Monitoring Entities in an Uncertain World: Entity Resolution and Referential Integrity

Steven N. Minton,[1] Sofus A. Macskassy,[2] Peter LaMonica,[3] Kane See,[2]
Craig A. Knoblock,[4] Greg Barish,[2] Matthew Michelson,[2] and Raymond Liuzzi[5]

[1]InferLink Corp. El Segundo, CA
[2]Fetch Technologies El Segundo, CA
[3]Air Force Research Laboratory Rome, NY
[4]USC Information Sci. Inst. Marina Del Rey, CA
[5]Raymond Technologies Whitesboro, NY

Abstract

This paper describes a system to help intelligence analysts track and analyze information being published in multiple sources, particularly open sources on the Web. The system integrates technology for Web harvesting, natural language extraction, and network analytics, and allows analysts to view and explore the results via a Web application. One of the difficult problems we address is the *entity resolution problem*, which occurs when there are multiple, differing ways to refer to the same entity. The problem is particularly complex when noisy data is being aggregated over time, there is no clean master list of entities, and the entities under investigation are intentionally being deceptive. Our system must not only perform entity resolution with noisy data, but must also gracefully recover when entity resolution mistakes are subsequently corrected. We present a case study in arms trafficking that illustrates the issues, and describe how they are addressed.

Introduction

In this paper we describe a system that monitors and analyzes information published on the Web about people, organizations and other entities of interest. The system is the result of collaborative effort by corporate, government and university researchers. Our goal is to reduce the manual effort required by intelligence analysts to track data on a large, diverse set of Web sources. As an example of an application of interest to Air Force Intelligence Analysts, we will describe a case study tracking information about entities related to arms trafficking.

A difficult technical issue that arises when aggregating data from multiple sources is that there may be multiple ways to refer to the same entity. For example, "Air Cess" headquartered in "Sharjah" is the same company as "Aircess International" registered in the "UAE". This problem, which we refer to as Entity Resolution, has been addressed by many researchers (see e.g., Knoblock et al., 2007; Koudas et al., 2006) and a variety of commercial solutions have been developed. However, most existing approaches focus on matching entity references to a clean master list, or reference set, of entities. In the application addressed here, the entity resolution problem is particularly difficult for two reasons. First, there is no pre-existing reference set. The application continuously aggregates information about entities over time from sources that may mention new entities at any time, and those mentions can be noisy/dirty. Second, this application tracks individuals that may purposely attempt to be furtive or deceptive. Individuals may be associated with multiple aliases and multiple suspect organizations, and they may purposely engage in activities to thwart tracking, such as relabeling the tail numbers of airplanes.

This paper focuses on how the entity resolution problem is addressed within the larger architecture of a knowledge-based application. We point out that entity resolution cannot simply be treated as an "added feature", but must be carefully designed into the architecture. The system may discover over time that two entities that were initially believed to be distinct are the same entity, or conversely, that the records describing what was initially inferred to be one entity actually correspond to two or more entities. The system must be able to make these corrections over time, revising its representation of the entities appropriately. If this issue is not carefully addressed, it can lead to problems such as referential inconsistency in the system's knowledgebase, which can in turn make it difficult to run network analytics and other analysis algorithms. We discuss how our initial design for the system was eventually improved based on our experiences, and suggest advice for others implementing solutions where entity-resolution is a consideration.

An Application Domain

As the fight against asymmetric threats continues to grow, intelligence analysts have stepped up their efforts to track information on the Web. Our work with ENTEL (Entity Intelligence Portal) is intended to help intelligence analysts track and analyze information being published in multiple sources, particularly *open sources* on the internet ("Open Source Intelligence", or OSINT). One particular domain of interest to the Department of Defense is the illicit transportation of goods that are often associated with terrorist activities (Hoffman, 2006).

To illustrate the issues involved in this type of tracking and analysis, we will describe the case of Russian

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

businessman Viktor Bout. Bout was the owner of several air cargo companies that have allegedly facilitated illicit trafficking of contraband. In addition, Bout is currently being tried in the United States on several conspiracy charges, including providing support to a designated foreign terrorist organization. This case is illustrative of the issues that are involved in tracking suspected entities. Because of the publicity surrounding the case, many details of the case are public and therefore can be discussed here.

In tracking a suspected arms trafficker such as Viktor Bout, intelligence analysts would collect as much information as possible about the air cargo companies he is associated with, the planes used by those companies, the plane's capabilities, known routes that they have flown, reported observations of those planes, and other people, organizations and companies that Bout deals with or communicates with. In addition, analysts would attempt to infer any aliases or front companies he uses, and document any suspicious and/or deceptive behavior.

Deceptive behaviors come in many forms. One common behavior which Bout allegedly engaged in is transferring aircraft between multiple companies. Doing so makes tracking more difficult, since an aircraft can be re-registered with a new tail number when it is transferred.

There are many information sources on the Web that can aid an analyst. Sites like airliners.net and myaviation.net have millions of pictures of aircraft at airports around the world, submitted by professionals and amateurs. The ASN Aviation Safety Database and AeroTransport Data Bank have large databases with public records about aircraft and air transport companies. In addition, news, blogs and internet forums can contain relevant data.

The site ruudleeuw.com includes a substantial Viktor Bout dossier. The site, authored by a private individual, includes detailed documentation describing Bout's activities, and is illustrative for our purposes. For instance, one page (www.ruudleeuw.com/vbout12.htm) lists over 35 companies with Bout connections, and details the history of a fleet of 5 planes that belonged to one these companies, Air Cess, tracing a series of tail number re-registrations and transfers to other companies.

This history illustrates some of the problems inherent in monitoring entities, even relatively simplistic entities such as airplanes. For instance, the site includes the following statements about an Il-18 (Ilyushin-18) aircraft, with tail number 3D-SBZ:

"In Jan.2004 a photo appeared on Airliners.net of a shot up Il-18 at Kalemie, Katanga…while the registration is not shown, it seems to have the Air Cess c/s and "Air" can be read on the fuselage. ...Another photo [of the plane] appeared on Airliners.net, again at Kalemie but now seen robbed of all its part (07Dec04); unfortunately the tail number has been painted over.

…[Another Ilyushin] was purchased from Santa Cruz in 1999, possibly as replacement for Il-18D ([construction #] 188010903) 3D-SBZ (ex/SP-FNZ) which had disappeared from sight….

Michel Bonnardeaux went to Kalemie to check for the wrecked Il-18 again and have just returned. Identity of 3D-SBZ in faded Air Cess colours is confirmed."

Note that the tail number (the registration number) of the plane is not a unique identifier, due to the fact that the planes can be re-registered (3D-SPZ previously was registered as SP-FNZ), and it is not always observable. The authors also refer to the construction number of aircraft – a unique serial number assigned by the manufacturer – but though this doesn't change, it is even less frequently observable. Over time, earlier observations such as the photo referred to above (shown in Figure 1) can be linked to a distinct serial number, as additional evidence is collected.

Figure 1: Photo of IL-18 aircraft linked to Viktor Bout

ENTEL: System Architecture

We have been building an end-to-end system to address the operational and analytical requirements for monitoring and tracking entities of interest. This system, called ENTEL, is possible due to many years of AI research in areas such as information extraction, information integration and aggregation, entity resolution and knowledge management.

The ENTEL system architecture, depicted in Figure 2, can be viewed as an information pipeline. The high-level information flow starts with (1) the Fetch Agent Platform, a commercial AI system for harvesting Web data. The harvested data is aggregated and stored on an ongoing basis (2). The data can consist of structured records and/or or unstructured text. The unstructured text is then further processed to extract structured facts and relations about entities (3). Up to this point in the pipeline, any entities

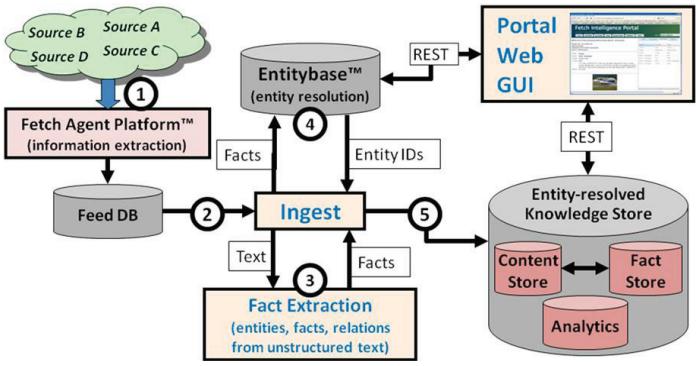

Figure 2: ENTEL System Architecture

that we have information about are simply represented by strings. Our EntityBase component resolves these references to persistent unique entity identifiers (EIDs) which we use for analysis and exploration (4). Finally, we publish the entity-resolved data to the entity knowledge store (5). The knowledge store is used for various analytics and maintains more sophisticated data structures. For example, the system maintains a network of related entities and keeps track of watchlists of entities of interest.

Once the data is in the entity knowledge store, then we can use a Web interface to query the data through a REST-based API layer. As Figure 2 depicts, the Web Interface interacts both with the entity knowledge store as well as the EntityBase component. These communication paths allow a user both to search for entities (entered as strings) and also explore what is known about entities in the knowledge store. We will now briefly describe each of these main components.

Data Harvesting (1)

The Fetch Agent Platform harvests semi-structured information from online Web sources on a scheduled basis. The Fetch Agent Platform is a commercial system that enables users to rapidly build web agents to retrieve data from websites. This process employs machine learning methods so that extractors can be induced by example (Muslea et al., 2001; Ticrea & Minton, 2003). At runtime, the system navigates to the target data and can extract news articles, lists, structured records, and so forth. The harvested data is stored in a feed database, which serves as a staging area for data ingestion into ENTEL.

Information Integration and Aggregation (2)

The Ingest component takes the raw data from the feed database and integrates it into the system. It does this using a multi-step process where it first converts incoming data into raw RDF (Resource Description Framework) graphs centered around the entities appearing in the data. These RDF graphs are an intermediate data format which are then further processed to resolve entities and added to the ENTEL knowledge store. The nodes in the raw RDF graphs represent either entities or literals (e.g., the string "113 Main St"), and edges represent either simple attributes (e.g., "address") or relations (e.g., "CEO of" where the nodes at each end-point represent an entity).

Harvested data which is structured (such as lists and structured records) is directly converted into RDF graphs. Harvested data which is unstructured text is fed into natural language components (see the Fact Extraction component below) that extract facts and relations and return them as RDF graphs.

Fact Extraction from Unstructured Text (3)

In the first step of ingestion, any unstructured text is processed by natural language components to extract RDF graphs. At this point, the RDF is relatively raw, since the extracted entities are simply names mentioned in the text.

The fact extraction module is written as a plug-and-play component making it easy to integrate a variety of fact extractors. It integrates output from these extractors into a single RDF graph for each piece of text. We currently use multiple third-party natural language-based text-mining tools including the Reuters OpenCalais service[1] and the Semantex engine from Janya Inc.[2]

Entity Resolution (4)

Entity resolution is critical to ENTEL, as having entity resolved data is essential for relational or network analytics. As we mentioned earlier, there may be multiple ways to refer to the same entity, including aliases and more commonly, there can be misspellings and errors in the extracted data. Our EntityBase component (Knoblock et al., 2007) imports RDF descriptions of entities, and clusters them so that (ideally) each cluster corresponds to a distinct entity in the world. EntityBase maintains a name space where each cluster is associated with a unique *entity identifier*, referred to as an EID.

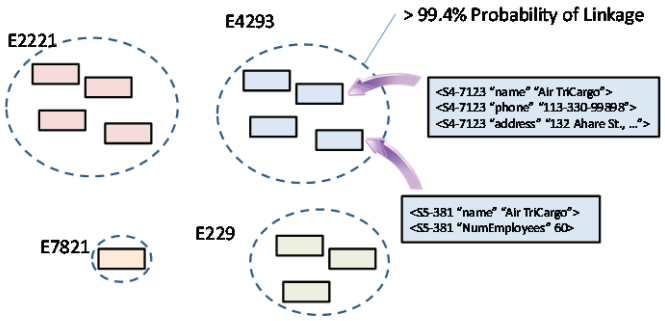

Figure 3: Clusters in EntityBase

[1] http://www.opencalais.com
[2] http://www.janya.com/products/the-semantex-platform/overview

The input to EntityBase is generated from the RDF graphs described earlier. In particular, the system decomposes the RDF to create a distinct sub-graph for each entity mentioned. We define a *descriptor* to be a set of RDF triples, (subject, attribute, object) describing an entity, where the subject is a *Descriptor ID,* the attribute is a predicate name, and the object is a literal. The Descriptor ID, or DID, is a unique ID for each descriptor. Each descriptor can be thought of as a database record, though as we have discussed, the descriptors in ENTEL are harvested from both structured sources and unstructured text.

Figure 3 illustrates our representation scheme. Each cluster consists of an EID and a set of descriptors. For instance, Entity E4293 is a cluster of 4 descriptors. On the right of the figure, two of these descriptors are expanded to show their details. The top descriptor consists of three triples with Descriptor ID[3] S4-7123, indicating a name, phone number and address. The second description has DID S5-381, and contains attributes for name and number of employees. Note that because multiple records can have the same attributes, there can be multiple names, addresses, etc. associated with any given entity.

An incoming descriptor is added to exactly one cluster, either a pre-existing cluster, or a new cluster that is created with a new EID. EntityBase's clustering is based on probabilistic inference. During the import process, the system estimates the probability that a descriptor belongs in a cluster, and only places it in the cluster if it exceeds a predefined probability threshold (e.g., 99.4% in Figure 3).

Note that the addition of new data can also provide evidence that existing clusters should be merged or split. We discuss this issue later in this paper.

The Knowledge Store and Analytics Engine (5)

Once we have resolved entities, we tag all the entity nodes in the RDF graph with their respective entity IDs. We use these to incorporate the data into a continuously growing network of entities -- a social network. The social network is stored as triples of the form <EID, predicate, EID> for relations between entities. These triples are stored locally, with the rest of the RDF triples, in the "Fact Store". We call each such triple a fact. The original content from the data feeds are stored in the "Content Store" with a reference pointer back to their original online location. We only store one copy of any single fact in our fact store, but keep an "attribution" mapping from each fact to the record it appeared in. The result is that we have a compact set of facts, which make up our entity graph for analytics, as well as a complete set of attributions so that we can always trace back where a fact came from. We also create and store metadata about each entity, such as its canonical name (currently the most frequent name) and when it was seen first and last.

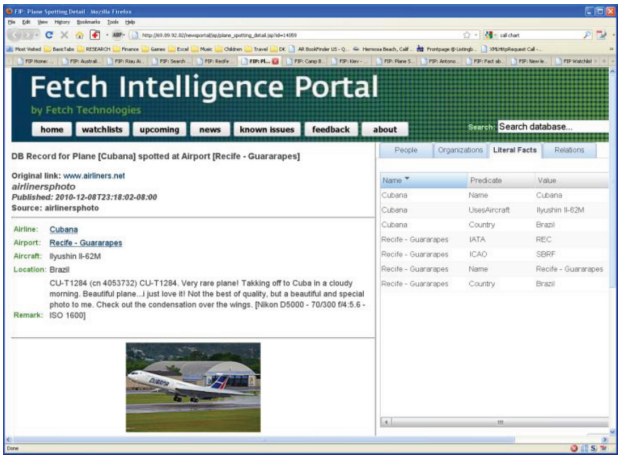

Figure 4: ENTEL Web Interface

As information is integrated into the knowledge store, ENTEL performs up-front analytics by executing "persistent queries" so that watchlists and other monitoring functions can be accomplished.[4] For example, ENTEL monitors incoming content to see if any entities of interest appear and flags them for the user. These lists and matches are stored in the "Analytics" database.

Web Interface

While the primary goal in ENTEL is the generation, maintenance and analysis of entity graphs, it also needs a user interface in order to make it useful as an application. We developed a front end to support various analytic use cases, some of which we have described above. An example page from the Web interface is shown in Figure 4 where it shows the details of a plane spotting record. This data was extracted from a semi-structured website (as opposed to being extracted from text). It contains useful information such as the name of an airline (with a link to the page about that airline), the name of an airport (with a link to the page about that airport), the type of plane, and even the tail number and serial number of the plane. Details of the record are shown in the left pane, together with the picture from the site. On the right we see some of the literal facts we pulled out of this record. The Web interface provides links to all entities, facts and relations as well as watchlists, profile pages and more. These capabilities depend on having entity resolved data.

[3] By convention, the Descriptor ID is composed of two hyphenated parts, the first part identifying a data source, and the second part identifying the data record within the source.

[4] Note that the watchlists themselves can be acquired from online sources (e.g., online sources of known terrorists, banned airlines, etc.) as well as from users who can use the Web interface to build their own lists.

Status and Performance

ENTEL is currently being evaluated by Air Force personnel. We note that the system relies on multiple technologies that are the subject of current AI research, including entity and fact extraction, and entity resolution. Because these components are not perfect, and can interact in unexpected ways, ENTEL sometimes makes mistakes that seem naïve. Nevertheless, our preliminary experiments show that the system can be quite useful. For example, in one recent experiment we harvested 217 text documents, and had a contractor mark up these documents (with a markup tool) to identify entities and facts, and to resolve the entities. This process took 66 man hours, and ENTEL processed these same documents in under 10 minutes. The fact extraction component extracted at least one RDF graph per document. A total of 2524 entity nodes were generated, of which 748 were identified as named persons and 1776 were identified as named organizations (companies, airports, airlines, etc.). In addition, 3583 attributes and 408 relations were extracted.

From this data, EntityBase produced 343 person clusters. One split and nine merges were required to revise this initial clustering to produce a "correct" clustering of 329 person entities (according to human judges). The average precision[5] across all person clusters was .997 and average recall was .987. As for organizations, a total of 586 clusters were created, which required 3 splits and 19 merges to produce a correct clustering of 571 entities. The average precision was .989 and average recall was .988.

We caution that these figures provide a very limited view of the system's performance. An in-depth empirical analysis is in preparation.

An Imperfect World: Splits and Merges

As described above, EntityBase clusters incoming data records and assigns each cluster a unique Entity Identifier, or EID. However, our knowledge about entities in the world is not perfect. When new information is acquired, evidence may suggest that the clusters should be merged or split up by EntityBase (Knoblock et al., 2007). For instance, earlier we explained that, over time, we might conclude that the aircraft in Figure 1 is the same as the aircraft reported to have tail number 3D-SPZ. In EntityBase, such a conclusion would result in the clusters representing the two entities being merged. Similarly, if we discovered that the registration number 3D-SPZ was used by two aircraft with different construction numbers, this would result in the entity being split.

In some cases, entities such as companies can also split and merge in the world, as in a corporate merger. For the purposes of this paper, we consider only split and merge operations that correct previous, imperfect clustering decisions. That is, these splits/merges constitute belief revisions intended to correct the system's belief state.

There are a variety of reasons why an incorrect clustering decision may be made. One reason is that the incoming data may be noisy. For example, a natural language extractor may misparse text that starts "Outspoken Qatar Airways chief Akbar..." and return "Outspoken Qatar Airways" as a named entity. Depending on the order in which facts are encountered, these types of parsing mistakes can result in separate clusters that need to be merged.

A second source of mistakes is due to the statistical entity resolution process, which makes heuristic probability estimates. For instance, the system may assume that "Iran Air" is the same as "Iran Air Force" due to an overly generous estimate that "Force" was simply missing in the mention of "Iran Air".

A third type of mistake occurs when there is simply not enough data to determine the correct clustering. For instance, we may have two texts that refer to a person named "Harold Knopf Merriweather", and only later encounter additional information indicating there are two different people with this name (unlikely as that may be).

While the first two sources of error can be addressed by developing more accurate extraction and resolution algorithms, the uncertainty due to incomplete data depends on what data we have about each entity, and the order that the data is ingested. In fact, one way to improve the performance of an entity resolution system is to first feed the system a source of clean, complete data, often called a *reference set*, so as to create a perfect initial clustering (where each cluster contains a single prototypical data description). Then, when descriptions from subsequent data sources are imported, they are simply inserted into existing clusters. This strategy is often the default approach used in commercial enterprise applications.

While EntityBase can accommodate this strategy, reference sets do not exist for every domain. In particular, consider the arms trafficking domain that we described earlier. We can feed the system a list of all the world's airports, since these are essentially unchanging. And we can feed the system a list of all the major airlines and their executives. However, there is no complete list of all the entities the system might encounter. New airlines, cargo companies, associated individuals, etc. can be expected to turn up in the shadowy world of arms trafficking, and they need to be tracked.

Thus, entity resolution mistakes that are due to incomplete data may be unavoidable in many domains. To handle these situations, EntityBase includes the capability

[5]Precision for an entity E is computed by identifying the cluster C_E which best corresponds to entity E, and then determining the proportion of descriptors that correctly refer to E in cluster C_E. Recall for entity E is computed as the proportion of descriptors that describe E which were correctly put into cluster C_E.

to merge and split clusters. In particular, the system can be configured to automatically split clusters that have many inconsistent attributes, and to merge clusters when two or more clusters closely match a new incoming description. In addition, analysts can also manually specify merges/splits through the Web Interface when EntityBase's probabilistic inference capabilities are insufficient.

As a consequence of merges and splits, EIDs are retired. When a merge happens between two or more formerly distinct clusters, the EIDs of the old clusters are retired, and a new EID is created for the new cluster. Similarly, when a split occurs, the old EID is retired, and new EIDs are created for the new clusters.

Maintaining Referential Integrity

Merges and splits can create difficulties for client systems that interact with EntityBase. In particular, when EIDs are retired and new clusters created, this constitutes a form of belief revision that a client system may need to be aware of. In this section, we describe two interaction modes that we developed to support EntityBase clients, and their advantages/disadvantages.

The first mode of interaction, which we call "Refer-by-Identifier", supports EntityBase clients that need to store and use Entitybase's EIDs. Consider, for instance, the social network maintained by ENTEL's Analytics Engine and Knowledge Store, which represents the relationships between entities. ENTEL naturally relies on EntityBase's EIDs to represent the nodes in the network. However, this dependency comes at a cost. When entities in EntityBase merge/split, the social network must be similarly updated to maintain referential integrity. Otherwise, when EIDs are retired, the network would contain EIDs that would, in effect, be dangling pointers. This loss of information would in turn degrade the performance of the analytics. Moreover, any intelligence report based on the network critically depends on the network being as accurate and up-to-date as possible.

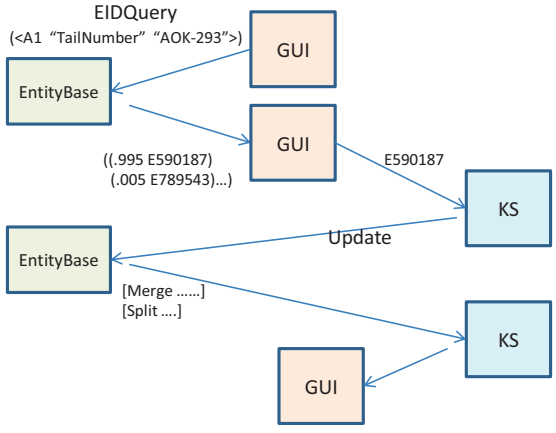

Figure 5: Refer-by-Identifier, Example Interaction

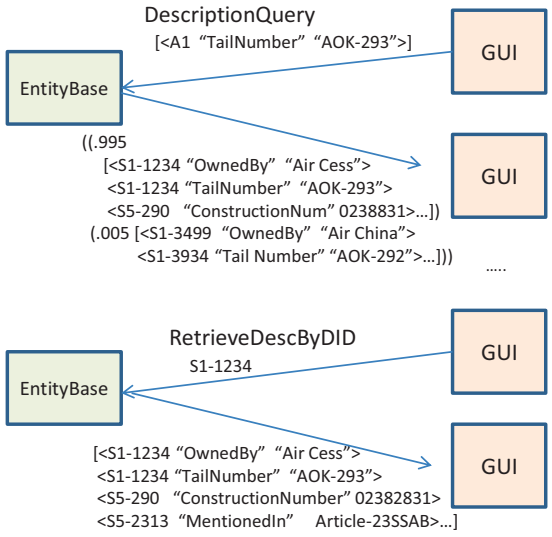

Figure 6: Refer-by-Description, Example Interaction

Below we summarize the main API functions that EntityBase supports to implement Refer-by-Identifier:

EIDQuery(query)
    returns ((probability, EID) (probability, EID)..)
RetrieveDescByEID(EID)
    returns SetOfDescriptors
Update(LastUpdateID)
    returns (SyncOperation, SyncOperation,…)

The first call, QueryForID, enables a client to find the EIDs of all entities that match a query (a partial description of an entity). The second call retrieves, for any EID, the full set of the descriptors associated with an EID. The last call is used to maintain referential integrity with EntityBase. A SyncOperation is simply a specification of a merge/split enabling the client to update its own database to maintain consistency with EntityBase.

One pattern of interaction using Refer-by-Identifier is shown in Figure 5. The user issues a query via the GUI to find a plane with tail number AOK-293. EntityBase finds two possible matches, one of which is a close match, E590187. The user then wants to view the relationships between E590187 other entities in the social network. To achieve this, the GUI calls the Knowledge Store, which updates the social network to make sure it is consistent with EntityBase, and then displays the entities related to 590187 (the local neighborhood in the graph) to the user.

A client that uses Refer-by-Identifier is responsible for synchronizing its belief state to be consistent with EntityBase. For instance, consider the process of maintaining the social network. When a split/merge occurs, the client must rebuild the relations that refer to any entity whose EID has changed. In particular, when a merge occurs, all of the links associated with the two old EIDs must be updated to reflect the EID of the new merged

entity. Similar, with a split, the system must examine every relation associated with the old EID and determine which of the new EIDs should replace it. This requires examining the attribution mapping for that relation to identify the descriptor that was responsible for creating the relation, and then determining which new EID is now associated with that descriptor.

Obviously, synchronization has a cost, not only in time, but also in terms of the effort involved in implementing the synchronization code. Refer-by-Description is designed to shield clients from these costs. Refer-by-Description allows clients to refer to clusters indirectly, via the descriptors encompassed by the cluster. To illustrate this, let us consider another ENTEL use case. In particular, consider a simple Web GUI for querying EntityBase, where one could, for instance, ask for all companies owned by Victor Bout. This type of GUI simply needs to display each matching entity to the user, and allow the user to click on an entity if he/she wants to find out more details. The GUI may allow the user to bookmark a set of "favorite" entities (i.e., creating a simple form of a watchlist). For such a lightweight application it is be prohibitively expensive to implement the synchronization process. On the other hand, if a merge/split does indeed happen, we would also prefer not to throw an error if the user selects an entity that has been merged/split.

To support this type of use case, EntityBase allows clients to refer to an entity cluster using any Descriptor ID in the cluster. Specifically, EntityBase supports Refer-by-Description through the following calls:

```
DescriptionQuery(Query)
        returns ((probability, SetOfDescriptors)
                (probability. SetOfDescriptors)..)
RetrieveDescByDID(DescriptorID)
        returns SetOfDescriptors
```

The first call enables a client to find all the entities that match a query. The complete set of entity descriptors are returned, rather than EIDs, since in the Refer-by-Description paradigm the client does not retain EIDs. (That is, the system returns the clusters themselves, rather than their EIDs.) The second, RetrieveDescByDID, gives clients the capability to "follow up" on an initial query. For instance, returning to our GUI example, the user may search for information about a plane with tail number AOK-293. As shown in Figure 6, the GUI sends a query to EntityBase, which returns two potential matches to display to the user, one of which is an exact match to the tail number, and one which is a close match. Notice that EntityBase sends the descriptions of the entities, not the EIDs. Later, the user might issue a followup query to find out more about that aircraft. Without an EID to refer to, how can the GUI phrase this query? Since each predicate in the entity descriptors includes a DescriptorID assigned by the source, the GUI can select any individual DescriptorID (in the figure, the selected DescriptorID is S1-1234), and via RetreiveDescByDID, map this back to its current entity (which could have changed in the meantime) and obtain a full entity description. This process effectively shields the client from the responsibility of maintaining EIDs.

The advantage of Refer-by-Description is that the client need not implement synchronization. A DescriptorID functions as an pointer to the cluster in which the descriptor belongs. We note that our original design for ENTEL was based solely on Query-by-Description, since we believed that we could confine the complexity of splits/merges to EntityBase. However, once we began building more sophisticated clients, we found that the need for synchronization favored Refer-by-Identifier.

In fact, we can classify clients into two categories, depending on how they use EntityBase. Formally, if we describe EntityBase's knowledge state by a set of formulas E, then a series of split/merge operations will transform state E_1 to state E_2. After these operations, a client with state C_1 that was consistent with E_1 may be forced to revise its state to C_2 to preserve consistency with E_2. For clients that may need to make such revisions, Refer-by-Identifier is likely to be an appropriate strategy because it supports synchronization. On the other hand, if a client never needs to revise its own state to preserve consistency, then Refer-by-Description is a viable approach, and is simpler to implement.

Related Work

Mechanisms for maintaining referential integrity are of growing importance as the Web matures and using distributed knowledge becomes a real possibility. As we have pointed out, it can be painful to enforce referential integrity. Because of the costs involved, many application designers choose to simply ignore referential integrity; instead, they simply throw an error when an out-of-date ID is dereferenced. For instance, on the Web, when a URL is out of date, Web browsers often simply report a 404 "Page Not Found" error, and users have learned to understand what this means. However, it is possible to do better, and in certain applications, such as the ENTEL arms trafficking domain, referential integrity is critical if the application is to be useful.

The issues we have raised here will eventually be a concern for Semantic Web applications. Using OWL, the owl:sameAs predicate can be used to declare that two different URIs denote the same thing. If the statement is later discovered to be incorrect, it can be retracted. Some systems, such as the BigOWLIM semantic repository (Bishop et al., 2007) include special purposes optimizations for handling SameAs inferences and for

handling retractions, both of which can be potentially expensive. However, approaches for efficiently revising beliefs about entity identity on the Semantic Web have only just begun to receive consideration (Glaser, Jaffri, & Millard, 2009).

Many modern relational database management systems offer automated processes for enforcing referential integrity. In a relational database, referential integrity refers to the requirement that each foreign key value in a table exists as a primary key in the referenced table. One enforcement mechanism is to simply issue a warning whenever referential integrity is violated. Some systems offer alternative enforcement mechanisms such as "cascading deletes", which upon identifying an integrity violation automatically deletes offending rows as well as any offending rows in related tables. The simplistic approaches offered by traditional database systems are based on the assumption that we have perfect knowledge about entities. In the application described here, as well as many other practical applications, this is clearly not practical.

To handle noisy data, some commercial firms offer more sophisticated Master Data Management approaches, which allow data to be matched against clean reference sets or "master data lists" (e.g., White et al., 2006) which are carefully expanded over time. However, while this approach accommodates noise in the incoming data, ENTEL addresses an even more challenging situation. In our case, there is no known reference set. Instead, data is automatically aggregated from noisy, uncertain data sources over time. It is not acceptable to discard incoming data even when it is uncertain which entity the data refers to. At some later point, the system may find additional facts which allow previously-acquired records to be linked.

Ultimately, our application requires a form of belief revision. AI researchers have proposed a variety of general formalisms for sophisticated belief maintenance and belief revision (e.g., Hunter & DelGrande, 2011). However, the practical issues involved in designing software systems that perform belief revision remain challenging. Fortunately, ENTEL does not need to revise arbitrary inferences. In our application we limit our focus to dynamic entity resolution and its ramifications. Our approach offers the choice of minimal support for entity splits/merges, or alternatively more extensive maintenance of entity IDs in a shared namespace.

Conclusion and Discussion

This paper introduced ENTEL, a system that integrates several AI technologies to help intelligence analysts track and analyze information being published on the Web. The system has been aggregating live data over the past few months, and is currently being deployed so that analysts in the Air Force can informally try using it in practice.

As we described, entity resolution is an important capability but also introduces complexity. Persistent entity identifiers are necessary for advanced analytics, such as network analysis, but changing those identifiers due to belief revision has downstream ramifications. Whereas in other applications, referential integrity may not be critical (witness Web links that can become out-of-date), in intelligence applications keeping track of entity references is critical. In fact, in ENTEL, splits/merges are most likely to occur when deception is being practiced. These are the most interesting and challenging cases and it is precisely on these cases where the system must perform well.

Acknowledgements

The project described here was based partly upon work at Fetch Technologies supported by the Air Force Research Laboratory under contracts FA8750-10-C-0055, FA9550-09-C-0064 and FA8750-09-C-0015 and by DARPA under contract W91CRB-11-C-0037. The opinions and findings expressed here are solely those of the authors.

References

Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R. (2011) OWLIM: A family of scalable semantic repositories, Semantic Web, (Pre-Release), DOI: 10.3233/SW-2011-0026

Glaser, H., Jaffri, A. and Millard, I. (2009) Managing Co-reference on the Semantic Web. WWW2009 Workshop: Linked Data on the Web (LDOW2009).

Hoffman, F.G. (2006). Complex irregular warfare: The next revolution in military affairs. *Orbis, 50*(3), 395-411.

Hunter, A.,and Delgrande, J. P. (2011) "Iterated Belief Change Due to Actions and Observations", Journal of Artificial Intelligence Research, Volume 40, pages 269-304

Knoblock, C.A., Ambite, J.L., Ganesan, K., Muslea, M., Minton, S., Barish, G., Gamble, E., Nanjo, C., See, K., Shahabi, C. and Chen C.C. "EntityBases: Compiling, Organizing and Querying Massive Entity Repositories", Proc. of the International Conference on Artificial Intelligence (ICAI'07) , 2007.

Koudas, N., Sarawagi,S., and Srivastava, D. 2006. Record linkage: similarity measures and algorithms. Proc. SIGMOD 2006

Muslea, I., Minton, S.N. & Knoblock, CA (2001) Hierarchical Wrapper Induction for Semistructured Information Sources. Autonomous Agents and Multi-Agent Systems 4(1/2): 93-114

Ticrea, S.I &, Minton, S. (2003) Inducing Web Agents: Sample Page Management. Proc. Information and Knowledge Engineering - IKE 2003: 399-403

White, A., Newman, D., Logan, D. and Radcliffe, J. (2006) Mastering Master Data Management. #G0013695, Gartner Group