UNIVERSITY OF CALIFORNIA,
IRVINE

# Pruning Large Search Spaces using Context Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Arjun Satish

Dissertation Committee:
Professor Ramesh Jain, Chair
Professor Nalini Venkatasubramanian
Professor Deva Ramanan
Professor Bill Tomlinson
Dr. Amarnath Gupta

2013

# DEDICATION

(Optional dedication page)
To ...

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

# CURRICULUM VITAE

## Arjun Satish

### EDUCATION

**Doctor of Philosophy in Computer Science** **2012**
University name *City, State*

**Bachelor of Science in Computational Sciences** **2007**
Another university name *City, State*

### RESEARCH EXPERIENCE

**Graduate Research Assistant** **2007–2012**
University of California, Irvine *Irvine, California*

### TEACHING EXPERIENCE

**Teaching Assistant** **2009–2010**
University name *City, State*

## REFEREED JOURNAL PUBLICATIONS

**Ground-breaking article** **2012**
Journal name

## REFEREED CONFERENCE PUBLICATIONS

**Awesome paper** **Jun 2011**
Conference name

**Another awesome paper** **Aug 2012**
Conference name

## SOFTWARE

**Magical tool** `http://your.url.here/`
*C++ algorithm that solves TSP in polynomial time.*

# ABSTRACT OF THE DISSERTATION

Pruning Large Search Spaces using Context Networks

By

Arjun Satish

Doctor of Philosophy in Computer Science

University of California, Irvine, 2013

Professor Ramesh Jain, Chair

The search spaces of real world AI problems are extremely large. Consider the example of tagging faces in a person's photo album. The search space contains a few billion potential candidates. Any algorithm which attempts to directly tag one of billion people in a given photo will perform poorly. Measures are taken by systems to prune the search space prior to invoking a decision making algorithm. The strategy adopted by such measures is to model the environment accurately to reason which parts of the search space can be pruned without hurting the performance of the algorithm. Deriving models for a real world application like personal photo tagging is challenging due to the diversity of environments in which the photo capture occurs.

# Chapter 1

# Introduction

Artifical Intelligence is defined as the study and design of intelligent systems. An intelligent system is one which perceives its environment and takes actions that maximizes its chances of success. The nature of this environment varies from system to system. For example, the environment for a chess playing system can be enumerated with a set of rules; the environment for an autonomous car is the size, position of cars, their relative speeds on the street and the position of important objects like traffic lights, stop signs. The search space of an AI problem is the total number of candidate objects over which a decision needs to made. In both the above examples, the search space contains an enormous of candidates. But for the second case, the problem of enumerating the search space is non-trivial. It requires a large amount of knowledge about the environment. Also, the interactions between elements in the real world are very hard to predict. They need to be *sensed* in a real time fashion for decisions to be made accordingly. With the growing amount of sensors in the real world, its becoming possible to record events and activities of increasingly finer granularities. Sensory inputs in the real world could range from news articles on the web to stock market tickers to thermometer readings from wildlife parks. Such a large amount of heterogenous information cannot be easily stored, processed and analyzed.

This dissertation addresses the problem of constructing computational representations of real world enviornments from various heterogenous sensors, to reason which parts of the search space can be pruned without hurting the overall performance of the intelligent system. We refer to such a representation as the **Context Network** of the environment. The network describes real-world events occuring in the environment, the entities participating in them, and their semantic inter-relationships.

## 1.1 Approach

Before embarking on a mission to model the entire world, we ask ourselves the question: How much of the real world information is actually relevant to the intelligent system? Constructing the entire world model is extremely challenging and often unnecessary. For example, there might not be much value in representing sports events in New York in a model being used to help cars navigate in Japan.

This dissertation presents a ***progressive discovery*** algorithm to ingest information from various real world data sources to construct context networks containing the most relevant information for pruning the search space for the system. Examples of data sources include social media web services to provide information about events and entities like Facebook, Twitter; services which can be queried to find information about places like Yelp; Sensors on personal mobile phones, for example GPS which inform applications of the location of a person is present at any given point in time.

**What is progressive discovery?** Progressive discovery is an incremental process where knowledge of real world events and entities can be added to a given context network. If we look at this definiton recursively, it says that given a context network and some data sources describing events and entities, a progressive discovery algorithm will recursively ob-

tain information from the sources and relate it to context network. By repeatedly executing this algorithm, we can grow a context network until the data sources can provide no further information or the information in the network prunes the search space well enough for the AI problem to be fully solved.

In the following chapters, context networks, and their discovery from various data sources will be done in conjunction with an application to **tag faces in personal photos**. The face tagging algorithm, whose search space contains a few billion entities is a very hard real world AI problem. But if a real world model of the world existed, the search space which is relevant to this photo contains just the entities who are present within the field of view of the camera at the time the photo was captured.

## 1.2 Overview

This dissertation is organized into the following chapters. Chapter 2 provides an overview of context, and how context has been used to address problems in various communities. Chapter 3 describes the related work in computer science uptil now, and how this work is informed by them. Chapter 4 describes our context discovery framework, how it models various data sources, and how our progressive discovery algorithm constructs models for real world problems. We facilitate this discussion with an example real world application to tag faces of people in personal photos. Chapter 5 analyzes the algorithmic complexity of different parts of the system, and provides experiments to confirm the same. We also present experiments to confirm the efficacy of our approach in the light of the real world application. Chapters 6 and 7 describe two extensions to the CueNet framework to solve problems of missing context and that of source selection. Finally, chapter 8 attempts to describe the future possibilities of context discovery.

## 1.3 Terminology

Before starting the discussion on Context Networks, it is necessary to include this short note on terminology to avoid any ambiguities. We use the word 'Object' to collectively refer to events and entities. An entity includes persons, places in the world, for example 'Starbucks, UC Irvine', 'The Eiffel Tower, Paris, France', or organizations, for example 'Google Inc', 'Royal Society of London'. 'Object' has been used in literature to refer to things which have no spatio-temporal properties. But, in our discussion, an 'object' could imply an event which exhibits spatio-temporal properties.

# Chapter 2

# What is Context?

## 2.1 Event Context

Our justification for the use of context begins with the statement: *For a given user, the correctness of face tags for a photograph containing people she has never met is undefined.* This observation prepares us to understand what context is, and how contextual reasoning assists in tagging photos. The description of any problem domain requires a set of abstract data types, and a model of how these types are related to each other. We **define** contextual types as those which are semantically different from these data types, but can be directly or indirectly related to them via an extended model which encapsulates the original one. Contextual reasoning assists in the following two ways. **First**, contextual data restricts the number of people who might appear in the photographs. We can also argue that all the personal data of a user (her profile on Facebook, LinkedIn, email exchanges, phone call logs) provides a reasonable estimate of all these people who might appear in her photos. **Second**, by reasoning on abstractions in the contextual domain, we can infer conclusions on the original problem. We exploit this property to develop our algorithm in the later

sections. Though CueNet can be applied to a variety of recognition problems, we focus on tagging people in personal photos for concreteness, where, the image and person tag form the abstractions in the problem domain. The types used in the contextual domain, but not limited to, are the following:

- **Events**: includes description of events like conferences, parties, trips or weddings, and their structure (for example, what kind of sessions, talks and keynotes are occurring within a particular conference).
- **Social Relationships**: information about a user's social graph, people whom she corresponds with using email and other messaging services.
- **Geographical Proximity**: various tools like Facebook Places, Google Latitude or Foursquare provide information about where people are at a given time.

The above classes of contextual data can be obtained from a variety of data sources. 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. We classify sources into the following types:

- **Personal Data Sources**: include all sources which provide details about the particular user whose photo is to be tagged. Some common examples of personal data sources include Google Calendar, Email and Facebook profile and social graph.
- **Social Data Sources**: include all sources which provide contextual information about a user's friends and colleagues. For example, LinkedIn, Facebook and DBLP are some of the commonly used websites with different types of social graphs.
- **Public Data Sources**: include all sources which provide information about public organizations (like restaurants, points of interest or football stadiums) or about public events (like fairs, concerts or sports games).

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.

At this point we will mention 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, as now, sources are not hardwired to photo, but instead need to be discovered gradually.

# Chapter 3

# Related Work

The role of context in computing has been studied in **?**. The use of context in image retrieval is emphasized in **??**. Barthelmess et al. extract semantic tags from noisy datasets containing discussions, speeches about a set of photos in question**?**. Naaman et al. have exploited GPS attributes to extract place and person information **??**. Rattenbury **?** devised techniques to find tags which describe events or places by analyzing their spatiotemporal usage patterns. Ames and Frohlich **??** independently describe a survey conducted to study motivations for people to tag their photos. They noticed two broad motivations: Organization of photos and Communication with photos. Time alone is used for organizing photos in **??**. Brave new world applications for photography have been described in **??**, where life logs were collected in the form of photos, emails, document scans and stored in SQL Server database, and photos were retrieved using SQL queries. The photo content was tagged by the user in this case. The Computer Vision community has contributed extensive work in the area of detecting scenes **?**, humans **?** or geo localization **?**. Context information and image features are used in conjunction by **????** identify tags. The semantic web community is using linked data technologies to annotate and query photographs **??**. Collaborative games also have been evaluated as a possible way to tag photos**?**. Systems like Picasa, iPhoto and **?** organize

photos based on time, GPS coordinates and sometimes faces in the photo. These attributes of the photo do not capture event semantics **?**. Events are a natural way of categorizing photo media. Events also allow large number of photos captured during a single event be organized hierarchically using subevents.

## 3.1 Ontologies

Lack of facilities to express spatio-temporal constraints.

## 3.2 Data Integration

## 3.3 Computer Vision: Face Recognition/Verification

# Chapter 4

# Context Discovery Framework

Automatic media annotation algorithms essentially assign one or more labels from a search space to a given input image. Figure **??** shows the various approaches of constructing such a search space for such an algorithm. The traditional approach is shown in **??**(a). These spaces were limited to a set of labels chosen by an expert, with no way of pruning the search space in case it got very large. The focus was instead on extracting the best features from images, to obtain high overall classification accuracy**?**.

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 like conferences, weather events and sports games. With such context sources, the search space construction is being delegated to one or a few sources **?????** (figure **??**(b)). These approaches rely on a single *type* of context. For example, time and location information or 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

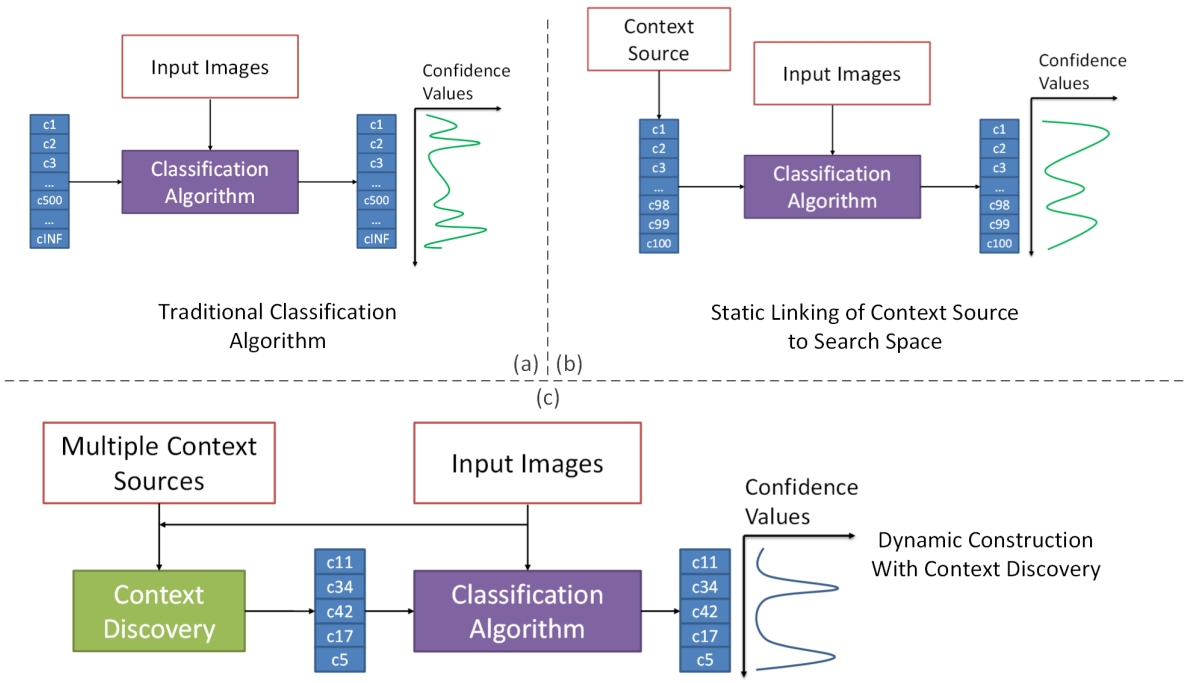


Figure 4.1: The different approaches in search space construction for a multimedia annotation problem. A traditional classifier setup is shown in (a) where the search space candidates are manually specified. Context is used to generate large static search spaces in (b). The desired framework is shown in (c), which aims to produce small search spaces with many correct annotations.

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 **??**(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 a classification 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

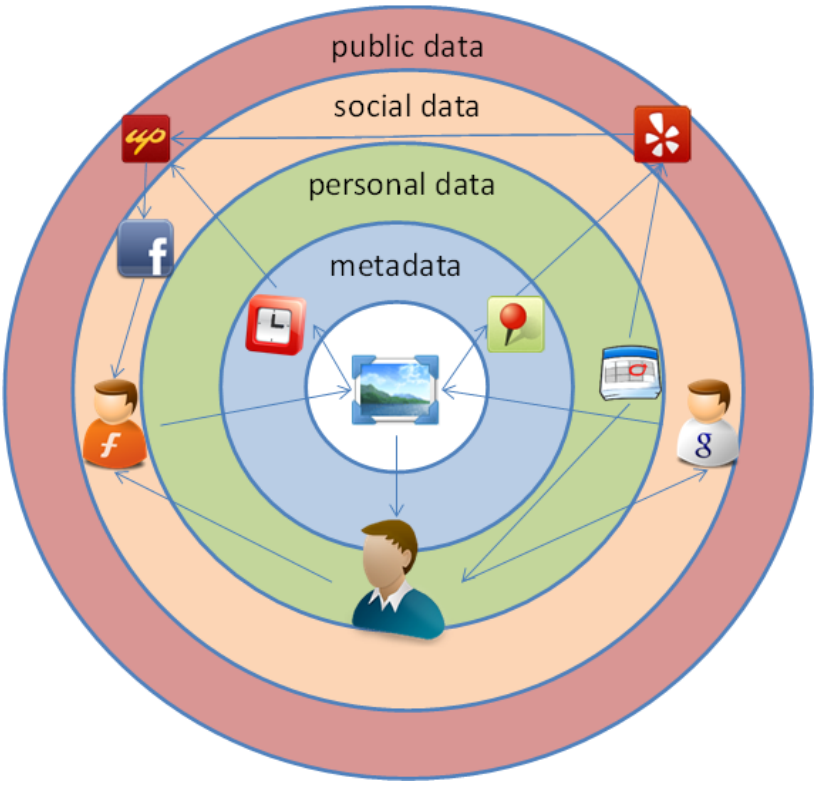


Figure 4.2: Navigation of a discovery algorithm between various data sources.

problem. To ensure 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 by different users. These photos could contain friends, colleagues, speakers giving very interesting talks, or newly found acquaintances (who are not yet connected to the user through any social network). This makes the conference photos particularly interesting because no single source can provide all the necessary information. It emphasizes the need to utilize multiple sources in a meaningful way.

Here is an **example** to illustrate CueNet's discovery process. Let's suppose that Joe takes a photo with a camera that records time and GPS in the photo's EXIF header. Additionally, Joe has two friends. One with whom he interacts on Google+, and the other using Facebook. The framework checks if either of them have 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 identified by CueNet, they are passed to the face tagging algorithm (like **?**), which can perform very well as their search space is limited to two candidates.

Figure **??** shows the different components of the CueNet framework. The Ontological **Event Models** specify various event and entity 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 database of people, their profile information and photos containing these people. 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 find the confidence threshold. In this section, we describe each module, and how the context discovery algorithm utilizes them to accomplish its task.

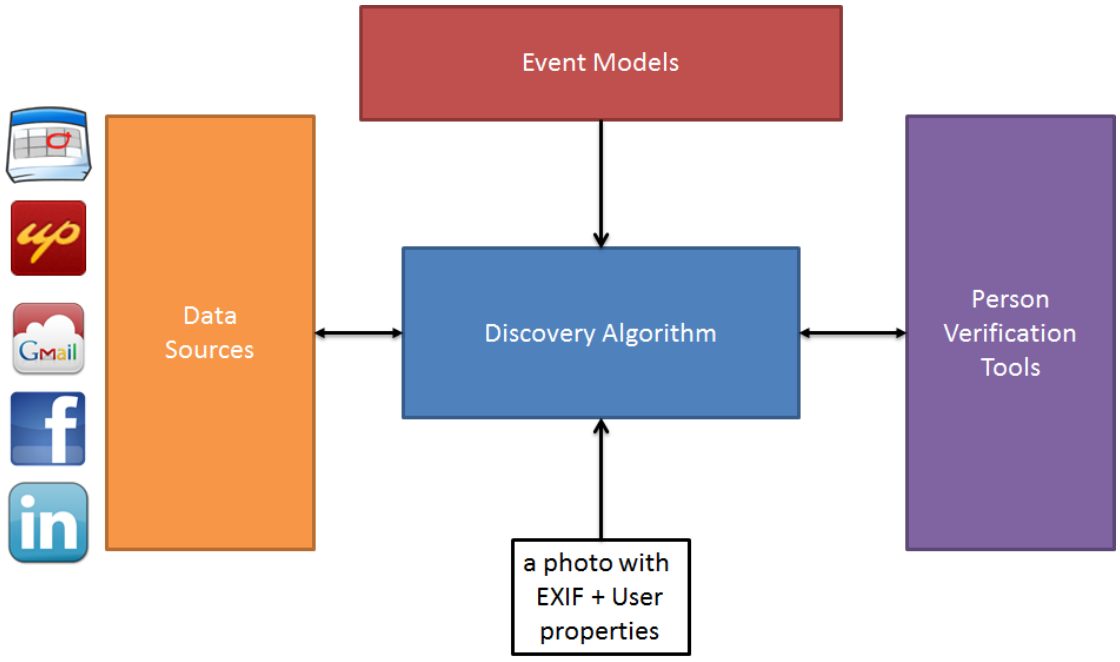


Figure 4.3: The Conceptual Architecture of CueNet.

## 4.1 Event Model

Our ontologies extend the E* model**?** 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 entity 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 $\mathcal{L}_P$ during the time $\mathcal{T}_P$ and an event $E$ occurs within the spatiotemporal region $< \mathcal{L}_E, \mathcal{T}_E >$, we say $P$

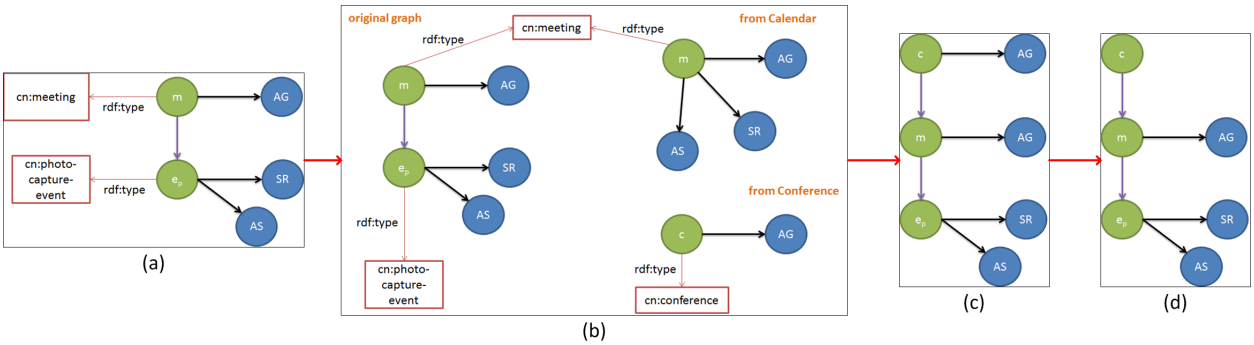


Figure 4.4: The various stages in an iteration of algorithm **??**.

is a participant of $E$ if the event's spatiotemporal span contained that of the participant.

$$\texttt{participant}(\text{E, P}) \iff (\mathcal{L}_P \sqsubset_L \mathcal{L}_E) \wedge (\mathcal{T}_P \sqsubset_T \mathcal{T}_E) \tag{4.1}$$

The symbols $\sqsubset_L$ and $\sqsubset_T$ indicate spatial and temporal containment respectively. Please refer to **?** for more details. In later sections, we refer to the location and time of the event, $\mathcal{L}_E$ and $\mathcal{T}_E$ as $E$.**occurs-at** and $E$.**occurs-during** respectively.

## 4.2 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 to allow developers of the system to specify these declarations. The example below describes a source containing conference information.

```
(:source conferences
   (:attrs url name time location title)
```

```
(:rel conf type-of conference)
(:rel time type-of time-interval)
(:rel loc type-of location)
(:rel attendee type-of person)
(:rel attendee participant-in conf)
(:rel conf occurs-at loc)
(:rel conf occurs-during time)
(:axioms
   (:map time time)
   (:map loc location)
   (:map conf.title ltitle)
   (:map conf.url url)
   (:map attendee.name name)))
```

A source declaration comprises of a single nested s-expression. We will refer to the first symbol in each expression as a keyword, and the following symbols as operands. This above declaration uses five keywords (`source`, `attrs`, `rel`, `axioms`, `map`). The `source` keyword is the root operator, and declares a unique name of the data source. The source mapper can be queried for finding accessors using this name. The `attrs` keyword is used to list the attributes of this source. Currently we assume a tuple based representation, and each operand in the attrs expression maps to an element in the tuple. The `rel` keyword allows construction of a relationship graph where the nodes are instances of ontology concepts. And edges are the relationships described by this particular source. In the above example, we construct individuals *conf*, *time*, *loc* and *attendee* who are instances of the *conference*, **time-interval**, **location** and **person** class respectively. We further say that attendee is a ***participant of*** the conference, which ***occurs-at*** location loc and ***occurs-during*** the interval time. Finally, the `mapping axioms` are used to map nodes in the relationship graph

to attributes of the data source. For example, the first axiom (specified using the map keyword) maps the time node to the time attribute. The third map expression creates a literal called title, and associates it to the conference node, whose value comes from the ltitle attribute of the conference data source.

Formally, we represent the given ontology as $O$. The various classes and properties in $O$ are represented by $C^O$ and $P^O$ respectively. Since our upper ontology consists of DOLCE and E*, we assume the inclusion of the classes `Endurant`, `Perdurant`, `Event` and `Person` in $C^O$. Each source $S$ consists of three parts, a relation graph $G^S(V^S, E^S)$ where the nodes $V^S \in C^O$, specify the various "things" described by the source. The edges $E^S \in P^O$ specify the relations among the nodes. Any graph retrieved from such a source is an instance of the relation graph, $G^S$. Further, the tuple $A_T^S$ consists of the attributes of the data source. Finally, the mapping $M^S : \{G^S \rightarrow A_T^S\}$ specifies how to map different nodes in the relation graph to the different attributes of the native data source.

## 4.3 Conditions for Discovery

CueNet is entirely based on reasoning in the event and entity (i.e., person) domain, and the relationships between them. These relationships include participation (event-entity relation), social relations (entity-entity 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:

**Entity Existence Axiom**: Entities can be present in one place at a time only. The entity cannot exist outside a spatiotemporal boundary containing it.

**Participation Semantics Axiom**: If an entity 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 $n \in D$. 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:

$$\begin{aligned} v_i^I = v_r^R \iff & (v_i^I.\textbf{type-of} = v_r^R.\textbf{type-of}) \wedge \\ & (v_i^I.\textbf{occurs-at} =_L v_r^R.\textbf{occurs-at}) \wedge \\ & (v_i^I.\textbf{occurs-during} =_T v_r^R.\textbf{occurs-during}) \end{aligned} \tag{4.2}$$

**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 `Endurant`.

$$\begin{aligned} &v_i^I \sqsubset_L v_r^R, \\ &v_i^I \sqsubset_T v_r^R \end{aligned} \tag{4.3}$$

$$v_i^I.\textbf{Endurants} \cap v_r^R.\textbf{Endurants} \neq \{\phi\} \tag{4.4}$$

Here $v_i^I$.**Endurants** is defined as a set $\{w | w \in V_i^I \wedge w.\text{type-of=Endurant}\}$. If equation (**??**) does not hold, we 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 **??**(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 (**??**) is used. Because of the entity existence and participation semantics axioms, all events which contain a common participant are connected to their respective super event using the subevent

relation (equations (**??**) and (**??**) 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 (**??**) 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.

### 4.3.1 Context Discovery Algorithm

Algorithm **??** below outlines the tail recursive discovery algorithm. 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 entity 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 **??** graphically shows the main stages in a single iteration of the algorithm.

The event graph is traversed to produce a queue of entity and event nodes, which we shall refer to as DQ (discovery queue). The algorithm consists of two primary functions: **discover** and **merge**. The discover function is tail recursive, invoking itself until a termination condition is reached (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 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 entity 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 $T$). Push down will try to verify if any of the newly discovered entities are present in the photo and if they are (if the tagging confidence is higher than the given threshold), the entities are removed from the super event, and linked to the photo capture event as its participant. On the other hand, if this number 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. Ranking is done by simply sorting the candidate list by descending order of score. The face verification runs only on the top ranked $T$ 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 **??** 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 merg-

ing these graphs. (d) The `prune-up` function removes the duplicate reference to AG.

A live visualization of these steps for different photos can be found at http://cuenet.site44.com.

## 4.4 Implementation

**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
```

# Chapter 5

# Analysis and Experiments

## 5.1 Analysis

## 5.2 Experiments

In this section, we analyze how CueNet drives a real world face tagging application. The application contains a set of photos, and a database of people, and its goal is to associate the right persons for each photo, with high accuracy. The goal of CueNet, and the focus of our analysis, is to provide small search spaces so that the application can exhibit high accuracy in all datasets.

In the following evaluation, we investigate three hypotheses. **First**, what sources provide the most interesting context? **Second**, how small are the candidates lists constructed by the discovery algorithm, which are provided to the classification algorithm as a "pruned" version of the search space? And **third**, what percentage of true positives does this pruned search space contain?

### 5.2.1 Setup

We use 368 photos taken at 7 different conferences in our face tagging experiment. The person database consists of 660 people. Each photo contains one or more persons from this database. The owner of the photos was asked to provide access to their professional Google Calendar to access personal events. Information from social networks was gathered. Specifically, events, social graph, photos of user and their friends from Facebook. In order to obtain information of the conference event, we used the Stanford NER**?** to extract names of people from the conference web pages. Descriptions of the keynote, session and banquet events were manually entered into the database. Our sources also included personal emails, access to public events website upcoming.com (Yahoo! Upcoming) and used Yahoo! PlaceFinder for geocoding addresses.

The ground truth was annotated by the user with our annotation interface. For each photo, this essentially consisted of the ID of the persons in it. We will denote each dataset as 'Di' (where $1 \leq i \leq 8$ for each dataset). Table **??** describes each dataset in terms of number of photos, unique annotations in ground truth and the year they were captured. The total number of unique people who could have appeared in any photo in our experiments is 660. This set forms the exhaustive search space, $L$ from section **??**.

| Dataset | Unique People | No. of Photos | Year |
|---|---|---|---|
| D1 | 43 | 78 | 2012 |
| D2 | 24 | 108 | 2012 |
| D3 | 6 | 16 | 2010 |
| D4 | 7 | 10 | 2010 |
| D5 | 36 | 80 | 2009 |
| D6 | 18 | 65 | 2013 |
| D7 | 7 | 11 | 2013 |

Table 5.1: Profile of datasets used in the experiments.

We divide the sources into different categories to facilitate a more general discussion. The

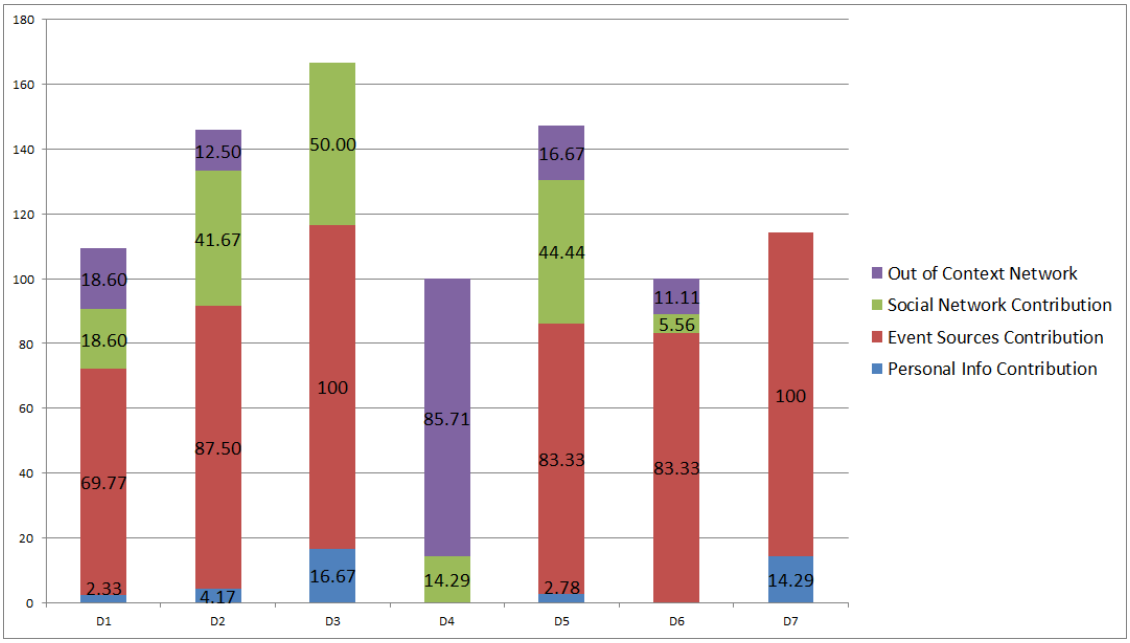


Figure 5.1: The distribution of annotations in the ground truth across various sources.

categories are "Personal Information" (same as Owner Information in section **??**), "Event sources", and "Social Networks". Event sources include Facebook events, Yahoo Upcoming web service, our conference events database among other sources. Social networks include Facebook's social graph. Personal information contained information about the user, and a link to their personal calendars. An annotation is considered "Out of Context Network" if it is not in any of these sources.

Figure **??** shows the distribution of the ground truth annotations 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.

### 5.2.2 Context Discovery

Now, lets look at reduction obtained in state space with the discovery algorithm. The total number of people in our experiment universe is 660. By statically linking the sources, we would expect the search space to contain 660 candidates for tagging any of the datasets. However, the context discovery algorithm reduced the size of the search space as shown in table **??**. The search space varies from 7 people in D7 (1%) to 338 people in D2 (51%). We denote the term hit rate as the percentage of true positives in the search space. Even if our search space is small, it might contain no annotations from the ground truth, leading to poor classifier performance. The hit rates are also summarized in table **??**. For D4, the algorithm found no event sources (as seen in figure **??**), and therefore constructed a search space which was too small, thereby containing none of the ground truth. With the exception for D4, the hit rate is always above 83%. We observe an overall reduction in the search space size, with a high hit rate for majority of the datasets.

| Dataset | Reduced Search Space Size | Hit Rate |
|---|---|---|
| D1 | 42 | 83.72% |
| D2 | 338 | 87.5% |
| D3 | 231 | 100% |
| D4 | 1 | 0% |
| D5 | 254 | 83.33% |
| D6 | 20 | 88.89% |
| D7 | 7 | 100% |

Table 5.2: Sizes of Search Space for each dataset.

We now investigate the role of different context sources in the discovery algorithm. If an entity in the search space was merged into the event graph by an event source, they are

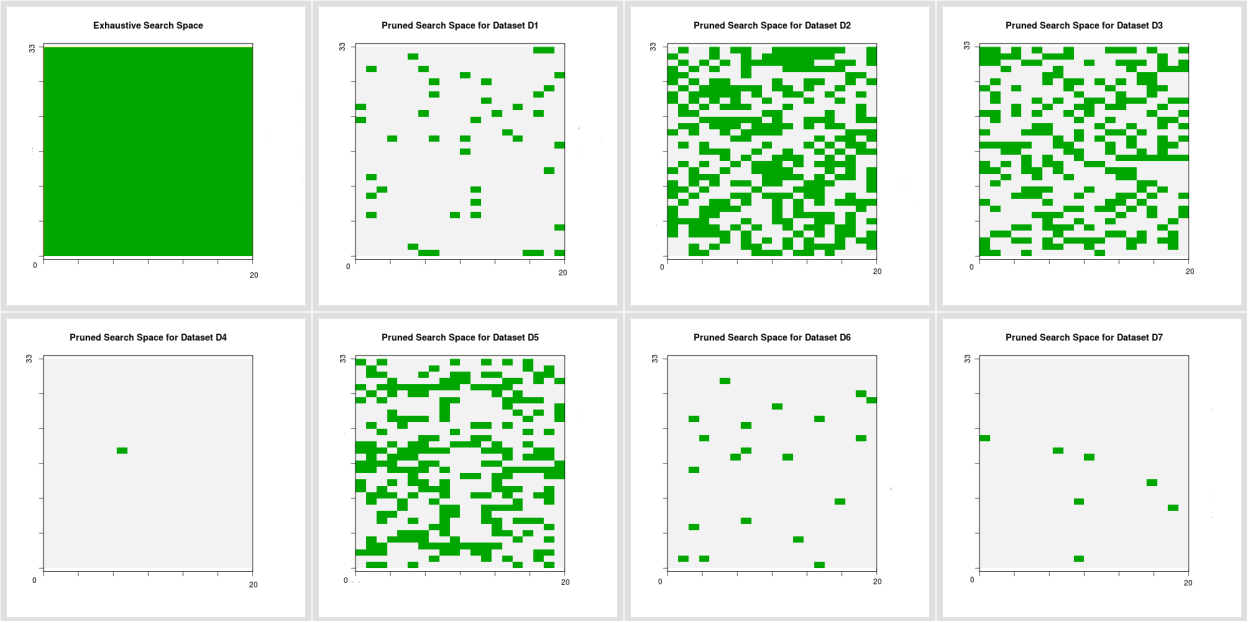


Figure 5.2: Grid plots showing the exhaustive search space and pruning in search space for different datasets.

said to be "contributed" from it. We profiled our algorithm to log all contributions which were true positives for the classification algorithm. Figure **??** shows the contribution from various sources for all datasets. For example, D1 obtained 69.77% of true positives in its search space from event sources, 2.33% from personal information and 11.63% from social networks. 16.28% of true positives for D1 were obtained from no source, and were therefore marked as "Out of Context Network".

This graph brings to light our argument that most of the true positives, for all datasets, were obtained as a result of navigating the event sources. It will also be noted that the role of social networks is minimal. It was found useful for only one dataset. Relying on social networking sources would have led to a large number of false positives in the classifier performance. Even though the role of personal information is negligible, it is critical in linking in photos to the owner, and from there to different events. Without the availability of personal information, the algorithm would not have reached the context rich event sources.

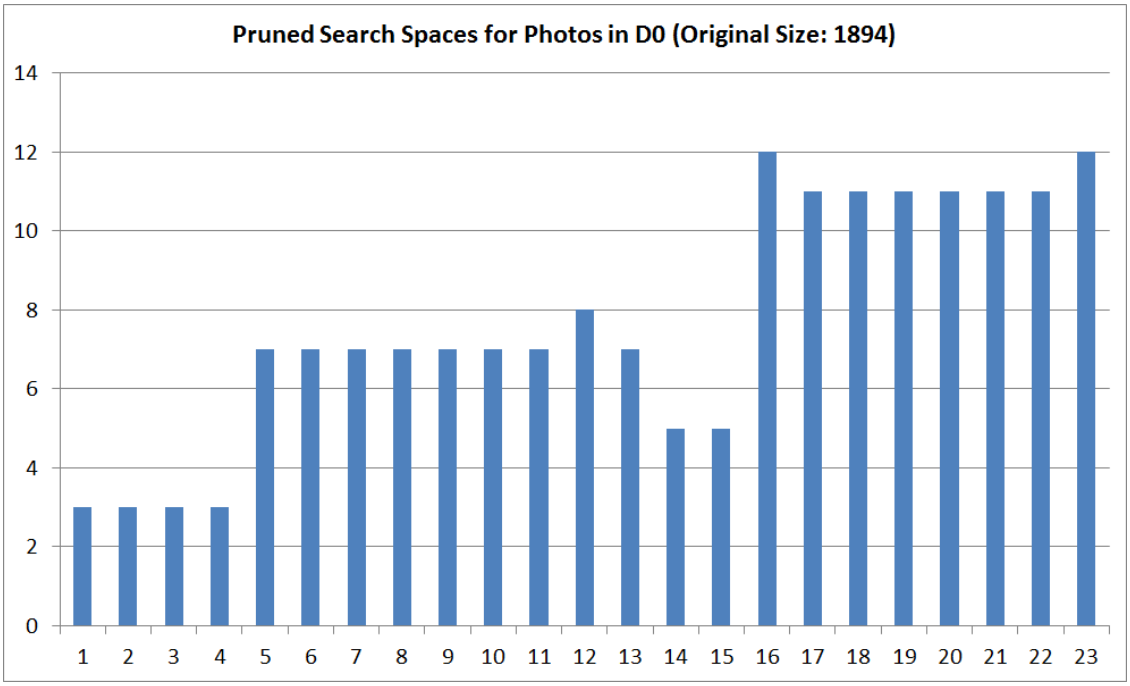


Figure 5.3: Pruned search space for photos in D0.

### 5.2.3 Individual List Sizes in a Dataset

Here we look at how CueNet reduces the number of possible candidates for all photos in a dataset. For this setup, the complete candidate set $L$, contained 1894 labels (total number of people present at the conference, user's emails and social graph). The figure **??** shows various statistics for each photo, 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) was 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).

### 5.2.4 Search Spaces

Finally, we compare the various search spaces constructed by discovery algorithm. We represent all people in our experiment universe in a color grid (with 33x20 cells for 660 people). Each cell represents the presence or absence of a person in the search space. If a person was present in the candidate list provided to the tagging algorithm, we color the corresponding cell green, otherwise it is colored white. Figure **??** shows the color grids describing search spaces for all datasets, and an exhaustive search space. The positioning of people along the grid is arbitrary, but consistent across grids. Our aim in this visualization is to see the diversity in search spaces created by the algorithm. The purpose of the exhaustive search space is to provide easy comparision to appreciate the reduction in search space.

It can be seen that CueNet prunes the search space very differently for different datasets. As we move from dataset to dataset, the data sources present different items of information, and therefore CueNet constructs very search spaces. Dataset D2, D4 and D5 are very large conferences hosting hundreds of people in the same field. This explains why a large portion of the grid is covered. Also, this was the same conference held in three different years, and therefore, had a lot of common attendees resulting in overlap.

### 5.2.5 Conclusion

These experiments validate our three hypotheses. **First**, Event sources contain a large portion of true positives. From 70% in D1 to 100% in D7. There are events for which there is no documentation, and event sources are not able to contribute anything here, as in the case of D4. **Second**, the discovery algorithm is able to prune the search space using event, personal and social information. The reduction is atleast 50% for D2 (338 candidates out of 660) but can be very large in some cases (7 candidates for D7). **Third**, The reduced search space retains a high number of true positives. The hit rate is between 83% to 100% (with

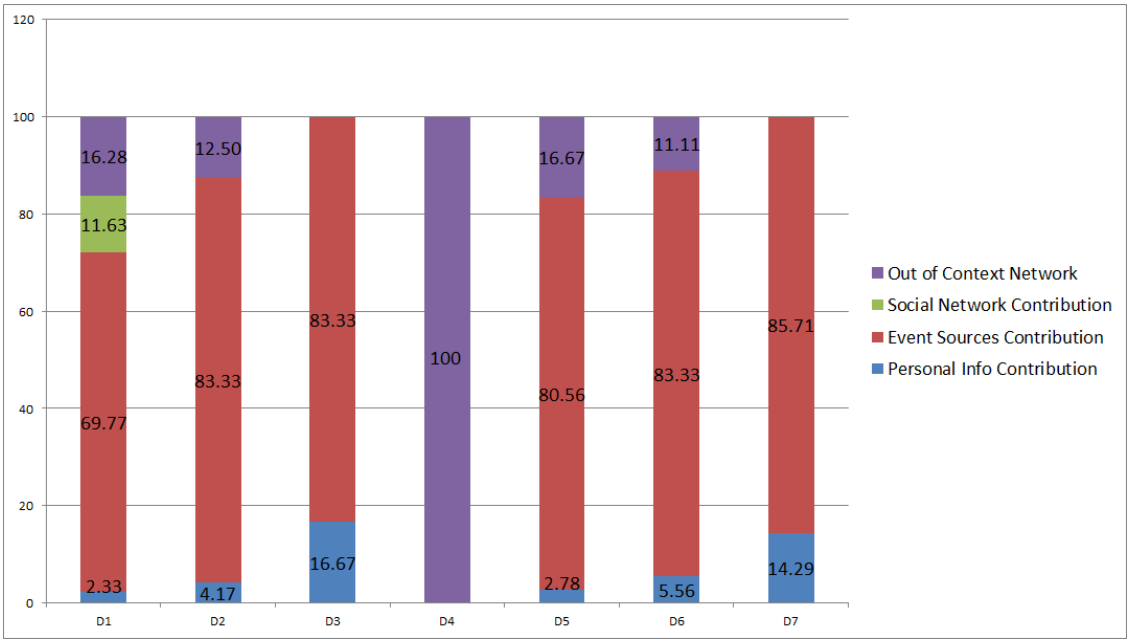


Figure 5.4: Graph showing the contribution of each source type in context discovery.

the exception of D4, where the search space provided no true positives). We also saw how unique the search spaces are, to each dataset, thereby demonstrating the dynamic nature of the algorithm.

## 5.3 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 range from entity matching and record linkage problems **?** to correcting missing data in information networks **?**. These problems get trickier because of the different variations in representing tiny details such as representation names of people, addresses of places, and time. In fact, there is a whole school of anthroponomastics **?** dedicated to studying variations in human names. Ideas in this field indicate that these differences arise due to cultural, historical and environmental issues**?**. Such issues cannot be trivially addressed.

Face tagging 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 quality of photos are poor, resulting in weaker features, which would have otherwise allowed better matching.

An exhaustive scientific characterization of noise in social media is beyond the scope of this paper, and is being investigated step by step in social media research communities.

**CPU Efficiency**: 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 inefficient. Processing many photos 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 occlusion, image quality, face alignment and differing 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? 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 how and when to query the sources is an open question. 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 be to 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 so that new context discovery techniques can be evaluated independently and against each other, over a common platform.

# Chapter 6

# Conclusion and Future Work

# Appendix A

# Appendix

Supplementary material goes here.

## A.1 Source Mapping

```
(:source google-calendar
   (:attrs event email time location title description attendee)
   (:rel ev type-of event)
   (:rel owner type-of person)
   (:rel ti type-of time-interval)
   (:rel ev occurs-during ti)
   (:rel participant type-of person)
   (:rel owner participant-of ev)
   (:rel participant participant-of ev)
   (:io disk (:db mongodb))
   (:type personal)
```

```
   (:axioms
      (:map ev event)
      (:map ti time)
      (:map owner.email email B)
      (:map ev.occurs-during time)
      (:map ev.occurs-at location)
      (:map participant attendee)
      (:map ev.title title U)
      (:map ev.description description U)))

(:source fb-user
   (:attrs id name birthday location work email)
   (:rel person type-of person)
   (:rel named-place type-of named-place)
   (:rel address type-of address)
   (:rel person works-at named-place)
   (:rel person lives-at address)
   (:io disk (:db mongodb))
   (:type personal)
   (:axioms
      (:map person.name name)
      (:map person.dob birthday)
      (:map address.street-address location.name)
      (:map named-place.name work.name)))


(:source email
```

```
   (:attrs from to cc)
   (:rel pf type-of person)
   (:rel pt type-of person)
   (:rel pc type-of person)
   (:io disk (:db mongodb))
   (:type personal)
   (:axioms
      (:map pf.email from)
      (:map pt.email to)
      (:map pc.email cc)))


(:source fb-relation
   (:attrs name1 name2)
   (:rel p1 type-of person)
   (:rel p2 type-of person)
   (:rel p1 knows p2)
   (:io disk (:db mongodb))
   (:type personal)
   (:axioms
      (:map p1.name name1 F)
      (:map p2.name name2 U)))

(:source academix
   (:attrs name1 name2)
   (:rel p1 type-of person)
   (:rel p2 type-of person)
```

```
   (:rel p1 knows p2)
   (:io disk (:db mongodb))
   (:type public)
   (:axioms
      (:map p1.name name1 F)
      (:map p2.name name2 U)))

(:source conferences
   (:attrs time location ltitle stitle url)
   (:rel conf type-of event)
   (:rel time type-of time-interval)
   (:rel loc type-of location)
   (:rel conf occurs-at location)
   (:rel conf occurs-during time)
   (:axioms
      (:map time time)
      (:map loc location)
      (:map conf.title ltitle)
      (:map conf.name stitle)
      (:map conf.url url)))


(:source confattendees
   (:attrs url name time location ltitle stitle)
   (:rel conf type-of conference)
   (:rel time type-of time-interval)
   (:rel loc type-of location)
```

```
   (:rel attendee type-of person)
   (:rel attendee participant-in conf)
   (:rel conf occurs-at location)
   (:rel conf occurs-during time)
   (:axioms
      (:map time time)
      (:map loc location)
      (:map conf.title ltitle)
      (:map conf.name stitle)
      (:map conf.url url)
      (:map attendee.name name)))


(:source keynotes
   (:attrs url time location title name)
   (:rel conf type-of conference)
   (:rel k type-of keynote)
   (:rel k subevent-of conf)
   (:rel attendee participant-in k)
   (:axioms
      (:map conf.url url)
      (:map attendee.name name)
      (:map k.location location)
      (:map k.time time)
      (:map k.title title)))

(:source sessions
```

```
   (:attrs url time location title name)
   (:rel conf type-of conference)
   (:rel k type-of session)
   (:rel k subevent-of conf)
   (:rel attendee participant-in k)
   (:axioms
      (:map conf.url url)
      (:map attendee.name name)
      (:map k.location location)
      (:map k.time time)
      (:map k.title title)))


(:source talks
   (:attrs url time location title name)
   (:rel conf type-of conference)
   (:rel k type-of talk)
   (:rel k subevent-of conf)
   (:rel attendee participant-in k)
   (:axioms
      (:map conf.url url)
      (:map attendee.name name)
      (:map k.location location)
      (:map k.time time)
      (:map k.title title)))

(:source conflunches
```

```
   (:attrs url time location title name)
   (:rel conf type-of conference)
   (:rel k type-of lunch)
   (:rel k subevent-of conf)
   (:rel attendee participant-in k)
   (:axioms
      (:map conf.url url)
      (:map attendee.name name)
      (:map k.location location)
      (:map k.time time)
      (:map k.title title)))


(:source tweets
   (:attrs url name)
   (:rel conf type-of conference)
   (:rel attendee type-of person)
   (:rel attendee participant-in conf)
   (:axioms
      (:map conf.url url)
      (:map attendee.name name)))

(:source fb-events
   (:attrs event name time)
   (:rel ev type-of event)
   (:rel p1 type-of person)
   (:rel ti type-of time-interval)
```

```
(:rel ev occurs-during ti)
(:rel p1 participant-of ev)
(:axioms
   (:map p1.name name)
   (:map ev event)
   (:map ev.occurs-during time)
   (:map ti time)))
```