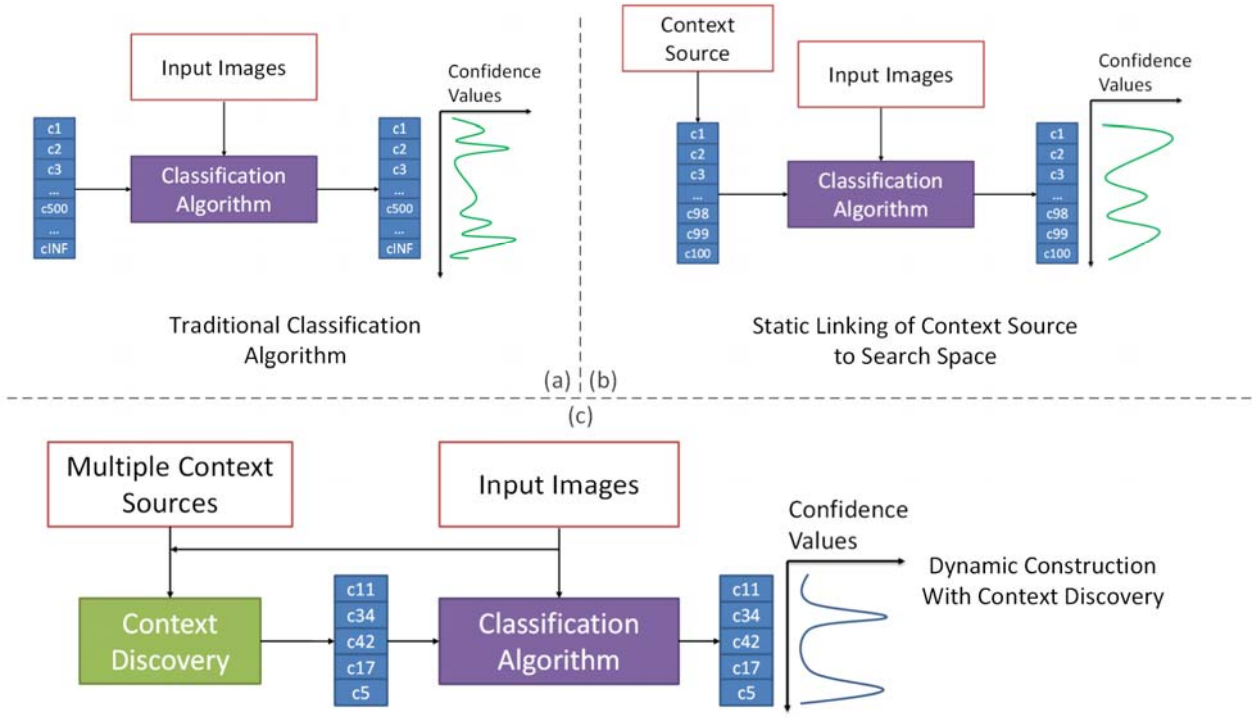

Figure 1.1. *The different approaches in search space construction for a multimedia annotation problem. A traditional setup, where the search space is manually specified is shown in (a). Context is used in (b), to generate large static search spaces. The CueNet framework in (c) produces small relevant search spaces.*

Here is an example to illustrate CueNet's discovery process. Let's suppose Joe takes a picture with a camera that records time and GPS in the photo's EXIF header (figure 1.2). Additionally, Joe has two friends. One with whom he interacts on Google+, and the other using Facebook. The framework checks if either of them has any interesting event information pertaining to this time and location. We find that the friend on Google+ left a calendar entry describing an event (a title, time interval and name of the place). The entry also marks Joe as a participant. In order to determine the category of the place, the framework uses Yelp.com with the name and GPS location to find whether it is a restaurant, sports stadium or an apartment complex. If the location of the event was a sports stadium, it navigates to upcoming.com to check what event was occurring here at this time. If a football game or a music

concert was taking place at the stadium, we look at Facebook to see if the friend "Likes" the sports team or music band. By traversing the different data sources in this fashion, the number of people, who could potentially appear in Joe's photograph, was incrementally built up, rather than simply reverting to everyone on his social network or people who could be in the area where the photograph was taken. We refer to such navigation between different data sources to identify relevant contextual information as **progressive discovery**. The salient feature of CueNet is to be able to progressively discover events, and their associated properties, from the different data sources and relate them to the photo capture event. We argue that given this structure and relations between the various events, CueNet can make assertions about the presence of a person in the photograph. Once candidates have been discovered by CueNet, they are passed to the face tagging algorithm ([12], for example), which can perform very well as their search space is limited to two candidates.

Contributions: Real-world search spaces are large and complex (because of their time varying relationships). Our contribution in this paper is a technique to discover the search spaces for multimedia annotation problems by using contextual information (events and their interrelations) from multiple data sources. We claim that this search space is significantly smaller than one obtained by static linking approaches, but retains a high number of true positives. We describe our findings when these ideas are applied to a personal photo annotation problem.

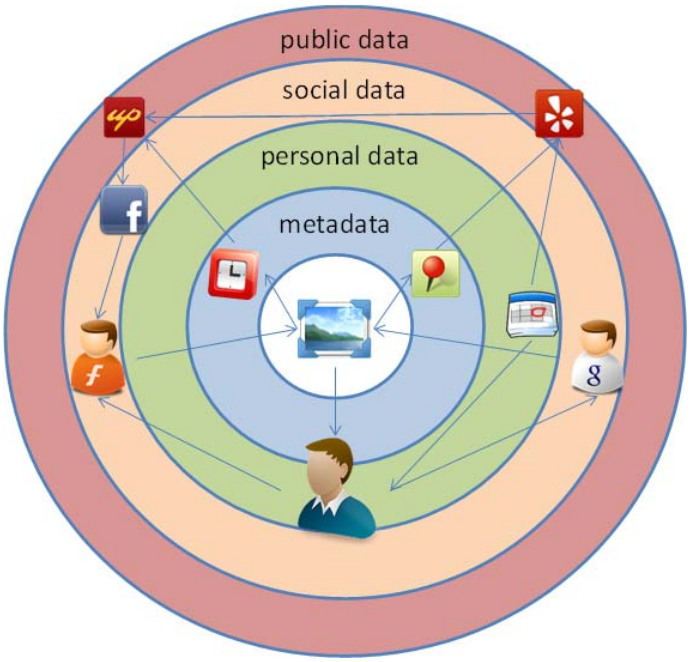

Figure 1.2. *Navigation of various data sources by the discovery algorithm.*

In the following sections, we develop our notion of context, and identify its properties which make discoveries like the above possible. We discuss the CueNet framework, its different components, and the conditions it creates which allow for progressive discovery. We present a context discovery algorithm to use these properties and tag faces in personal photos. Finally, we present an empirical evaluation to support our above claims.

2. Context

Photo capture is a real world event. If we had a perfect description of the world, a photo annotation problem simply becomes associating faces in the photo to the people who are related to the event. For example, the people who posed for a photo, or are participating in an event during which the photo was taken. Computationally, we can view the real world containing a large number of objects (for example, people and places), interacting with each other during events. The context for any entity in the real-world is the state of all the other entities in it. Effectively, it ***is the set of relationships which exist***

between all the objects and events at a point in time. For the photo capture event, all the relationships between objects and other events at the time of capture is its context. The important implication of this observation is that relationships change with time. Events mutate with time, and objects change relationships. What is context at a time for an entity, is completely different from what is context at a later time. For a given problem, some of these relationships might be more relevant than others, but in this work, is classified as context.

Unfortunately, a complete description of the world is far from available. But, we have disparate sources of information which provide a specific vantage into all the entities. Examples of data sources range from mobile phone call logs and email conversations to Facebook messages to a listing of public events at upcoming.com. Sources can be classified into **Personal Data Sources**: include all sources which provide details about the particular user whose photo is to be tagged (for example, email and Google Calendar); **Social Data Sources**: such as Facebook, DBLP or LinkedIn which provide contextual information about a user's friends and colleagues, or **Public Data Sources** which provide information about public events.

Social and public data sources are enormous in size, containing information about billions of events and entities. Trying to use them directly will lead to scalability problems faced by face recognition and verification techniques. But, by using personal data, we can discover which parts of social and public sources are more relevant. For example, if a photo was taken at San Francisco, CA (where the user lives) his family in China is less relevant. Thus, the role of personal information is twofold. **Firstly**, it provides contextual information regarding the photo. **Secondly**, it acts as a bridge to connect to social and public data sources to discover interesting people connected to the user who might be present in the event and therefore, the photo.

We must note the **temporal relevance** property of a data source. Given a stream of photos taken during a time interval, the source which contributed interesting context for a photo might not be equally useful for the one appearing next. This is because sources tend to focus on a specific set of event types or relationship types, and the two photos might be captured in different events or contains persons with whom the user maintains relations through different sources. For example, two photos taken at a conference might contain a user's friends in the first, but with advisers of these friends in the next. The friends might interact with the user through a social network, but their advisers might not. By using a source like DBLP, the relations between the adviser and friends can be discovered. We say that the temporal relevance of these context sources is ***low***. This requirement will play an important role in the design of our framework. On the other hand, there are sources which indicate a ***high*** temporal context. For example, social networks contain photo albums which were created during a birthday party which tend to contain the same people over photos.

It is hard to predefine the structure of context if the relationships between entities change with time. Hence, these relationships needs to be ***discovered for every moment of time***. For example, conference events always have talks. But it is not possible to prescribe whether some attendees will decide to have an adhoc design discussion during the conference. Given the disparate nature of data sources, it is also hard to predict precisely which data source will provide relevant context. In the example from section 1, given a photo from Joe, should we check the conferences data source or the sports data source to check what event Joe might be participating in?

In the following sections, we will look at how the CueNet framework uses the different data sources to discover the context for a given photo.

3. The CueNet Framework

Figure 3.1 shows the different components of the CueNet framework.

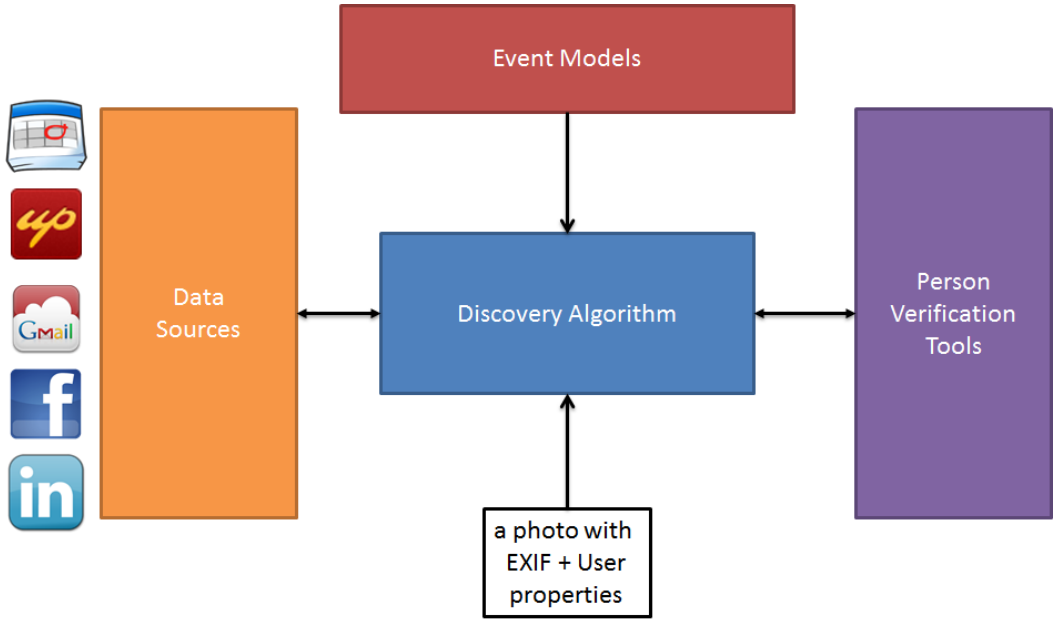

Figure 3.1. *The Conceptual Architecture of CueNet.*

The Ontological **Event Models** specify various event and object classes, and the different relations between them. These declared types are used to define the **Data Sources** which provides access to different types of contextual data. The **Person Verification Tools** consist of a a face tagging algorithm, trained from a database of annotated photos of people's faces (for example, [12]). When this module is presented with a candidate and the input photograph, it compares the features extracted from the candidate's photos and the input photo to determine if the candidate is present in the photo or not. It must be noted that our work does propose improvements in the face verification algorithm itself. Instead we aim to improve its efficacy by reducing it's search space by determining the photo's context. In this section, we describe each module, and how the context discovery algorithm utilizes them to accomplish its task.

3.1. *Event Model*

Our ontologies extend the E* model [8] to specify relationships between events and entities. Specifically, we utilize the relationships "**subevent-of**", which specifies event containment. An event *e1* is a subevent-of of another event *e2*, if *e1* occurs completely within the spatiotemporal bounds of *e2*. Additionally, we utilize the relations "**occurs-during**" and "**occurs-at**", which specify the space and time properties of an event. Also, another important relation between entities and events is the "**participant**" property, which allows us to describe which object is participating in which event. It must be noted that participants of a subevent are also participants of the parent event. A participation relationship between an event and person instance asserts the presence of the person within the spatiotemporal region of the event. We argue that the reverse is also true, i.e., if a participant P is present in location L_P during the time T_P and an event E occurs within the spatiotemporal region (L_E, T_E), we say P is a participant of E if the event's spatiotemporal span contained that of the participant.

$$\texttt{participant}(\mathrm{E},\,\mathrm{P}) \iff (\mathcal{L}_P \sqsubset_L \mathcal{L}_E) \wedge (\mathcal{T}_P \sqsubset_T \mathcal{T}_E) \quad [1]$$

The symbols $\sqsubset_L$ and $\sqsubset_T$ indicate spatial and temporal containment [8]. In later sections, we refer to the location and time of the event, L_E and T_E as E.**occurs-at** and E.**occurs-during** respectively.

3.2. *Data Sources*

The ontology makes available a vocabulary of classes and properties. Using this vocabulary, we can now declaratively specify the schema of each source. With these schema descriptions, CueNet can infer what data source can provide what type of data instances. For example, the framework can distinguish between a source which describes conferences and another which is a social network. We use a LISP like syntax (as shown in figure 2.2) to allow developers of the system to specify these declarations. The example below describes a source containing conference information.

The source declaration consists of a s-expression, where the source keyword indicates a unique name for the source. The **:attributes** keyword is used to list the attributes of this source. The **:relation** keyword constructs the instances conf, time, loc, attendee which are of conference, time-interval, location and person class types respectively, and relates them with relations specified in the ontology. Finally, the **:mapping**s are used to map nodes in the relationship graph constructed above to attributes of the data source. For example, the first mapping (specified using the map keyword) maps the conference's time-interval object (t) to the (time) attribute of the source.

```
(:source conferences
(:attributes name time title)
(:relation conf type-of conference)
(:relation t type-of time-interval)
(:relation attendee type-of person)
(:relation attendee participant-in conf)
(:relation conf occurs-during t)
(:mappings [ [t time] [conf.title title] [attendee.name name] ]))
```

Figure 2.2. *Defining a source for conference events.*

3.2. *Conditions for Discovery*

CueNet is entirely based on reasoning in the event and object (i.e., person) domain, and the relationships between them. These relationships include participation (event-object relation), social relations (object-object relation) and subevent relation (event-event). For the sake of simplicity, we restrict our discussions to events whose spatiotemporal spans either completely overlap or do not intersect at all. We do not consider events which partially overlap. In order to develop the necessary conditions for context discovery, we consider the following two axioms:

Object Existence Axiom: Objects can be present in one place at a time only. The object cannot exist outside a spatiotemporal boundary containing it.

Participation Semantics Axiom: If an object is participating in two events at the same time, then one is the subevent of the other.

Given, the ontology O, we can construct event instance graph $G^I(V^I, E^I)$, whose nodes are instances of classes in C^O and edges are instances of the properties in P^O. The context discovery algorithm relies on the notion that given an instance graph, *queries* to the different sources can be automatically constructed. A query is a set of predicates, with one or more unknown variables. For the instance graph $G^I (V^I, E^I)$, we construct a query $Q(D, U)$ where D is a set of predicates, and U is a set of unknown variables.

Query Construction Condition: Given an instance graph G^I (V^I, E^I) and ontology $O(C^O, P^O)$, a query $Q(D, U)$ can be constructed, such that D is a set of predicates which represent a subset of relationships specified in G^I. In other words, D is a subgraph induced by G^I. U is a class, which has a relationship $r \in P^O$, with a node n ∈ D. Essentially, the ontology must prescribe a relation between some node n through the relationship r. In our case, the relation r will be either a **participant** or **subevent** relation. If the relationship with the instances does not violate any object property assertions specified in the ontology, we can create the query $Q(D, U)$.

Identity Condition: Given an instance graph $G^I(V^I, E^I)$, and a result graph $G^R(V^R, E^R)$ obtained from querying a source, we can merge two events only if they are identical. Two nodes $v^I_i \in V^I$ and $v^R_r \in V^R$ are identical if they meet the following two conditions **(i)** Both v^I_i and v^R_r are of the same class type, and **(ii)** Both v^I_i and v^R_r have exactly overlapping spatiotemporal spans, indicated by the $=_L$ and $=_T$. Mathematically, we write:

$$v_i^I = v_r^R \iff (v_i^I.\textbf{type-of} = v_r^R.\textbf{type-of}) \wedge (v_i^I.\textbf{occurs-at} =_L v_r^R.\textbf{occurs-at}) \wedge (v_i^I.\textbf{occurs-during} =_T v_r^R.\textbf{occurs-during}) \quad [2]$$

Subevent Condition: Given an instance graph $G^I(V^I, E^I)$, and a result graph $G^R(V^R, E^R)$ obtained from querying a source, we can construct a subevent edge between two nodes $v^I_i \in V^I$ and $v^R_r \in V^R$, if one is spatiotemporally contained within the other, and has at least one common E**ndurant (**a time-independent entity (for example, a person) which is in contrast to a perdurant, which is a time dependent entity (for example, a conference event)).

$$v_i^I \sqsubset_L v_r^R, \quad v_i^I \sqsubset_T v_r^R \quad [3]$$

$$v_i^I.\textbf{Endurants} \cap v_r^R.\textbf{Endurants} \neq \{\phi\} \quad [4]$$

Here v^I_i. Endurants is defined as a set {w | ∈ V^I ∧ w.**type-of** = **Endurant**}. If equation [4] does not hold, we simply say that v^I_i and v^R_r co-occur.

Merging Event Graphs: Given the above conditions, we can now describe an important building block for the context discovery algorithm: the steps needed to merge two event graphs. An example for this is shown in figure 3.3(b-d). Given the event graph consisting of the photo capture event on the left of (b) and a meeting event m and conference event c, containing their respective participants. In this example, the meeting event graph, m is semantically equivalent to the original graph. But the conference event, c is telling that the person AG is also participating in a conference at the time the photo was taken. The result of merging is shown in (d). An event graph merge consists of two steps. The first is a **subevent hierarchy join**, and the second is a **prune-up** step.

Given an original graph, O_m, and a new graph N_m, the join function works as follows: All nodes in N_m are checked against all nodes in O_m to find identical counterparts. For entities, the identity is verified through an identifier, and for events, equation [2] is used. Because of the object existence and participation semantics axioms, all events which contain a common participant are connected to their respective super event using the subevent relation (equations 3 and 4 must be satisfied by the events).

Also, if two events have no common participant, then they can be still be related with the subevent edge, if the event model says it is possible. For example, if in a conference event model, keynotes, lunches and banquets are declared as known subevents of an event. Then every keynote event, or banquet event to be merged into an event graph is made a subevent of the conference event, if the equation 3 holds between the respective events. It must be noted that node *AG* occurs twice in graph (c). In order to correct this, we use the participation semantics axiom. We traverse the final event graph from the leaves to the root events, and remove every person node if it appears in a subevent. This is the **prune-up** step. Using these formalisms, we now look at the working of the context discovery algorithm.

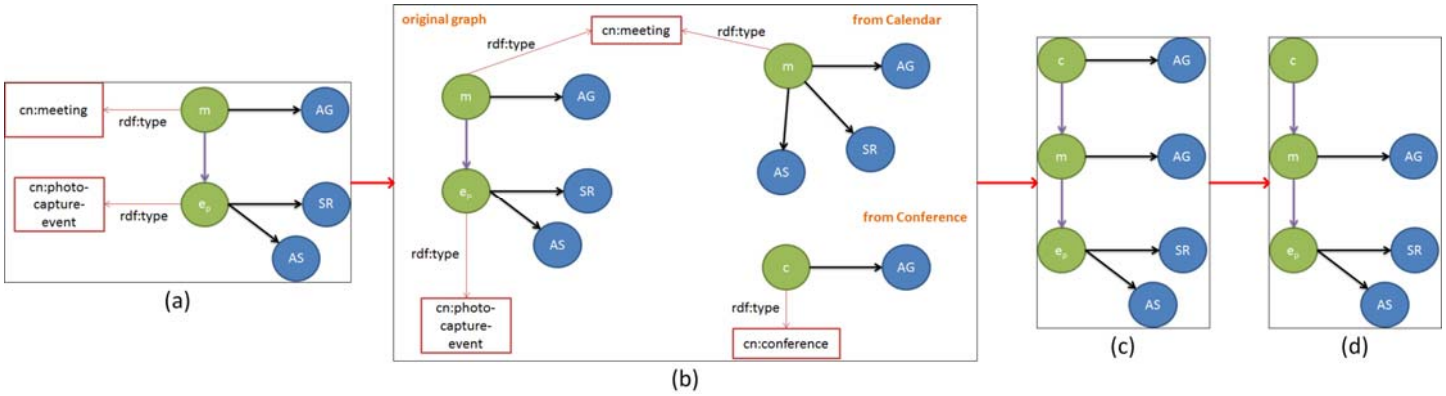

Figure 3.3. *The different stages in an iteration of the context discovery algorithm; (a) shows an example event graph describing a photo taken at a meeting. The meeting consists of three participants AG, SR and AS. The photo contains SR and AS. (b) shows two events returned from the data sources. One is a meeting event which is semantically identical to the input. The other is a conference event with AG. (c) shows the result of merging these graphs. (d) The prune-up function removes the duplicate reference to AG.*

3.3. *Context Discovery Algorithm*

Algorithm 1 (in figure 3.4) below outlines the discovery algorithm, denoted as the **discover** function. The function is tail recursive, invoking itself until a termination condition is reached (the condition here being when at most *k* tags are obtained for all faces or no new data is obtained from all data sources for all generated queries). The input to the algorithm is a photo (with EXIF tags) and an associated owner (the user). It must be noted that by seeding the graph with owner information, we bias the discovery towards his/her personal information. An event instance graph is created where each photo is modeled as a photo capture event. Each event and object is a node in the instance graph. Each event is associated with time and space attributes. All relationships are edges in this graph. All EXIF tags are literals, related to the photo with data property edges. Figure 3.3 graphically shows the main stages in a single iteration of the algorithm.

The event graph is traversed to produce a queue of event and object nodes, which we shall refer to as DQ (discovery queue). The algorithm consists of two primary functions: **query** and **merge**. The behavior of the query function depends on the type of the node. If the node is an event instance, the function consults the ontology to find any known sub-events, and queries data sources to find all these subevents, its properties and participants of the input event node. On the other hand, if it is an object instance, the function issues a query to find all the events it is participating in.

Results from data source wrappers are returned in the form of event graphs. These event graphs are merged into the original event graph by taking the following steps. First, it identifies **duplicate** events using the conditions mentioned above. Second, it identifies subevent hierarchies using the graph merge conditions described above, and performs a **subevent hierarchy join**. Third, the function **prune_up** removes entities from an event when its subevent also lists it as a participant node. Fourth, **push_down**

is the face verification step if the number of entities in the parents of the photo-capture events is small (less than *T*).

Algorithm 1: The Context Discovery Algorithm

Data: A photograph H, with a set of detected faces F. Voting threshold, T. The owner O of the photo.
Result: For each face f ∈ F, a set of atmost k person tags.

```
1  begin
2
3      function discover(): {
4          while (DQ is not empty): {
5              node = DQ.deque()
6              results = query (node)
7              E ← merge (E, results)
8              if (termination_check()):
9                  return prepare_results();
10         }
11         reconstruct DQ ← E
12         discover()
13     }
14
15     function merge(O, N): {
16         remove_duplicates()
17         M ← subevent_hierarchy_join(O, N)
18         prune_up(M)
19         if (less than T new candidates were discovered):
20             push_down(M)
21         else:
22             vote_and_verify(M)
23         return M;
24     }
25
26     E ← construct event graph with H and O
27     construct discoverable nodes queue, DQ ← E
28     return discover()
29 end
```

Figure 3.4. *The Context Discovery Algorithm.*

The **push_down** step will try to verify if any of the newly discovered objects are present in the photo and if they are (if the tagging confidence of this object, obtained from the face verification algorithm, is higher than the given threshold), the objects are removed from the super event, and linked to the photo capture event as its participant. In other words, they are pushed down the subevent hierarchy. Alternatively, if the number of new objects is larger than *T*, the algorithm initiates the **vote-and-verify** method, which ranks all the candidates based on social relationships with people already identified in the photo. For example, if a candidate is related to two persons present in the photo through some social networks, then its score is 2. Ordering is done by simply sorting the candidate list by descending order of score. The face verification runs only on the top ranked k candidates. If there are still untagged faces after the termination of the algorithm, we vote over all the remaining people, and return the ranked list for each untagged face.

Figure 3.3 shows the various stages in the algorithm graphically. (a) shows an example event graph describing a photo taken at a meeting. The meeting consists of three participants AG, SR and AS. The photo contains SR and AS. (b) shows two events returned from the data sources. One is a meeting

event which is semantically identical to the input. The other is a conference event with AG. (c) shows the result of merging these graphs. (d) The **prune-up** function removes the duplicate reference to AG.

3.4. *Merging Context Networks*

In this section, we look more closely at the merge function. Algorithm 2 (in figure 3.5) presents the pseudo-code for merging a secondary context network, *S*, into a primary context network *P*. A terminology of primary and secondary is to signify that all data instances from a secondary network will be merged into a primary network. While merging networks, we also assume that events have at most one super-event. Thus, no diamond structures are found in either network. The algorithm below shows the steps needed to merge two networks each with a single root. A root event is one which has no super-events. The symbol $\forall_{se}$ stands for "for all subevents".

Algorithm 2: The Merge Algorithm

```
Data: Context Network P and S, where S will be merged into P.
Result: All event instances in S will be merged into P.
1  begin
2
3      function merge(P, S): {
4          [Pr] = root event of P
5          [Sr] = root event of S
6          recursive_merge(P, Pr, S, Sr)
7      }
8
9      function recursive_merge(P, Pr, S, Sr): {
10         addAsSubevent = true
11         containedSubevents ← {φ}
12         ∀se (ps of P) {
13             if (equals(ps, Sr)) {
14                 addAsSubevent = false
15                 mergeInformation(ps, Sr)
16                 ∀se (s of Sr): recursive_merge(P, ps, S, se)
17             }
18             if (contains(ps, Sr)) {
19                 addAsSubevent = false
20                 recursive_merge(P, ps, S, Sr)
21             }
22             if (contains(Sr, ps)) {
23                 addAsSubevent = false
24                 containedSubevents.add(ps)
25             }
26         }
27
28         if (addAsSubevent) {
29             addSubeventEdge(Pr, Sr)
30             if (containedSubevents.size() > 0) {
31                 removeSubevents(Pr, containedSubevents)
32                 createSubevents(Sr, containedSubevents)
33             }
34             ∀se (s of Sr): recursiveMerge(subtree, Sr, other, s);
35         }
36     }
37
38 end
```

Figure 3.5. *Algorithm to merge two context networks.*

Once the two root nodes, *Pr* and *Sr* have been identified, we descend the subevent trees of the two context networks, and do one of the following operations. For each subevent of *Pr*, we check if any subevent is equal to *Sr*, then merge the information from *Sr* to this subevent, and continue recursively merging the children of *Sr* into the children of the subevent. If a subevent of *Pr* contains the new

instance, *Sr*, we simply continue recursion. But, if *Sr* can contain the subevent node, then we add of the siblings sub-events which can become subevents of *Sr* to a list **containedSubevents**, and add *Sr* as a subevent of *Pr*, remove all the children from *Pr* in **containedSubevents**, and add them as subevents of **Sr**. Recursion is continued on the children of **Sr** from the newly connected **Sr** node in the primary network. Any new literal properties, spatio-temporal attributes and participant information in events which exist in the primary networks are copied using the **mergeInformation** function (which simply copies attributes from one node to the other).

4. Experiments and Analysis

In this section, we analyze how CueNet helps tags personal photos taken at different events. For a given user, we will construct a dataset consisting of photos taken at a particular event. For each of these users, we will create a candidate set by aggregating people over personal, public and social data sources. In order to evaluate CueNet, we will attempt to reduce this candidate set, and analyze how many of the faces can be correctly tagged by this reduced set. We will also compare this performance over techniques like location based ranking, where candidates are ranked according to their last known location. Our final conclusion is that, in order to rank candidates for tagging faces in photos, CueNet provides an event-agnostic platform, where as other techniques perform inconsistently across different types of events.

4.1. *Setup*

We use photos taken during three different types of events: Social Parties, Academic Conferences and Trips. This diversity allows for different distributions of face tags across different aspects of a user's life. Social Parties generally tend to have close friends who are spatially co-located. Conferences tend to have people from different parts of the world, but those who are affiliated with the area of the conferences. Trips cannot always rely on location as a useful metric and can involve people from either social, personal or professional circles from a user's life.

For each event type, we collect multiple datasets from 6 different people. A dataset consists of multiple photos during the event, the user's personal information, which contains information from sources like Google Calendar, personal email and profile information from social networks like Facebook and Twitter. We also collect a person's social networking information which consists of tweets written by the user or their friend during the time the event was occurring, the social network itself (friends on Facebook and Twitter, along with their profile information). Conference proceedings are downloaded from DBLP and the conference website. Facebook events are also obtained and stored in our database. Besides, location databases like Yahoo Placefinder were used to geocode addresses and reverse geocode EXIF GPS coordinates. We assume that all photos have a valid EXIF tag, especially the timestamp and GPS coordinates. This assumption is not a very hard one, as almost all photos captured in the last two years are through iPhone or Android smartphones, which add reasonably accurate GPS tags and accurate timestamps (where the phone clock is synced with the cell tower). The ground truth was annotated by the user with our annotation interface. For each photo, an annotation consisted of the ID of the person in the candidate set in it. Face verification was achieved initially with Face.com, but after their service was discontinued, we used the web service, Automatic Face Systems, maintained by Neeraj Kumar (of University of Washington), based on the work described in [12].

We use 1889 photos taken at 17 different events in our face tagging experiment. Each photo contains one or more faces. We will denote each dataset as `Di' (where $1 \leq i \leq 17$ for each dataset). Table 3.1 describes each dataset in terms of number of photos, unique annotations in ground truth, the year they were captured and the type of the event. The total number of people who could have been present in the picture is 1894 (i.e., the total search space). The ground truth here means a list of labels created by the owner of the photo album labeling the faces in the photos.

Figure 4.1 shows the distribution of the ground truth annotations of the conference datasets across various sources, for each dataset. For example, the bar corresponding to D2 says that 87.5% of ground truth annotations were found in event sources, 41.67% in social networks, 4.17% in personal information and 12.5% were not found in any source, and therefore marked as "Out of Context Network". From this graph it is clear that event sources contain a large portion of ground truth annotations. Besides D4, a minimum of 70% of our annotations are found in event sources for all datasets, and for some datasets (D3, D7) all annotations are found in event sources. The sum total of contributions will add up to values more than 100% because they share some annotations among each other. For example, a friend on Facebook might show up at a conference to give the keynote talk.

First, we look at how CueNet reduces the number of possible candidates for all photos in a dataset. For this setup, the complete search space, contains 1894 labels (total number of people present at the conference, user's emails and social graph). The figure 4.2 shows various statistics for each photo in dataset D2, which includes the maximum size of the list which was generated by the discovery algorithm, the actual number of people in the photos, the number of true positives and false positives. As it can be seen, the size of the discovered set *S*, never exceeded 12. This is 0.5% of the original candidate list. Because the total number of possible participants (list size) was low, our False Positive rate (FP) were very low too. Most of the false positives were due to profile orientation of faces or obstructions (this was because the face detector was smart enough to pick up profile faces, but verification worked better only on frontal faces).

Dataset	Unique People	No. of Photos	Year	Event Type
D1	23	78	2012	Conference
D2	44	108	2012	Conference
D3	6	16	2010	Conference
D4	7	10	2010	Conference
D5	36	80	2009	Conference
D6	18	63	2013	Conference
D7	7	11	2013	Conference
D8	12	25	2009	Conference
D9	14	65	2011	Party
D10	13	131	2010	Party
D11	6	85	2008	Party
D12	50	74	2012	Party
D13	19	330	2009	Party
D14	14	363	2009	Trip
D15	2	208	2010	Trip
D16	4	217	2011	Trip
D17	7	23	2011	Trip

Table 4.1. *Profile of datasets used in the experiments*

Alternatively, the numbers below can be interpreted as tagging the event itself, and not individual photos. We do this modification to reduce the number of graphs drawn per experiment. We define a **hit**

as a correctly tagged face. Figure 4.3 shows the hits for datasets using the context discovery algorithm. The blue bars show the number of unique faces in the dataset. As we can see the number of hits is comparable to the number of unique faces, which implies that our context discovery algorithm was able to find the correct face tags for almost all datasets.

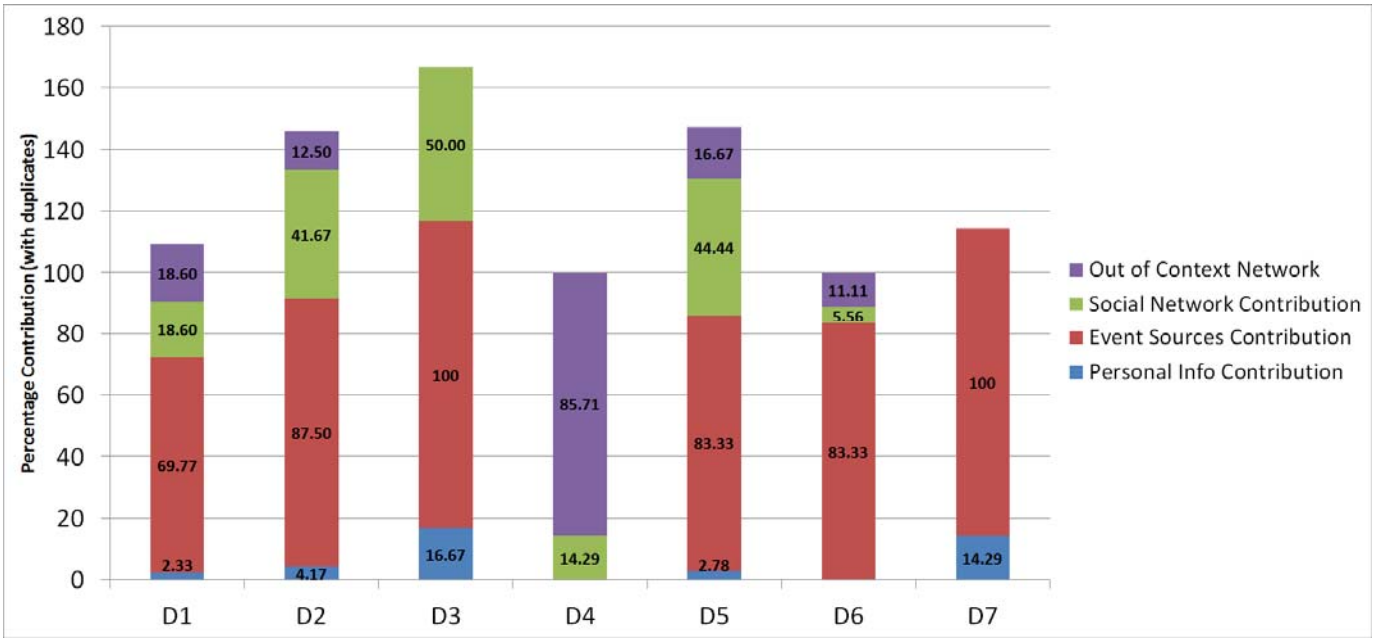

Figure 4.1. *The distribution of annotations in the ground truth for conference photos across various sources.*

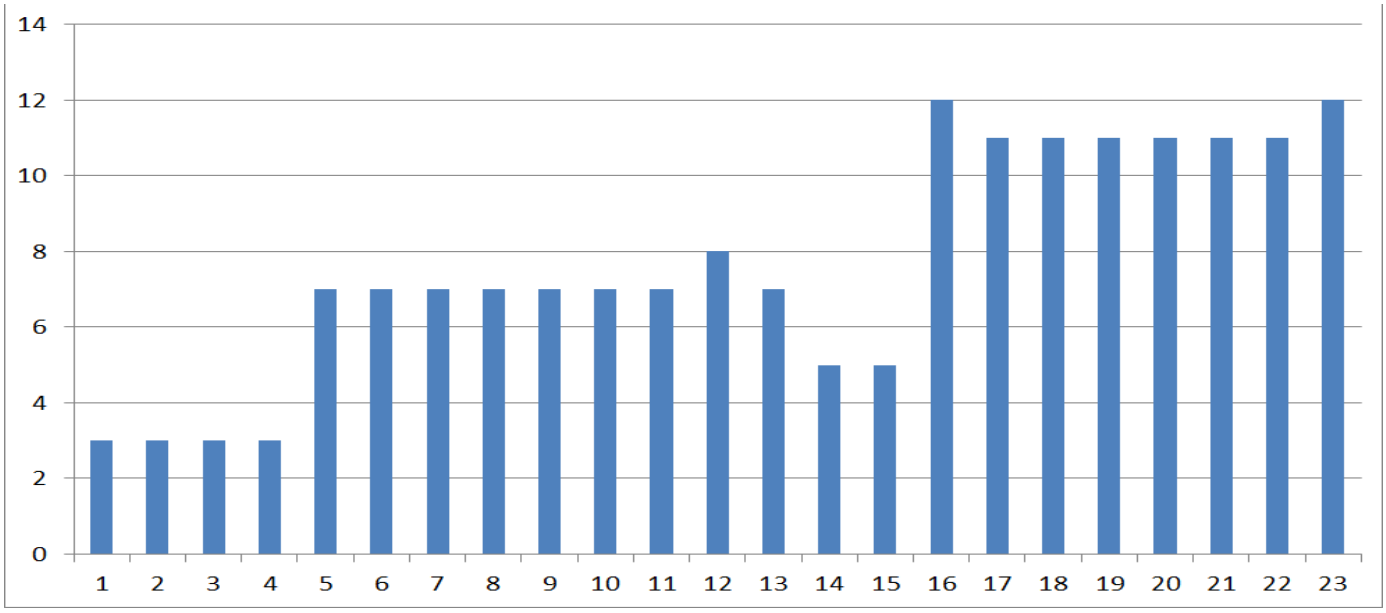

Figure 4.2. *The number of candidates in the context network of photos in dataset D2.*

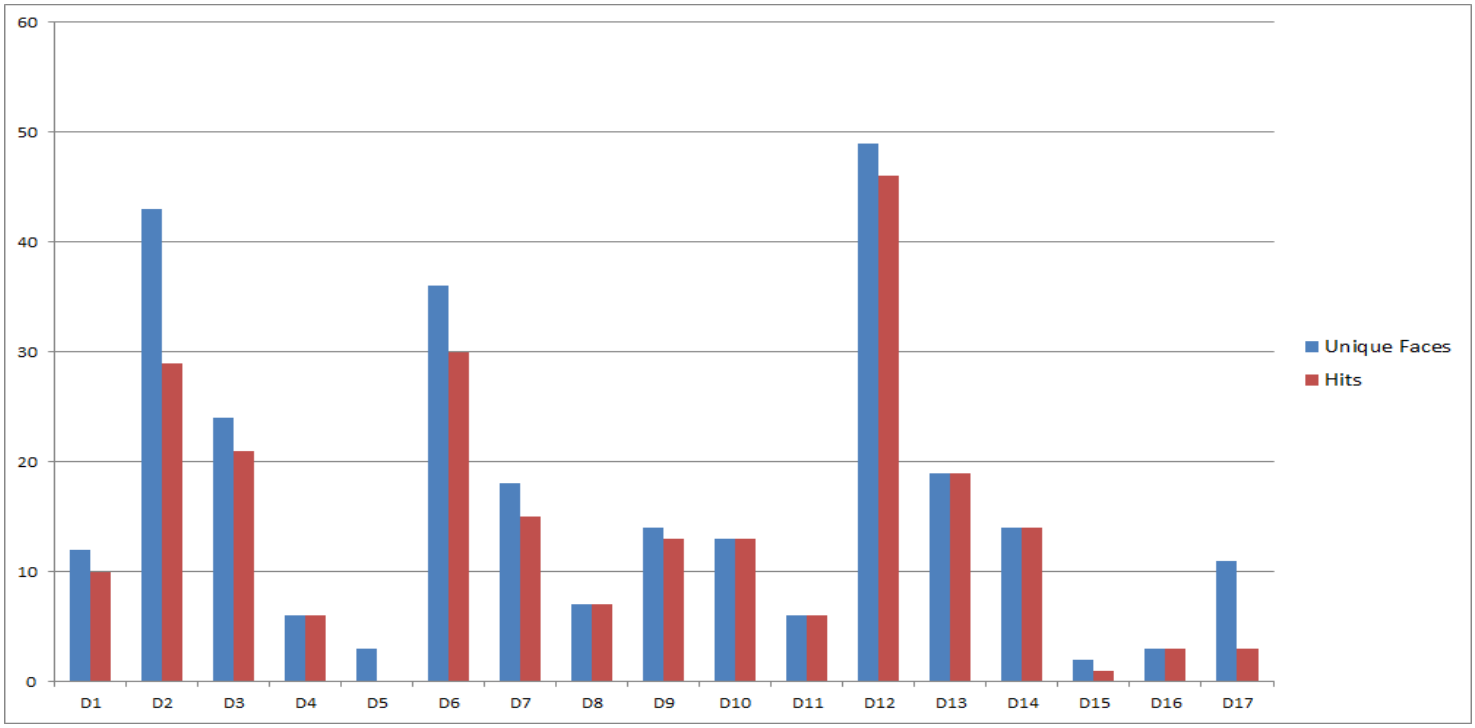

Figure 4.3. *The number of unique faces and hit counts as found by the discovery algorithm for all datasets.*

Most of the datasets contain hundreds of candidates. If we have to perform face verification on all datasets, then the discovery algorithm is not performing effectively. The fewer the number of verifications, the better the pruning power of the algorithm. In order to quantify this, we introduce the metric *Verification Ratio*, which is simply the number of verifications done per candidate in the Candidate Set. If this number is equal to 100%, that means we have performed a very large number of verifications (once per every candidate in the search space, which means no candidates were pruned). The closer it is to 0, the fewer the number of verifications that were done. This ratio also enables us to compare the differences across dataset, where the number of candidates varies. As it can be seen in figure 3.4 the verification ratio never exceeds 18.3%.

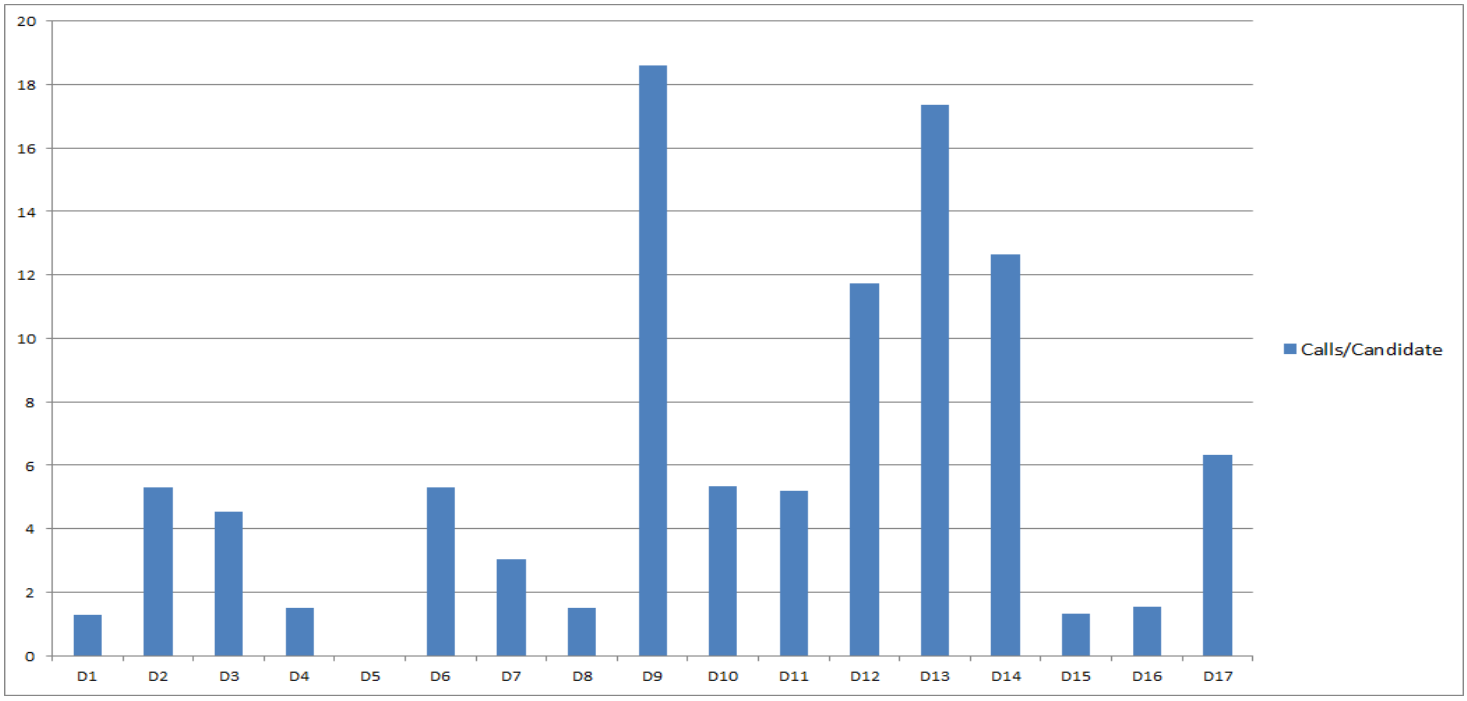

Figure 4.4. *Verification ratio (%) obtained using the context discovery algorithm.*

For a context discovery algorithm to be effective, it must demonstrate hit counts as high as possible (up the number of unique faces) at the same time maintaing low verification ratios.

In order to compare these results, we perform a tagging experiment using location information alone. Candidates are ordered according to their last known location, and presented to the user in the same style as the information presented with the context discovery algorithm. In this case, it must be noted that some users do not have location information related to them, and are excluded from the candidate set. The effect of this fact is seen in the reduced hit count seen using location information.

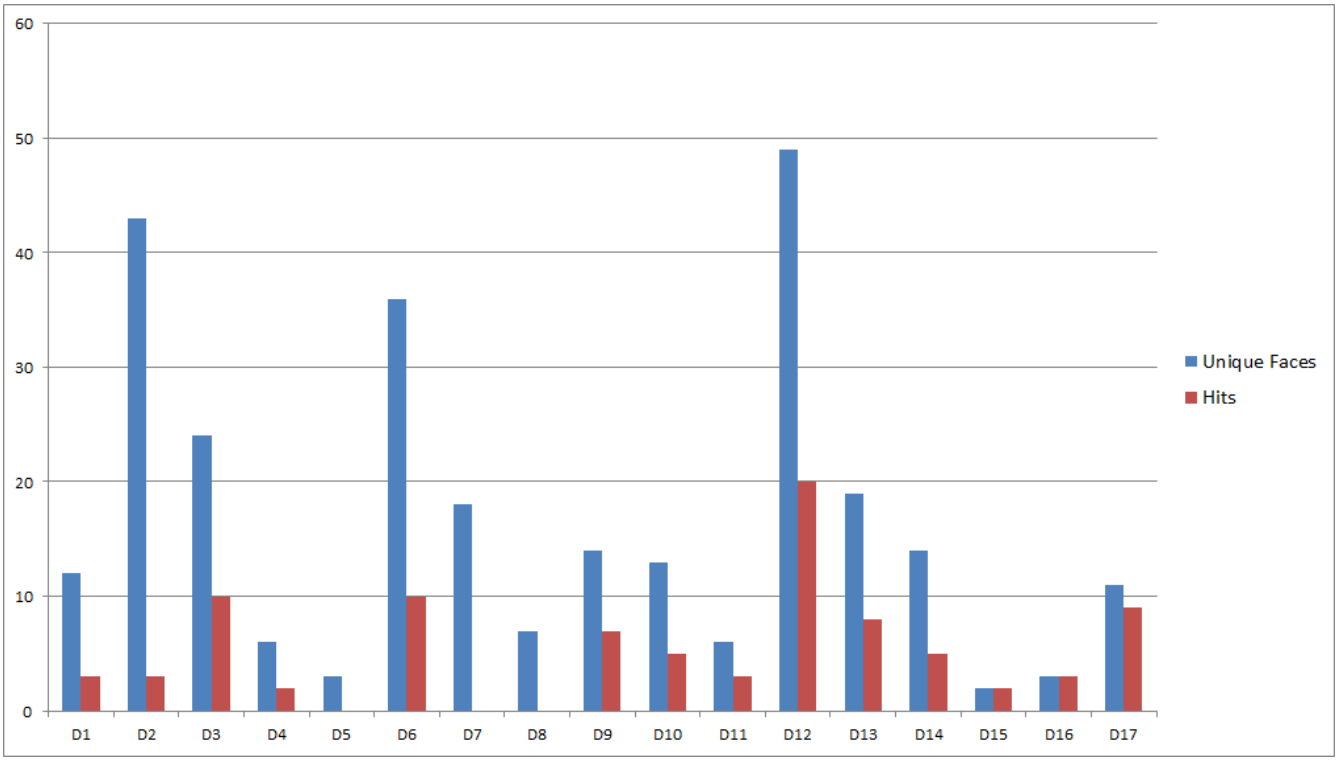

Figure 4.5. *Hit counts for all datasets using location only.*

Figure 4.5 shows the hit counts when using location information only. The number of hits is far lower than the number of unique faces in each dataset. In some cases, there is absolutely no location data at all for any of the candidates. This shows the relative weakness of relying on a single type of information.

Similarly, we measure the verification ratio using location only. We see an adverse effect in in figure 4.6, which shows that the maximum ratio has now increased to almost 29%.

In figure 4.7, we compare the **hits ratio** of the context discovery algorithm with the location based algorithm. The hit ratio is the number of hits per unique face in a dataset. As it can be seen, the context discovery algorithm outperforms the simple location based ranking algorithm. The more important insight in this graph is that the context discovery algorithm performs consistently across different types of events, whereas the location based metric is good only for the party events. This was because in such social gatherings, participants lived closed to the location of the event. And therefore, ranked very highly when only location was used as context. Figure 4.8 compares the different verification ratios. The relatively lower numbers for the context discovery algorithm indicate its superiority over the location based algorithm. Again, the important thing is to note its **superior performance across event types**.

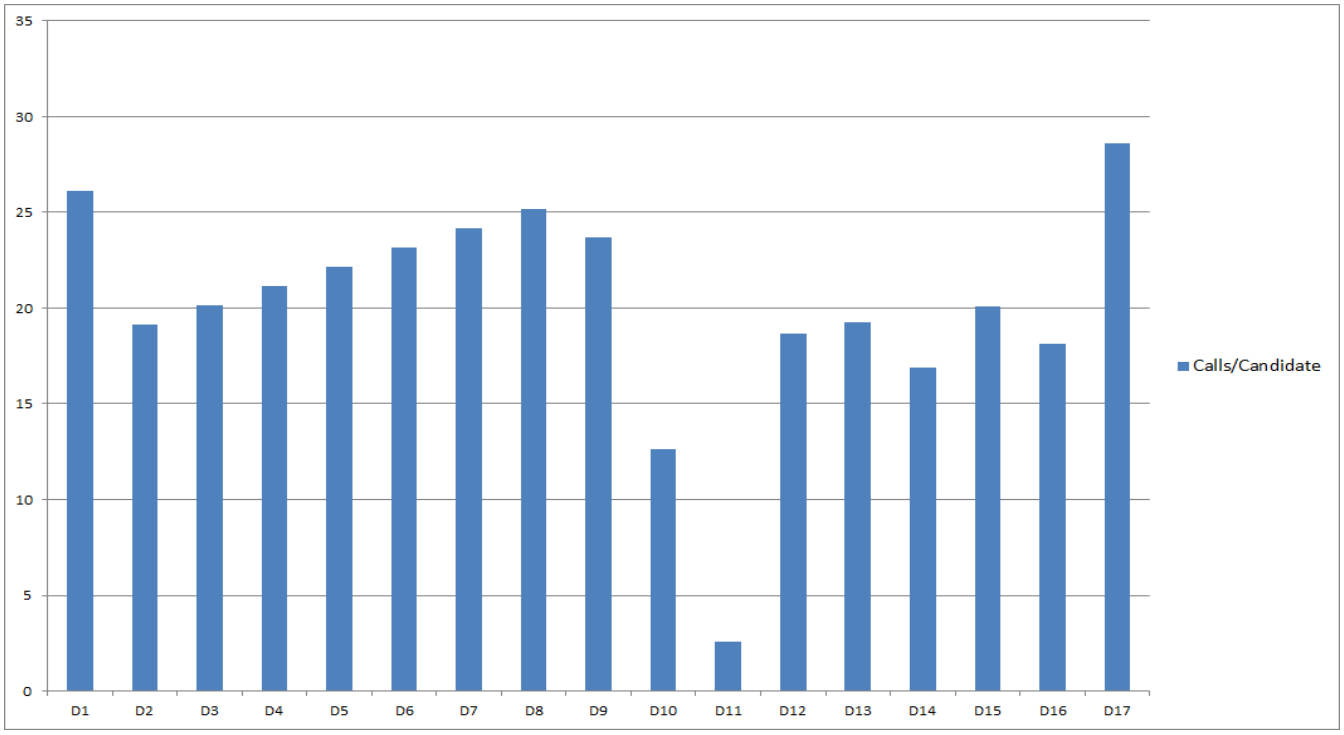

Figure 4.6. *Verification ratio (%) for all datasets obtained using location only.*

4.2. *Importance of Individual Sources*

So far, we have used different sources, without evaluating their individual contributions to the discovery process. In order to empirically study this, we will perform the tagging experiments without each one of the sources. And for each such run of the algorithm, tabulate the hit counts. This will give us an insight into the contribution of each source. Figure 4.9 plots the hit counts for two conference datasets(D2, D8), two party (D9, D12) and two trip (D14, D16) datasets from our original collection. The hit counts are normalized to 100, to allow for easy comparison. In the graph, the source 'Facebook' collectively refers to event and social network information obtained from the website. The source 'Conference' refers to the keynotes, sessions, lunch/coffee breaks and talk information obtained from conference proceedings. For each dataset, figure 4.7 plots the hit counts when one of the sources is absent.

Figure 4.9 shows that the Twitter source is least important for most datasets. This could be because a variety of our users do not use the twitter service during their personal events. Personal events also do not have hashtags because of which it becomes hard for correlate which tweets correspond to events the user is participating. During the presence of such hashtags, their role becomes more prominent (in dataset D2), but not by a large amount. The conference source is most useful to tag photos taken at conferences. But shows no contributions otherwise. This is understandable as the source is curated exclusively for such events, and therefore shows high value for them.

The role of Facebook and Email are the most interesting parts of this experiment. The Facebook social network and their events repository are extremely valuable in tagging photos. Almost all datasets lose a portion of their tags if this source is not considered. Although, it does not affect all datasets, but on average, worse damage is caused when photos are tagged without email sources. The average loss being 31.013% for emails and 26.2184% for Facebook. This difference was due to the fact that people almost always correspond with emails before or after an event. As we will see in the next section, there are certain reasons which make Email even more valuable than Facebook for our current application. It should be noted that we are computing the hit count at the end of all iterations. It might happen that a source provides valuable context in the first iteration, which leads to discovering many tags in the later iterations. Although this source itself provided very little context, its utility was very high.

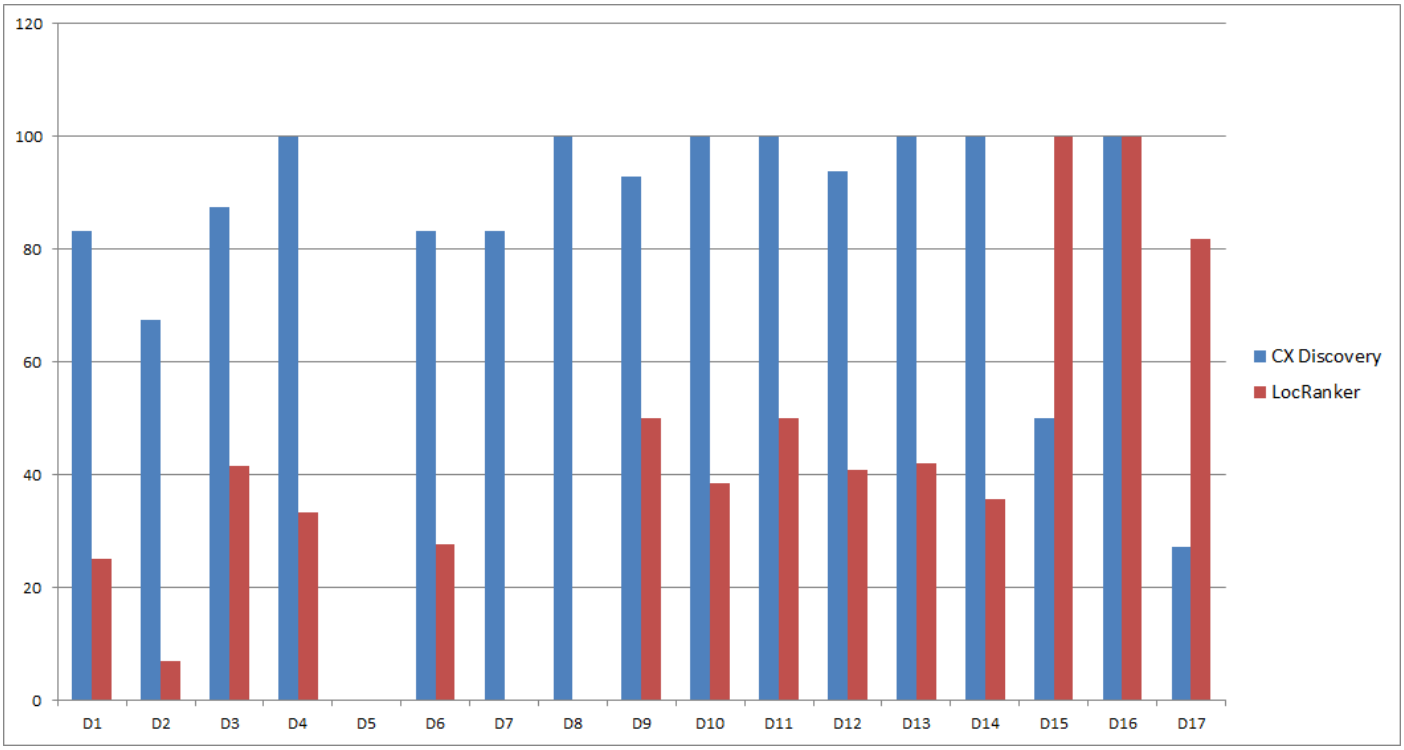

Figure 4.7. *Comparing hits ratio (%) for all datasets (higher is better).*

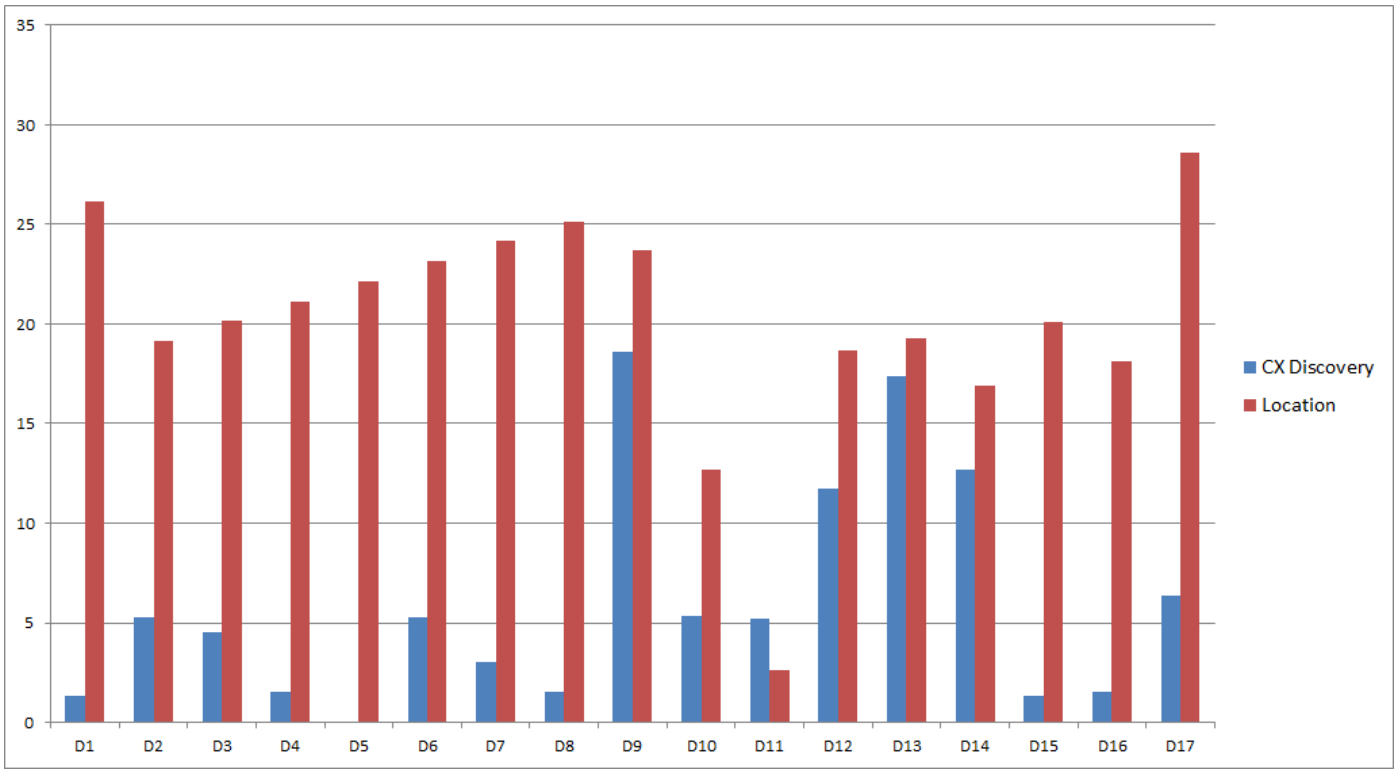

Figure 4.8. *Comparing verification ratio (%) for all datasets (lower is better).*

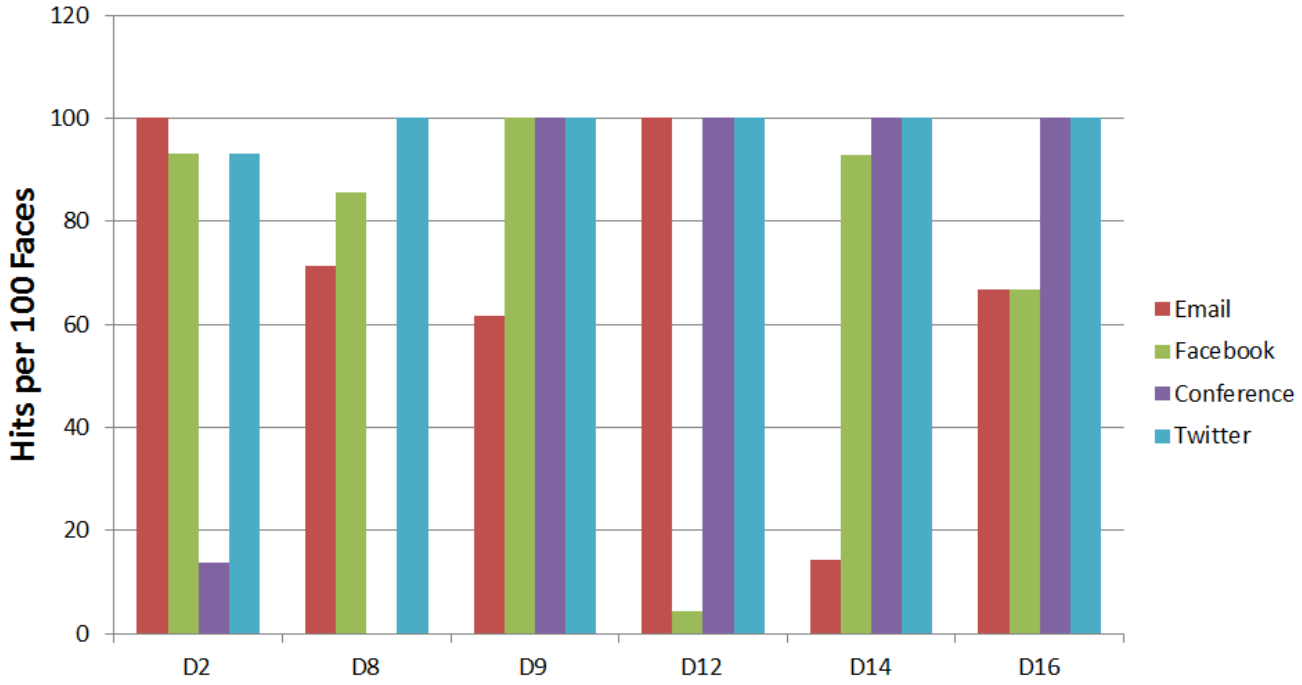

Figure 4.9. *Reduction in hit counts observed as a data source is removed.*

5. Known Issues

In this section, we list some of the issues encountered in designing and building CueNet. Some of these are active areas of research, and whereas others are specific to our framework, and can be considered potential areas of research. Our experience with CueNet indicates that the following issues should be approached in a holistic manner, i.e., in conjunction with each other. Approaching these problems in the "context" of each other reduces the individual complexity of each sub-problem by possibly increasing the complexity of the entire framework, but making the problem more tractable.

Noise in Social Media: The problem of noise filtering in web data is a prominent one, and is being addressed by various communities in different ways. These problems get trickier because of the different variations in representing tiny details such as representation names of people, addresses of places, and time. Anthroponomastics indicates that these differences arise due to cultural, historical and environmental issues. Such issues are non-trivial and are beyond the scope of our work. Face tags in social media sources like Facebook can also be very noisy. This strictly prohibits directly using this data to train verification/recognition models. Also, the average quality of photos is poor, resulting in weaker features, which would have otherwise allowed better matching.

Data collection: is a primary requirement for running the experiements described here. Even though the user simply had to sign into their sources from our UI, it was hard to get them to do so, especially when we requested them for personal email. Privacy concerns are often brought up during these setup discussions with users. Different data sources have different integration methods. For example, email uses a text based IMAP/POP3 protocol, whereas Twitter provides a modern REST API to access data. The data sources abstract their technicalities behind the data source description as shown above, but maintaining a growing list of sources is a non-trivial.

GPS Accuracy: Although the accuracy of outdoor GPS devices is increasingly over time, its accuracy indoors falls short because of certain fundamental problems. Specifically, the signals from the satellites attenuate drastically because of multiple walls in the path to the receiver. Also, increased GPS sensor activity has led to excessive battery consumption in mobile phones. Techniques are being developed to optimize battery usage in such situations.

Manual Data Creation: A large amount of contextual information is currently created manually. Scaling this process to create events from all parts of society is going to be a challenging one. One specific example, is the recorded interval of an event. An event such as a party might exceed the time interval specified on Eventbrite or Facebook by more than a few minutes. In this case, the photos, which according to the user were taken at the party, are no longer going to be associated with the photo. In our current work, we allow for some error margins to associate events to a given photo. At the same time, with the Internet of things movement becoming more mainstream, the challenge is to scale context discovery to billions of sources. This problem of discovering context from billions of sources has been addressed in a different light in the data integration community, and some of its lessons can be applied to this problem.

CPU Efficiency: The personal data (social media, emails, conference events) were downloaded and installed locally on a computer before running the discovery algorithm over photos. Looking up data during tagging time would make the algorithm very expensive (due to multiple round trips calling web APIs or parsing native file formats). The computational complexity arises mainly from merging large event graphs. Although in our case, the graphs never got larger than a few nodes. This is because our datasets were tailored to a specific person. We did not combine data from different users in our experiments.

The query engine in CueNet is responsible for extracting data from different sources. If a very large number of photos are being tagged, our scheme of query generation and merging will prove redundant and inefficient. Scaling the context discovery algorithm to many concurrent photo tagging requests from different people provides a very rich opportunity to develop interesting heuristics using event semantics for the multi-query optimization problem. Also, partitioning the discovery algorithm such that the computations can occur in a distributed manner is a complex problem. Such steps will be required if the application workload is of the scales of Facebook or processing photos in real time at the scales of Instagram.

Face Verification: Even though face recognition has been studied in research for the last two decades, face verification, and its specific application to faces in the wild has been a relatively recent venture. Although the accuracy of these systems is commendable, the problems of verification in photos with occlusion, poor image quality, and diverse lighting conditions exist. These hard problems need to be solved before “perfect” or “near-perfect” verification can be established.

Execution Patterns: When is a good time to execute the algorithm? When a user takes a photo? Or before she uploads it to her favorite photo sharing site? Or should we reevaluate the tags on a photo if a user updates her social network. For the current evaluation, contextual sources are assumed to be immutable. This is not true in the real world. Contextual sources are constantly being appended with new information, and old information is being updated. These updates may be vital in tagging a certain photo. So the question of when to execute the algorithm or when to query the sources is an unanswered one. If a large number of photos are to be tagged, and a busy source like Facebook is being used for context, the CueNet query engine must take into account various freshness metrics and crawling policies of the sources.

Open Datasets: The unavailability of a large public data set over which different techniques can be evaluated against each other is an open problem. As seen in our experiments, personal information is vital to contextual approaches, and this data is largely personal, and therefore cannot be shared openly. Optimal anonymization techniques need to be invented such that the privacy of the experiment participants are maintained, and at the same time the data is meaningful to be applied in contextual approaches to problems. This need to be solved before new context discovery techniques can be evaluated independently and against each other.

6. Related Work

The use of context in the sciences has been continuously increasing. It finds applications in various fields, starting from its use in holistic thinking to better understand biological and ecological phenomenon in [3], to associating the right external data to form coherent stories about economic phenomenon [13], to associating plausible connections between history and geography in [6]. The advances proposed in these works can be loosely characterized as "utilizing external information" or "out-of-the-box thinking". The reason they are included in this section is because of their common trait of relating multiple pieces of external data, to create coherent stories which allows the scientists to gain valuable insights into the problem they are solving.

In Computer Sciences, the Pervasive Computing community and the Human-Computer Interaction community have aggregated a wealth of knowledge on context aware applications. Their interpretation of the word 'context' is mainly inspired from the definition set by Anind Dey [5], i.e. **Context is any information which describes the situation of an entity**. The role of context in mobile computing has been studied in [4]. It shows the growing importance of contextual thinking in reasoning about networking problems in the mobile computing era.

More recently, information retrieval communities are showing interest in context based representation of data and context-based techniques. One of the most important works in information retrieval is the PageRank algorithm [15] developed by Google co-founders Larry Page and Sergey Brin in 1996. In this work, Page and Brin define context as the links between different web pages, and the anchor text of these links, and argued that this context is more descriptive of a page than the contents of the page itself. PageRank was derived by combining this insight with Jon Kleinberg's HITS [11] algorithm.

There are many definitions of the word context in academic works. Notable among them are those of Schilit [18], Dey [5], Viera [20], Zimmermann [22] and the work of Patrick Brezillon, a summary of which can be found in [13]. One of the problems with the term Context is the overloaded use of it. To this effect, Brezillon compiled a list of 150 definitions, and did to this list what lots of scientists do with large collections of text -- text analysis using natural language processing techniques, and derived the essential components which should make up a holistic definition of context. Their conclusion as described in [1] was that *context acts like a set of constraints that influence the behavior of a system (a user or a computer) embedded in a given task*. In spite of this promising "big-data" analysis, many questions are unanswered. For example, is context internal or external? Is it a phenomenon or an organized network? The most accepted definition of context is the one proposed by Dey: *as any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*. The information about relations between objects is implicit or specified by a designer at design time. In our work, we make this information explicit by taking a temporal relation centric view: a notion where context is determined by the relations between entities at a particular point in time. Also, in our work, we consider an entity to be an object (which doesn't display time sensitive properties) or an event (which are time sensitive).

Context is represented using three components: knowledge, external context and proceduralized context by [2]. Use of temporal, spatial and social types for a recommendation service has been explored in [19]. These ideas are represented using a contextual graph (CxG) representation of knowledge and reasoning. Henricksen et al. [9] also use a graphical notation to represent their context. The former approach uses the graph to model the flow of reasoning between different objects within an environment, whereas the latter approach uses graph to only represent the various objects and their inter-relationships. Their modeling framework allows representation of static and dynamic associations, which is very critical in modeling real world relationships. For our work, we rely on primitives like the one mentioned in this framework as well as event relations described in [8]. These

event semantics (*subevent-of* and *partitcipation* relation described in Section 2) provides an additional dimension of reasoning than the works described here. These events semantics allow user defined ontological models to be plugged into our system to allow it to operate in a variety of real world situations. In [16], Reignier et al. present a technique to transform contextual graphs to create concrete situation handling implementations using Petri Nets. In [10], Hong et al. present a detailed survey of various other context-aware systems.

7. Summary

We presented a innovative context based technique to prune complex search spaces for the face tagging problem in personal photos taken in the real world. We model context as a time varying relationship of objects and events to architect a novel context discovery framework CueNet, and designed its discovery algorithms to progressively query various heterogeneous data sources and merge context relevant to a given photo. We empirically analyzed the efficacy of context discovery to remove a large number of irrelevant candidates.

There are many challenges which need to be addressed in techniques presented here. For example, the application of our techniques to very large scale data sets has been proved difficult due to privacy and accessibility issues. It would be interesting to check if the experimental results we have obtained would be similar if ran over millions of users participating in hundreds of types of events. Also, it would be very interesting to see how probabilistic event models can be used to automatically learn and model events in the real world, and then if these models could provide context for new events. It would be very interesting to see how context discovery can be applied to other real world phenomenon such as personal health care, weather prediction, financial services, road traffic management or situation based advertising.

Bibliography

[1] BAZIRE, M., BRÉZILLON, P., Understanding context before using it, *Modeling and using context*, p. 113-192, 2005.

[2] BRÉZILLON, P., *Context-based modeling of operators' practices by contextual graphs*, In Proceedings of the 14th Conference on Human Centered Processes, p. 129-137, 2003.

[3] CAPRA, F., *The web of life: A new scientific understanding of living systems,* Anchor, 1997.

[4] CHEN, G., KOTZ, D., A survey of context-aware mobile computing research, *Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College*, no. 2-1, 2000.

[5] DEY, A., Understanding and using context, *Personal and ubiquitous computing,* p. 4-7, 2001.

[6] DIAMOND, J.M., *Guns, germs, and steel*, 2011.

[7] FAN, X., HU, Y., ZHANG, R., CHEN, W., BRÉZILLON, P., *Modeling temporal effectiveness for context-aware web services recommendation*, IEEE International Conference on Web Services (ICWS), p. 225-232, 2015.

[8] GUPTA, A., JAIN, R., *Managing event information: Modeling, retrieval, and applications, Synthesis Lectures on Data Management 3*, p. 1-141, 2011.

[9] HENRICKSEN, K., INDULSKA, J., RAKOTONIRAINY, A., *Modeling context information in pervasive computing systems, Pervasive Computing*, pp.79-117, 2007.

[10] HONG, J.Y., SUH, E.H., KIM, S.J., *Context-aware systems: A literature review and classification*, Expert Systems with Applications, 36(4), p. 8509-8522, 2009.

[11] KLEINBERG, J.M., *Authoritative sources in a hyperlinked environment*, Journal of the ACM (JACM) 46, no. 5, 1999.

[12] KUMAR, N., BERG, A., BELHUMEUR, P.N. AND NAYAR, S., *Describable visual attributes for face verification and image search*. IEEE Transactions on Pattern Analysis and Machine Intelligence, p. 1962-1977. 2011

[13] LEVITT, S.D., DUBNER, S.J., *Freakonomics: A Rogue Economist Explores the Hidden Side of Everything*, 2007.

[14] MOSTEFAOUI, G.K., PASQUIER-ROCHA, J., BRÉZILLON, P., *Context-aware computing: a guide for the pervasive computing community*, Pervasive Services, p. 39-48. IEEE, 2004

[15] O'HARE, N., SMEATON, A.F., *Context-aware person identification in personal photo collections*. IEEE Transactions on Multimedia, 11(2), p.220-228, 2009.

[16] PAGE, L., BRIN, S., MOTWANI, R., WINOGRAD, T., *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, 1999.

[17] REIGNIER, P., BRDICZKA, O., VAUFREYDAZ, D., CROWLEY, J.L., MAISONNASSE, J., *Context-aware environments: from specification to implementation*, Expert systems, p. 305-320, 2007.

[18] SCHILIT, B., ADAMS, N., WANT, R., *Context-aware computing applications*, In Workshop on Mobile Computing Systems and Applications, p. 85-90. IEEE, 1994

[19] STONE, Z., ZICKLER, T., DARRELL, T., *Autotagging facebook: Social network context improves photo annotation*, Computer Vision and Pattern Recognition Workshops, 2008.

[20] VIEIRA, V., TEDESCO, P., SALGADO, A.C., *Designing context-sensitive systems: An integrated approach*, Expert Systems with Applications, 2011.

[21] YEH, R.B., PAEPCKE, A., GARCIA-MOLINA, H. NAAMAN, M., *Leveraging context to resolve identity in photo albums*, Proceedings of the 5th Conference on Digital Libraries, p. 178-187 2005.

[22] ZIMMERMANN, A., LORENZ, A., OPPERMANN, R., *An operational definition of context,* p. 558-571, 2007.

