ARMY RESEARCH LABORATORY

U.S. Army Research Laboratory Fiscal Year 2010 Status Report for the Advanced, All-Source, Fusion Technology Program Annex

by Gary Moss, Mark Thomas, and Mark Mittrick

ARL-TR-5489 March 2011

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-5489 **March 2011**

U.S. Army Research Laboratory Fiscal Year 2010 Status Report for the Advanced, All-Source, Fusion Technology Program Annex

Gary Moss, Mark Thomas, and Mark Mittrick
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	***Form Approved OMB No. 0704-0188***

1. REPORT DATE *(DD-MM-YYYY)* March 2011	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** 1 October 2009–30 September 2010
4. TITLE AND SUBTITLE U.S. Army Research Laboratory Fiscal Year 2010 Status Report for the Advanced, All-Source, Fusion Technology Program Annex		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gary Moss, Mark Thomas, and Mark Mittrick		**5d. PROJECT NUMBER** 1TEP11
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-CII-C Aberdeen Proving Ground, MD 21005-5067		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-5489
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The U.S. Army Research Laboratory (ARL), via its participation in the Advanced, All-Source, Fusion Technology Program Annex (TPA) CE-CI-2010-02, is undertaking research in information fusion to assist the U.S. Army Intelligence Community. This TPA establishes collaboration with the Communications-Electronics Research, Development, and Engineering Center Intelligence and Information Warfare Directorate. Under this agreement, ARL is developing an integrated set of Web-enabled, service-oriented applications collectively named the Relationship Discovery Service (RDS). The purpose of the RDS is to provide a set of tools for the intelligence analyst or military decision maker to make sense of large amounts of soft intelligence data. In particular, the RDS should provide the means to rapidly identify high-valued individuals and facilitate collection of information on persons of interest over time.

15. SUBJECT TERMS
A2SF, data fusion, all-source, DCGS-A, HUMINT

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Mark Mittrick
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	30	**19b. TELEPHONE NUMBER** (*Include area code*) 410-278-4148

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

The U.S. Army Research Laboratory (ARL), via its participation in the Advanced, All-Source, Fusion (A2SF) Technology Program Annex (TPA) CE-CI-2010-02, is undertaking research in information fusion to assist the U.S. Army Intelligence Community. This TPA establishes collaboration with the Communications-Electronics Research, Development, and Engineering Center (CERDEC) Intelligence and Information Warfare Directorate (I2WD). Under this agreement, ARL is developing an integrated set of web-enabled, service-oriented applications collectively named the Relationship Discovery Service (RDS). The purpose of the RDS is to provide a set of tools for the intelligence analyst or military decision maker to make sense of large amounts of soft intelligence (HUMINT) data. In particular, the RDS should provide the means to rapidly identify high-valued individuals (HVI) and facilitate collection of information on persons of interest over time.

The RDS was developed to address issues related to the counter-insurgency problems of current asymmetric warfare. Force-on-force techniques are not sufficient against an adversary who is mobile, tactically amorphous, and stealthy. To rapidly understand the current situation, commanders require the ability to track HVIs within their areas of operations. This situational understanding then increases their ability to anticipate their opposition and helps them formulate plans that mitigate risk.

The RDS was designed to facilitate the identification of explicit links between individuals, organizations, and events—in essence, the "relationship discovery" problem. Designed to be a composable system, the RDS is a collection of web applications, each performing a vital role in the operation of the whole. All data access from the constituent web applications is provided via a web service interface.

Data are modeled using the World Wide Web Consortium,[*] Semantic Web,[*] knowledge representation standard Web Ontology Language (OWL).[*] OWL is used to describe "ontologies" that are basically class hierarchies defining entity types and their attributes (called "properties"). OWL is based upon Resource Description Framework (RDF)[*] and has various standard serializations, such as RDF/XML,[*] N-Triples,[*] Notation3,[*] and Turtle,[*] which can be imported and exported by the RDS. The RDS uses the open-source Jena (http://jena.sourceforge .net/) library for importing/exporting these formats, storing the data, and all forms of database access.

[*]Note: For more information on any of these terms, please see <http://wikipedia.org>.

The RDS has been developed over 3 years. Serving as a research and development platform under the Soft Target Exploitation and Fusion (STEF) Advanced Technology Objective (ATO), it was designed to utilize data extracted from HUMINT reports. These reports contain information about humans that was collected by humans. They are characterized by a structured format with unformatted content. As such, the research issues involved in reasoning over these data include extraction, correlation, entity resolution, storage, ontology, and large dataset processing. ARL research was focused on the storage, ontology, and graph matching techniques. Also provided are social-network-style visualization with SNA metrics and a SPARQL[*] query interface.

The A2SF research effort by CERDEC is an expansion of the STEF research. Integrating data from "hard" sensors with the "soft" data of HUMINT, the research issues are more complex. ARL has chosen to pursue a spiral develop/test/develop approach, adding new types of data sources systematically, since there is no current methodology for combining these disparate data formats.

The task deliverables for fiscal year (FY) 2010 were to extend the RDS for additional data sources and to identify and acquire an all-source dataset. This report describes ARL efforts to add new inputs to the RDS, the issues involved, solutions, and further work.

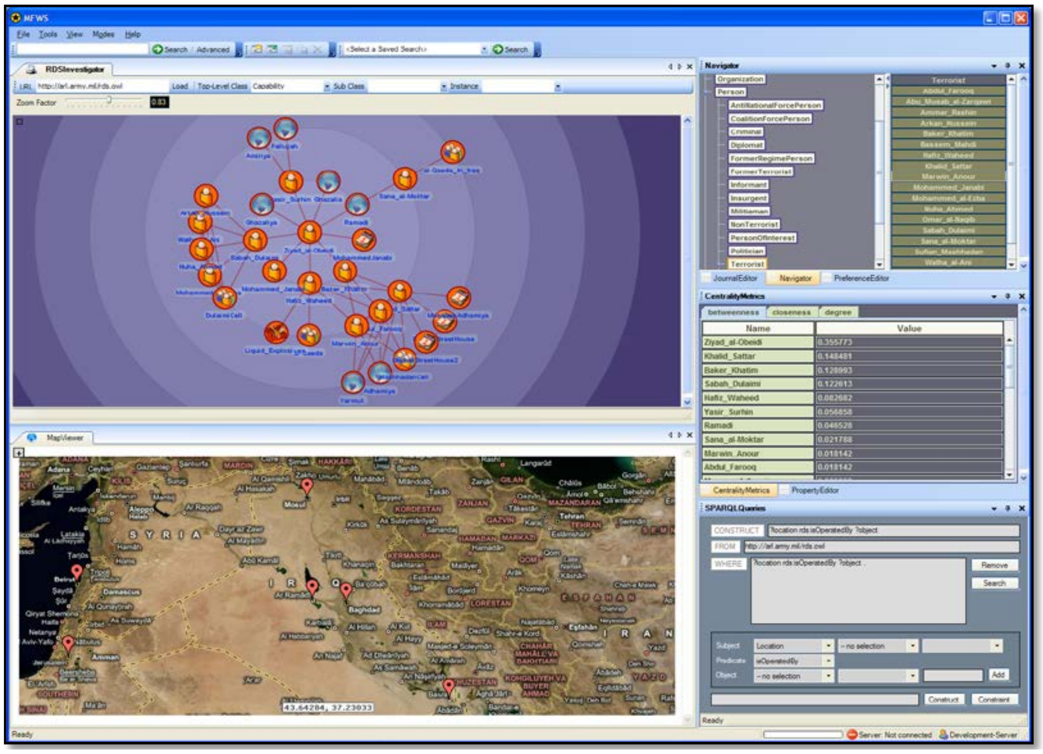
2. Relationship Discovery Service

2.1 Technical Evolution

2.1.1 DCGS-A Application Framework to Fusion Exploitation Framework

The RDS was originally developed under the STEF ATO in support of the Army Distributed Common Ground Systems (DCGS-A). For the duration of the STEF program (FY07–09), the DCGS-A, in phase 3 of development, required that all software utilize the DCGS-A Application Framework (DAF) (a.k.a. "Viper") for the development of user interface components. The DCGS-A DAF is based upon the Microsoft Visual Studio .NET/C# integrated development environment. User interface widgets created with the DAF are known as *Viper Components*, and they were designed to be combined in a portal-like Viper application called the *Multi-Function Workstation*. Communication between Viper Components was facilitated by the *Collaboration Data Manager*, which is basically an object-oriented pub/sub-messaging bus that facilitated the sharing of *serializable* data structures. An example of the RDS visual display is shown in figure 1.

[*]SPARQL is a recursive acronym that stands for SPARQL Protocol and RDF Query Language.



Name	Value
Ziyad_al-Obeidi	0.355773
Khalid_Sattar	0.148481
Baker_Khatim	0.128993
Sabah_Dulaimi	0.122613
Hafiz_Waheed	0.082682
Yasir_Surhin	0.056858
Ramadi	0.046528
Sana_al-Moktar	0.021788
Marwin_Anour	0.018142
Abdul_Farooq	0.018142

CONSTRUCT ?location rds:isOperatedBy ?object
FROM http://arl.army.mil/rds.owl
WHERE ?location rds:isOperatedBy ?object .

Figure 1. DCGS-A application framework version of the RDS.

Beginning in FY10 with the inception of the A2SF program, the DCGS-A development community is trending away from use of the DAF and C# to an open-source, Java-based development environment called the Fusion Exploitation Framework (FEF), which is being developed by defense contractor Potomac Fusion, Inc. (http://www.potomacfusion.com). The management of data input sources, data fusion, or other types of data manipulation can be implemented as composable FEF fusion modules. The recommended platform for deploying user interface components is now the Ozone Widget Framework, which is typically known simply as *Ozone*.

2.1.2 Automated Data Input

The role of the RDS throughout the STEF program was not to tackle the problem of extracting information from raw, unstructured text messages. The text extraction problem was to be worked under contract to CERDEC/I2WD by Orbis Technologies, Inc.

The RDS database was originally populated by data that were manually extracted. The 100-message set, developed under contract by Jeffrey D. Simon to CERDEC/I2WD**,** was used extensively for software development and testing. The dataset was manually extracted using the Protégé Ontology Editor using an ARL-developed ontology.[1]

[1]Apenyo, K. *Relationship Discovery Service (RDS) User Manual*; ARL-MR-0722; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2009.

However, the RDS needed a means to acquire data automatically. The RDS was capable of providing automated assistance to relationship discovery, but the input process was labor intensive. The long-range plan was that the product of the STEF program’s text extraction engine would be machine-readable data triples (subject, predicate, and object) conforming to RDF and OWL, typically referred to as RDF/OWL. Ultimately, the RDS was designed to be able to operate upon RDF/OWL and to use it as its data transmission standard. The effort at the outset of the A2SF program was to develop an automated data input mechanism for the RDS. As automated extraction was being researched and developed by CERDEC, it was desirable to explore other methods of data input without the requirement for “heavy” extraction. This report describes ARL efforts to solve this problem using tagged data.

2.1.3 Small to Large Datasets

RDS was developed completely with the Jeffrey Simon 100-message set (see figure 2). This provided a source of HUMINT data with ground truth for scientific investigations. The 100-message set, however, is small (19,632 words—all text—unstructured) in comparison to information systems, which process transactional data, for example. The experimental rigor of the 100-message set was overshadowed by its small size. Therefore, ARL focused on acquiring larger sets of data with ground truth, as well as some that had no ground truth but were desired for sheer size. The scalability of the RDS needed to be tested, and this required larger numbers of nodes and links.

Report Number: IIR 0096

Report Type: IIR

Entered By Individual: SGT Juan Gonzalez

DOI: 2007-01-06 13:00:00

Date Entered: 2007-01-06 23:32:03

Subject: Al-Qaeda Reading Material

Source: U.S. Soldier's Interview with Bookstore Owner

Summary: Bookstore owner on Dhubat Street in Adhamiya said many customers are asking if he has any books, magazines, or other material on al-Qaeda.

Text: A bookstore owner on Dhubat street in Adhamiya //MGRSCOORD: 38S MB 42493 94732 told a soldier that many customers are requesting material about al-Qaeda. This included inquiries about books, magazines, leaflets, etc. A few customers requested an al-Qaeda manual that he did not have. One man came into the store around 1000Z wanting to know where he could get DVDs or other video material about al-Qaeda. The bookstore owner told him to search the Internet.

Remarks: *Interpretation By SGT Barry Jones*: The increased demand for information on al-Qaeda by residents in Adhamiya is not surprising. This is related to the growing popularity of the al-Qaeda in Iraq terrorist organization and the increased number of attacks by Shiite militias on Sunnis in Adhamiya. The Sunnis are also feeling marginalized in the predominately Shiite-government and by the presence of Shiite extremists among the Iraqi police.

Figure 2. Example 100-message set text file.

The Ali Baba dataset was developed by the U.S. National Security Agency by Mark Jaworowski and Steve Pavlak in 2003[2] (see figure 3). The dataset consists of two scenarios.

```
Police Report 35
25 September 2003
London police are still searching for the man identified as Phil Salwah.  According to one of their
police informants, Salwah was approached by what was likely a recruiter for a terrorist group.
```

Figure 3. Example Ali Baba dataset message.

The first scenario is a collection of nearly 1000 unclassified, fictitious documents in Microsoft Word format replicating intelligence reporting on the actions of suspected terrorist activity in southern England. Among these documents are police reports, intelligence information reports (local HUMINT and detainee reports from abroad), Foreign Broadcast Information System (FBIS) reports, and local communications intercept (TACREP) messages. Approximately 75 of these 1000 messages provide the heart of the database—that is, the planning and activities of the main terrorist cell of the fictitious Ali Baba organization occurring over the course of several months. Several hundred more messages relate to the communications and activities of peripheral members of this terrorist cell to include friends and associates. Another several hundred messages represent innocuous but potentially interesting chatter predominantly from college campuses at Cambridge, Oxford, and others.

The second scenario is a collection of nearly 3000 unclassified, fictitious Word documents replicating intelligence reporting on the actions of suspected terrorist and insurgent activities in the mythical Middle-Eastern country of Pavlakistan. Among these documents are police reports, intelligence information reports (local HUMINT and detainee reports from abroad), FBIS reports, and TACREPs. Approximately 60 of these 3000 messages provide the heart of the database—that is, the planning and activities of nine terrorist and insurgent cells operating in Pavlakistan over the course of several months. Several hundred of the 3000 messages relate to potentially significant activities of suspected members of various terrorist cells and insurgent groups. The remaining ~2500 messages represent innocuous but potentially interesting chatter from locations throughout Pavlakistan.[2]

Although the Ali Baba dataset is fairly large, diverse, and has ground truth, the data are not marked up and contain errors (spelling, redundancy, etc.).

ARL has been focusing primarily on scenario 1. The steps taken to get the original Ali Baba dataset into RDS are outlined as follows:

1. Loaded the Ali Baba (scenario 1) dataset into a MySQL database to be processed.
2. Extended the STEF ontology in order to accommodate all of the new Ali Baba data (i.e., Message Text, etc.)

[2]Jaworowski, M.; Pavlak, S. Ali Baba Dataset Ground Truth; U.S. National Security Agency: Fort Meade, MD, 2003.

3. Developed code to read data from the MySQL database, as well as the individual messages, and to convert them into the necessary RDF N-Triple (.nt) format. This required the conversion of all the Ali Baba unstructured intelligence data reports from text (.txt) files and scrap (.shs) files into Adobe PDFs (.pdf) and Microsoft Word Documents (.doc).

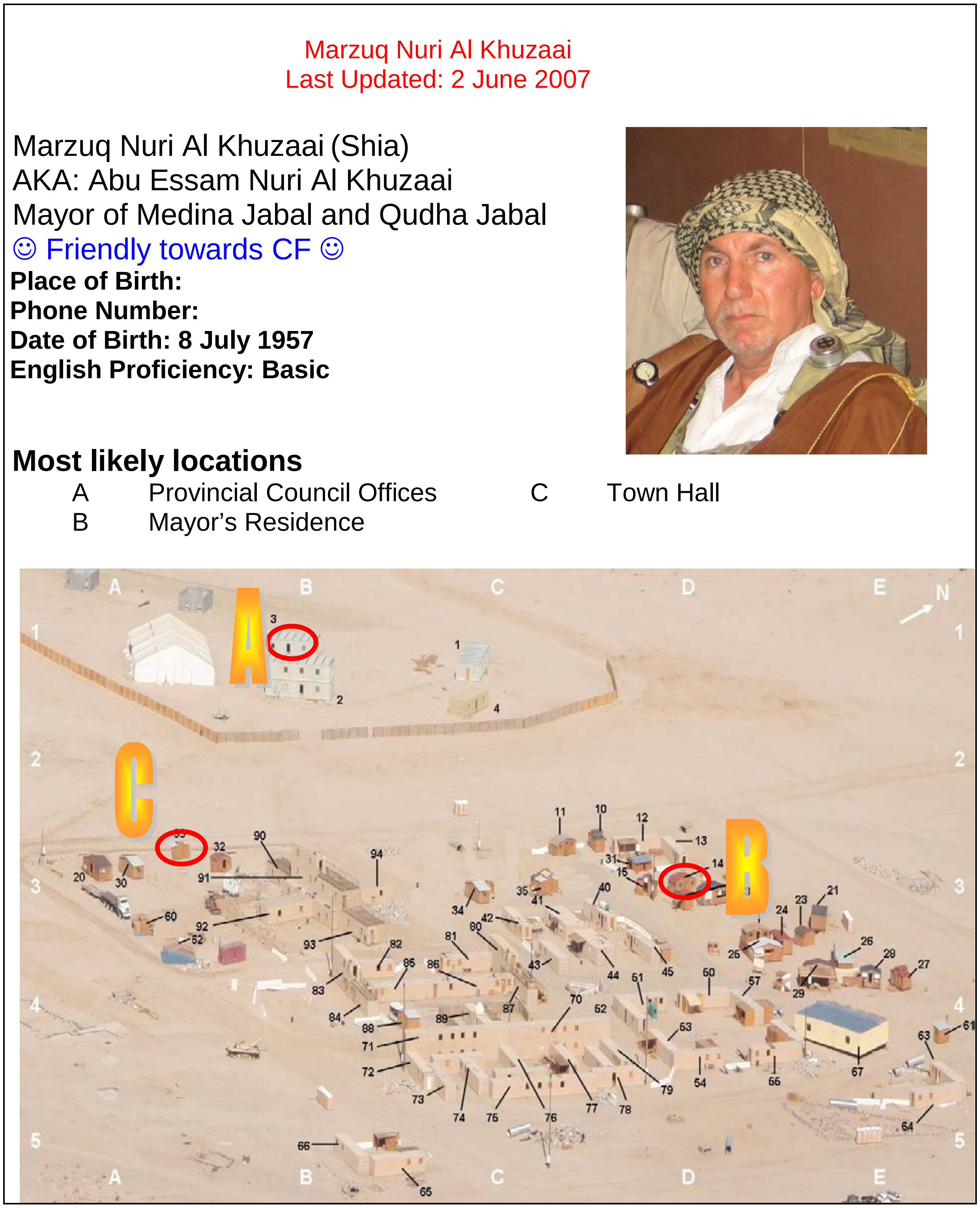
4. Tested/fixed/refined the Ali Baba RDF data (currently 27,000 lines of code and growing). This included fixing various parse errors in the RDF due to instance names not being proper *qualified names* (QName).* A QName cannot contain special characters, such as single and double quotes and cannot begin with a digit. It also included the process of adding and testing the inverse-relationship RDF code for the Ali Baba dataset (i.e., if a message entity contains a reference to a person entity, that person should contain a property that points back to the message).

5. Verified that the conclusions drawn by RDS match the Ali Baba (scenario 1) dataset ground truth.

The U.S. Army Research Office, under the direction of Dr. John Lavery, commissioned the development of a ground truth dataset based on Army exercises at the U.S. Army National Training Center. Named Hard/Soft-Test Nucleus (HASTEN-1) (see figures 4 and 5), it is an all-source dataset of ~10 terabytes of imagery, text, and unattended ground-sensor tracks. ARL received this dataset in February 2010 and is evaluating the contents. Many of the data formats are unfamiliar to ARL Tactical Information Fusion scientists, so ARL is in the process of procuring the tools (Analyst Notebook, MATLAB, Google Earth, ENVI Freelook, i2 ChartReader, and Quantum GIS) required to fully utilize the dataset.

Figure 4. Example HASTEN-1 dataset document.

*QName, or qualified name, is a valid identifier in RDF. For more information, see <http://en.wikipedia.org/wiki/QName>.



Marzuq Nuri Al Khuzaai
Last Updated: 2 June 2007

Marzuq Nuri Al Khuzaai (Shia)
AKA: Abu Essam Nuri Al Khuzaai
Mayor of Medina Jabal and Qudha Jabal
☺ Friendly towards CF ☺
Place of Birth:
Phone Number:
Date of Birth: 8 July 1957
English Proficiency: Basic

Most likely locations

A	Provincial Council Offices	C	Town Hall
B	Mayor's Residence		

Figure 5. Example HASTEN-1 dataset document with imagery and text.

Pictured in table 1 are various dataset details presented in table format for comparing/contrasting purposes. As shown in the table, each dataset has pros and cons. For example, the "100-Message Set" has ground truth but has a very small sample size. The "HASTEN-1" dataset is composed of various types of data that can be extremely useful, but due to the sheer volume of the dataset and the different tools required to view everything, it is a daunting task to undertake. Utilizing a combination of these datasets provides for the best results when testing RDS. The ability to continue to acquire new diverse datasets in the future will be the key to enduring success.

Table 1. Various datasets.

	100-Message Set	**Ali Baba Dataset**	**HASTEN-1 Dataset**	**Twitter Dataset**
Location	Iraq	Southern England Pavlakistan	Iraq	Variable
POC	Jeffrey D. Simon	National Security Agency Mark Jaworowski and Steve Pavlak	Defense Advanced Research Projects Agency U.S. Army Research Office (Dr. John Lavery)	Twitter.com
Date Ranges	January 2007–May 2007	2003	2007	Variable
Ground Truth	Yes, available	Yes, available	Under evaluation, partially available (Threads exist, need to be interpreted)	No, not available (Dynamic, need to be interpreted)
ISR Category	HUMINT	HUMINT/ FBIS/ TACREPS	HUMINT/ OSINT/ SIGINT/ GEOINT/ MASINT	HUMINT/ GEOINT
Theme	Intelligence/ military	Intelligence/military	Intelligence/military	Variable
Sensor/Source	NA	NA	Various sensors	Open source
Format/Schema	Plain text (.txt) XML (.xml)	Word files (.doc) PDF (.pdf)	Image files (.bip)(.bil)(.bsq)(.hdr)(.snc) (.ln1)(.pkt) Plain text (.txt) Database files (.sql) Video files (.avi)	Plain text (.txt) XML (.xml)
Classification	Unclassified	Unclassified	Unclassified// For Official Use Only	Unclassified
Size/Count	100 messages/ 400 kilobytes total	1000 messages scenario 1 3000 messages scenario 2 85 megabytes total	10 terabytes	Scalable
License/ Ownership	U.S. Department of Defense (DOD) use	DOD use	DOD use	Public

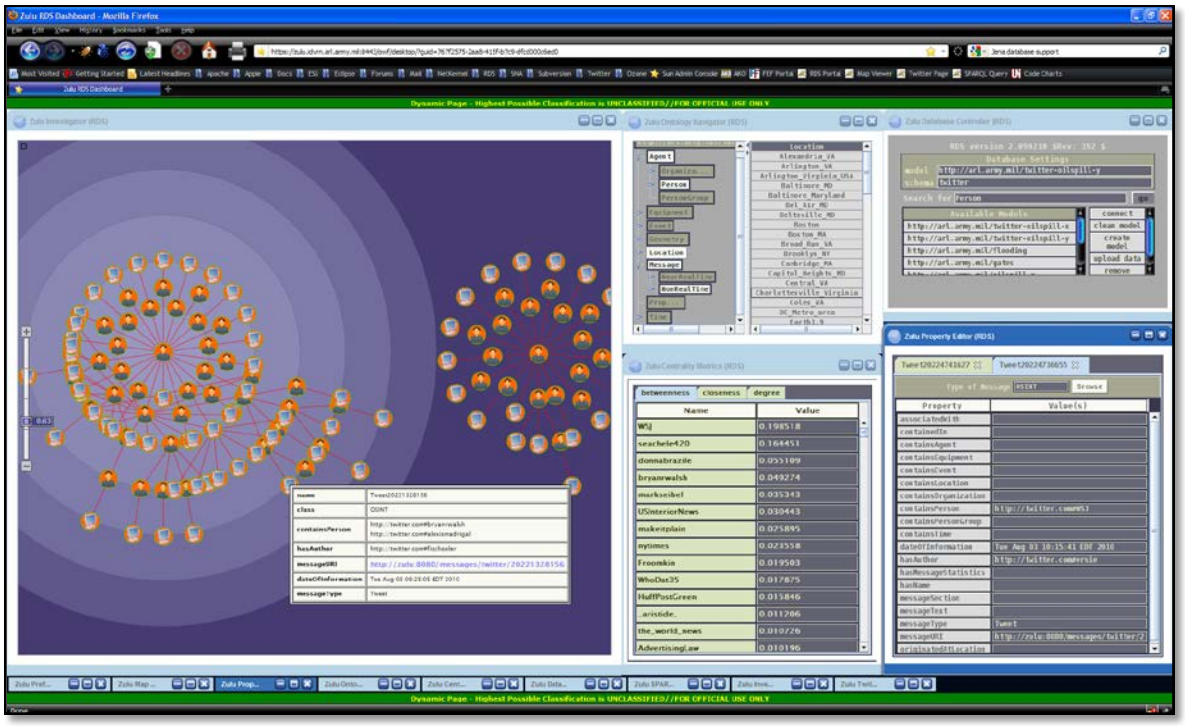
Note: NA = not applicable.

Automatic processing of data from unstructured sources is an active area of research. Typical techniques involve natural language processing, which is promising but several years away from a practical solution. ARL identified a practical and currently available solution using a data source that has structured headers, with tagged input. The Twitter (http://twitter.com) message type is one such format. The Twitter Application Programming Interface (API) (http://apiwiki .twitter.com/Twitter-API-Documentation) provides public access to user messages (called *status updates*) and associated metadata, such as the author, date and time the message was created, and geographic information. The Twitter processing will be described in detail later, and results of experimentation using the API will be presented.

2.2 Cloud Computing

2.2.1 Ozone Widget Framework

Ozone is a web-browser-based widget container or web portal that is implemented primarily in JavaScript. Communication between widgets is provided by a JavaScript library, which implements a pub/sub messaging bus where the payload can be either text strings or JSON (http://www.json.org) objects. This year, the RDS was migrated to Ozone (see figure 6) from the DAF. Immediately, there was a large benefit due to it being a pure web application—there is no longer any software that needs to be installed on the user's machine. The RDS now requires four servers: Apache Tomcat (http://tomcat.apache.org), MySQL (http://mysql.com), NetKernel (http://www.1060research.com/netkernal/roc), and Ozone.

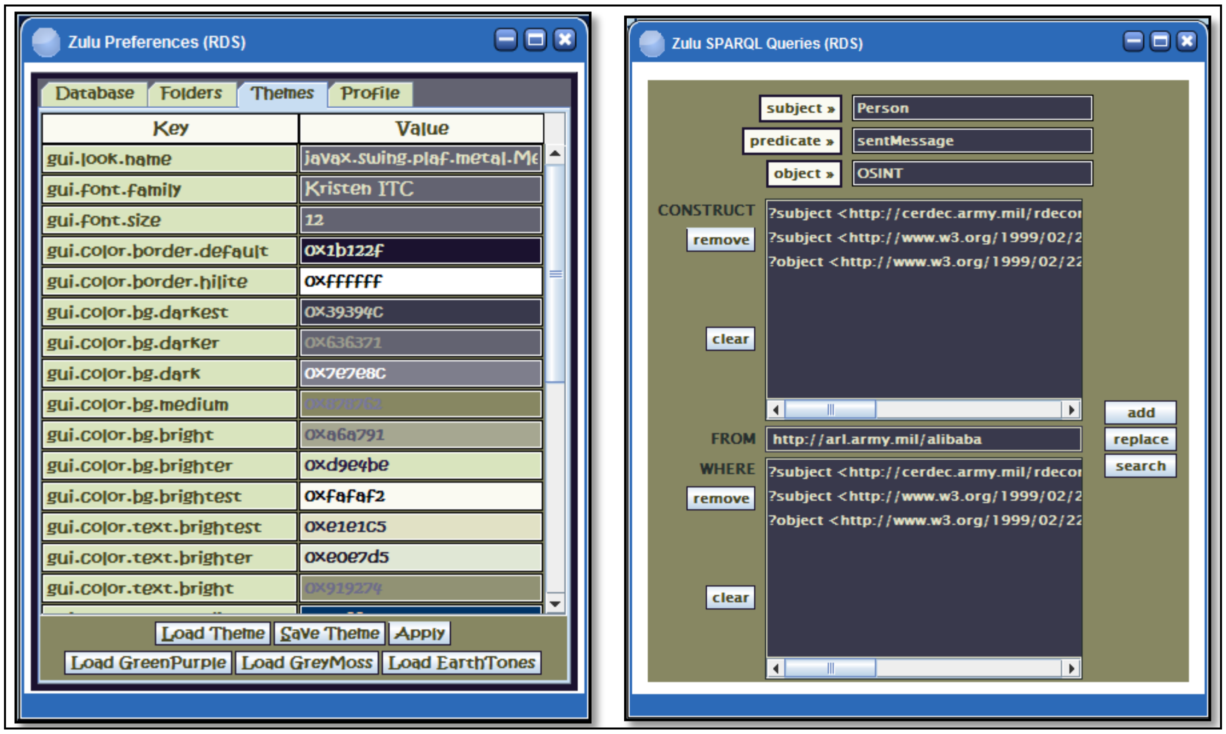

Zulu RDS Dashboard - Mozilla Firefox
Dynamic Page - Highest Possible Classification is UNCLASSIFIED//FOR OFFICIAL USE ONLY
Zulu Centrality Metrics (RDS)
betweenness
closeness
degree
Zulu Property Editor (RDS)

name	Tweet20221328156
class	OSINT
containsPerson	http://twitter.com#bryanrwalsh http://twitter.com#alexismadrigal
hasAuthor	http://twitter.com#fischoeler
messageURI	http://zulu:8080/messages/twitter/20221328156
dateOfInformation	Tue Aug 03 09:25:05 EDT 2010
messageType	Tweet

Name	Value
WSJ	0.198518
seachele420	0.164451
donnabrazile	0.055109
bryanrwalsh	0.049274
markseibel	0.035343
USinteriorNews	0.030443
makeitplain	0.025895
nytimes	0.023558
Froomkin	0.019503
WhoDat35	0.017875
HuffPostGreen	0.015846
aristide	0.011206
the_world_news	0.010726
AdvertisingLaw	0.010196

Figure 6. RDS running under the Ozone Widget Framework.

Most of the RDS is installed as a single web archive (WAR file) under Tomcat. Currently, RDS-specific NetKernel modules are installed separately. Users must have accounts under MySQL and Ozone, but there can be multiple instances of these servers available per instance of the RDS. Ozone is deployed as a website, configured with user accounts and widgets. Widgets are configured by their URLs so that each RDS widget could, in theory, be hosted on a different Tomcat server instance. The MySQL host and account login are configured with user preferences using the RDS Preferences widget (see figure 7), so data can be distributed to different MySQL instances.*



Key	Value
gui.look.name	javax.swing.plaf.metal.Me
gui.font.family	Kristen ITC
gui.font.size	12
gui.color.border.default	0x1b122f
gui.color.border.hilite	0xffffff
gui.color.bg.darkest	0x39394C
gui.color.bg.darker	0x636371
gui.color.bg.dark	0x7e7e8C
gui.color.bg.medium	0x878762
gui.color.bg.bright	0xa6a791
gui.color.bg.brighter	0xd9e4be
gui.color.bg.brightest	0xfafaf2
gui.color.text.brightest	0xe1e1C5
gui.color.text.brighter	0xe0e7d5
gui.color.text.bright	0x919274

subject » Person
predicate » sentMessage
object » OSINT

CONSTRUCT
```
?subject <http://cerdec.army.mil/rdecor
?subject <http://www.w3.org/1999/02/2
?object <http://www.w3.org/1999/02/22
```

FROM http://arl.army.mil/alibaba

WHERE
```
?subject <http://cerdec.army.mil/rdecor
?subject <http://www.w3.org/1999/02/2
?object <http://www.w3.org/1999/02/22
```

Figure 7. RDS preferences widget after loading an unusual font (left) and RDS SPARQL query widget (right).

NetKernel and RDS-specific modules are specified with a URL, but this currently is not configurable at runtime. Ultimately, all of these servers used to host the RDS software, and user data can be distributed across available network resources, but gigabit speeds are recommended for all but small datasets. For the most part, conversion of RDS components from DAF to Ozone was straightforward because most of them were already implemented as Java applets. One of the exceptions was the SPARQL query interface, which was originally written in C#. The actual SPARQL query was performed by a Java web service built upon the Jena/ARQ library, so the conversion to Ozone was just a matter of re-implementing the user interface as a Java applet. The resultant RDS SPARQL Query widget is shown in figure 7.

---

*Other databases can be used, such as Microsoft SQL Server, PostgresSQL, Oracle, Apache Derby, or HSQLDB; however, so far, only MySQL and Oracle have been configured.

2.3 Metadata Tagging of Unstructured Data

2.3.1 Rationale

In FY10, one of the main challenges confronting the development of the RDS was the difficulty in obtaining interesting, realistic, and unclassified datasets of considerable size. A fundamental assumption of the RDS project is that its data source would be RDF triples (as mentioned previously, the text extraction problem was being worked by Orbis); however, obtaining a large set of triples with any meaningful content was problematic. It was determined that use of classified data for developmental purposes was impractical. Furthermore, unclassified data that were generated manually for developmental purposes were too limited in size and scope. Efforts to develop analytical tools were hampered without the means to evaluate their effectiveness and gain insight into how to extend the platform. Perhaps most importantly, there was no opportunity to explore the performance bottlenecks of the system when operating on large datasets.

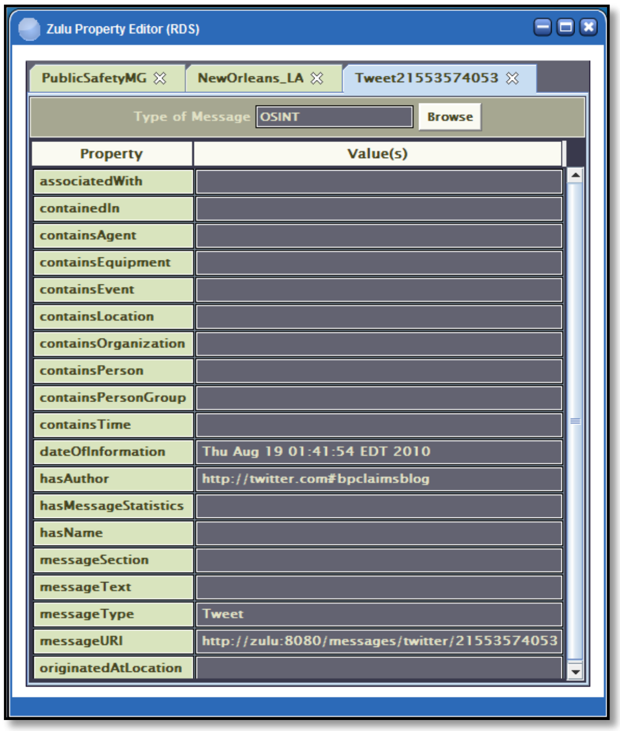
Meanwhile, it was recognized that Twitter was rapidly gaining popularity compared to other social networks and was interesting as a potential data source due to several key factors as follows:

1. Twitter is a message-based communication service that is implemented by a collection of public web services.
2. Twitter web services are centrally hosted and are accessible through several public APIs.
3. Twitter messages, or tweets, are assigned a unique ID and are transmitted with this and other useful metadata. Each tweet is packaged into a serialized data structure in one of several standard formats (e.g., extensible markup language [XML][*] and JSON).
4. Tweets can only originate from users logged in to a Twitter account, so each message is associated with a unique account; accounts are associated with a unique *screen name* and numeric identifier.
5. Tweets frequently contain references to other screen names that are tagged (prefixed with a '@' character), so they are easily extracted from the body of the message.
6. Tweets can be geo-referenced. If someone sends a tweet from a mobile device that is equipped with a GPS, they can optionally include geo-coordinates (latitude and longitude) in the metadata of the message.

It was decided to implement a Twitter client application to tap this resource as a virtually infinite source of unclassified, public-domain information. The largest benefit of this strategy was that a fully automated means of generating RDF triples and mapping them to an existing ontology (i.e.,

[*]For more information, see <http://en.wikipedia.org/wiki/XML>.

the STEF ontology) was not only attainable but affordable. In the process of creating the mapping between Twitter metadata fields and the STEF ontology, this ontology was extended by adding a new *Message* subclass called open source intelligence (*OSINT*). Twitter users were mapped directly to the *person* entity (which is a subclass of 'Agent') since tweeters could be of practically any person type—manual promotion of a person instance to a subclass such as *Terrorist* can be done with the RDS property editor (see figure 8). The STEF ontology was further extended to include new properties; for instance, the Twitter user who sent a message was associated with the message via a new *hasAuthor* property.



Type of Message: OSINT

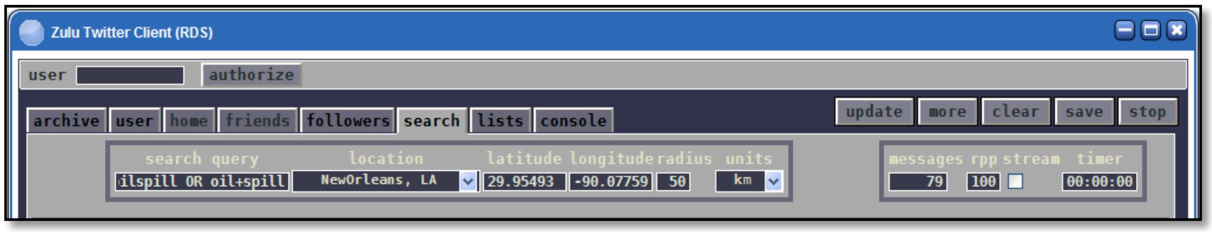
Property	Value(s)
associatedWith	
containedIn	
containsAgent	
containsEquipment	
containsEvent	
containsLocation	
containsOrganization	
containsPerson	
containsPersonGroup	
containsTime	
dateOfInformation	Thu Aug 19 01:41:54 EDT 2010
hasAuthor	http://twitter.com#bpclaimsblog
hasMessageStatistics	
hasName	
messageSection	
messageText	
messageType	Tweet
messageURI	http://zulu:8080/messages/twitter/21553574053
originatedAtLocation	

Figure 8. RDS property editor.

Of particular interest was the Twitter Search API because it has the capability to collect messages that are not only filtered by the terms of the query but could be restricted to messages emanating from specified geographic coordinates and radius. Therefore, the plan was formulated to store tweets as *message* entities uniquely identified by ID and tagged with the date and time that they were posted, message authors, and embedded references to other users as *person* entities uniquely identified by their screen names (or numeric ID). Also, any geo-located tweets could be associated with a *location* entity. This last part presented an interesting entity resolution problem that is described in the following paragraphs as part of the RDS Twitter Client implementation details.

2.3.2 Implementation Details

Development of the RDS Twitter Client (see figure 9) spiraled into one of the richest and most complex components of the RDS. It is composed of dynamic XHTML, i.e., an XML-compliant dialect of HTML. With the exception of a small static web page skeleton, this XHTML document is generated at runtime via JavaScript. There is a Java Applet embedded in the web page, but it is hidden. The Java Applet serves as a proxy for all of the Twitter web service communications. Implementing Twitter API web service calls in JavaScript is difficult due to security measures taken by browsers to protect against cross-domain scripting vulnerabilities. The Java Applet also provides for translation of tweets to RDF for storage in the database and for archiving tweets. All Twitter API calls used by the RDS return JSON format. As stated previously, Twitter API calls return data in at least two flavors, XML and JSON. Although XML is more widely supported, JSON is more useful for applications that use JavaScript, since it is the native object format for this scripting language.



search query: ilspill OR oil+spill
location: NewOrleans, LA
latitude: 29.95493
longitude: -90.07759
radius: 50
units: km
messages: 79
rpp: 100
stream: ☐
timer: 00:00:00

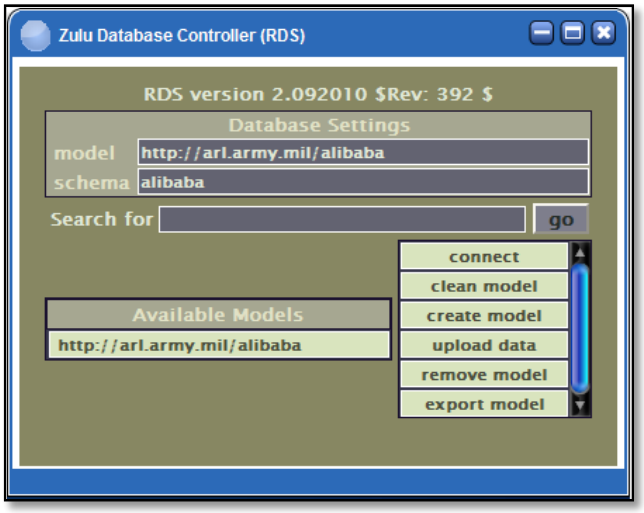
Figure 9. Search tab of RDS Twitter client.

The user interface provides a collection of tabbed panels; the *search* tab is selected in figure 9. There are tabs to retrieve the *home* or *user* timeline, links to *friends* or *followers*, and the timelines of any Twitter *lists* a user has created. So, for instance, if the user wishes to monitor certain Twitter users, or *tweeters*, the user can either create a Twitter account and *follow* the tweeters of interest so that they will appear on the home timeline, or the user can create named lists of them. When requesting API calls requiring user authentication, the RDS Twitter Client uses OAuth (http://oauth.net).[*] OAuth is required for Twitter authentication as of August 2010. It is a secure way for the user to authorize an application to access his or her Twitter account without revealing the password to the application. A user need only authenticate the RDS Twitter Client one time on a given computer. The user's OAuth credentials can safely be stored and are valid indefinitely unless the user revokes them. They are only valid for the particular application that was authorized.

[*]OAuth has been described as "an API access delegation protocol"; for more information, see <http://hueniverse.com/2007/09/explaining-oauth/>.

Each tab overlay is capable of asynchronous operation so the user does not have to wait for a query or update to complete before using other tabs or RDS applications. The search and timeline tabs also have a *stream* option that provides for automatic polling for new search results or status updates so that tweets can be accumulated over time to create larger datasets or provide coverage of a topic for a specific duration of interest. The polling operation self-throttles the frequency of requests depending on how many results are returned so it does not burden the Twitter servers more than necessary while ensuring that no messages are skipped. Twitter has terms of use that, if violated, will automatically *blacklist* applications that do not comply (i.e., frequency of requests must be at least 20 s apart). There are also limits on how many messages can be retrieved per web service call; for instance, searches are limited to 100 messages while timelines are 200, so if polling is too infrequent, messages will be missed. Note that the Twitter Client does not use the *Twitter Streaming API*. This is planned to be implemented in FY11 as a more efficient means of collecting filtered messages over time. This API was not released to the public until late in FY10 and is not yet stable. However, unlike the other Twitter APIs, the Streaming API is not available to any developer. Permission to connect is granted by Twitter based on a written justification, and ARL has not yet submitted an application.

To save a set of tweets to a data model, the user first selects an existing model or creates a new model using the RDS Database Controller (see figure 10), making sure that it has been loaded with the version of the STEF ontology containing Twitter extensions, as just described.



model: http://arl.army.mil/alibaba
schema: alibaba
Available Models: http://arl.army.mil/alibaba

Figure 10. RDS database controller.

The user then hits the *Save* button on the Twitter interface; this function performs a mapping of the interesting Twitter metadata to create new entity instances in the data model and populate those instances with properties (including links to other instances). The most challenging mapping involves the creation of location entities from tweets that are geo-located. When latitude and longitude coordinates are specified in the location field of a tweet's data structure, it is not desirable to create a unique location entity for each set of coordinates. Instead, the RDS maps each coordinate pair to a city name by using a Twitter API method called *reverse geocode*. When different sets of coordinates are mapped to the same city name, entity resolution is accomplished for locations.

The reverse geocode service is very useful, but it is only supported for coordinates in the United States. Different granularities are available (neighborhood, city, and state), but the RDS requests city granularity. Results are returned as JSON data structures with locations (e.g., cities) represented as bounding polygons (currently these are equivalent to rectangles but are generalized to allow for more detailed representations). However, the Web service call latency, on the order of 15–20 s per call, can be prohibitive. It was observed that a collection of 500 or more tweets could take over 1/2 h to process (depending on how many messages were geocoded).

To optimize the reverse geocode mappings, a *quad-tree* caching scheme was implemented in the RDS Twitter Client. Quad trees are data structures that represent two-dimensional (2-D) space as divided into equal-sized quadrants that, in turn, are subdivided as necessary in nested (or recursive) fashion until 2-D coordinates or shapes can be stored for fast retrieval (see figure 11). Whenever a reverse geocode lookup is needed, the quad-tree cache is checked first to see if the specified latitude/longitude coordinates are contained in any of the stored rectangles. If a hit is found, the name of the city is retrieved from the quad-tree cache, and no web service call is required. If it is not found, the reverse geocode web service is used, and the result is added to the cache. After all tweets are translated, the quad-tree cache is saved in the user's RDS-specific application folder. It is serialized in JSON format and written to a file that will be reloaded the next time it is needed. When using this caching implementation, the process of saving a Twitter dataset will typically take a couple minutes, even when starting from an empty cache. A geo-located search will typically be specified with a radius of 50 miles or so, which may only include one or two major cities and a few lesser municipalities. If at some point it is desired to optimize quad-tree usage, a centralized cache could be specified to be shared among users.

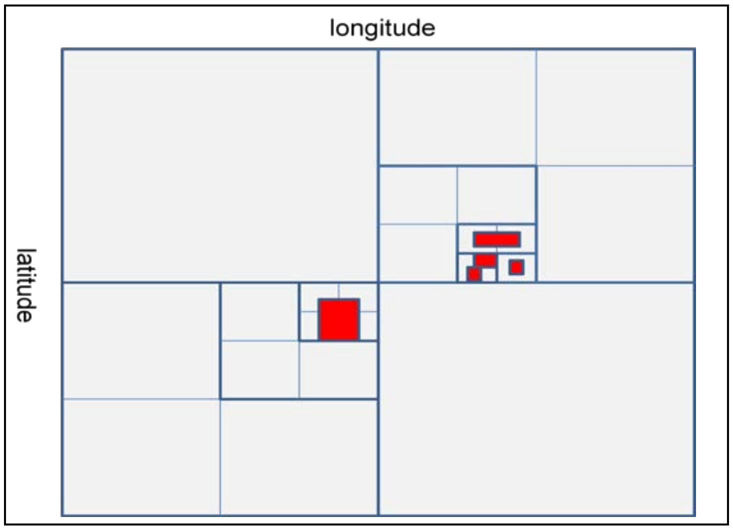


Figure 11. Illustration of quad-tree storage of rectangles.

In addition to populating the data model, all Twitter messages are archived using a NetKernel[*] module. Each message is saved in its entirety (with all metadata) as an individual XML file that contains the embedded JSON data structure (see figure 12).

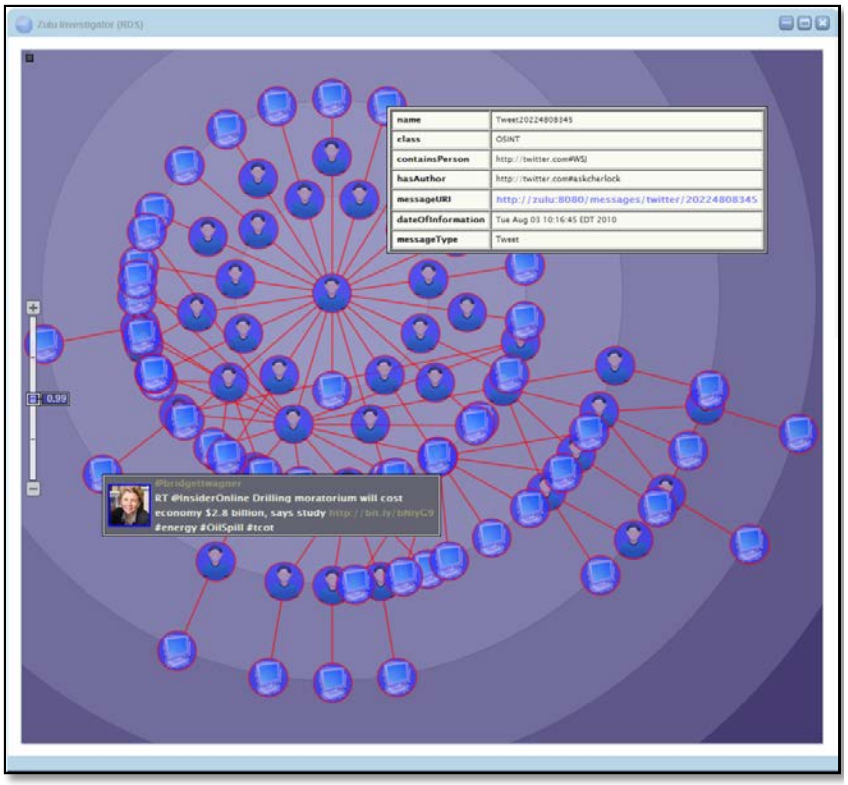
```
<?xml version="1.0" ?>
<twitter-message>
  <![CDATA[
    {
    "location":"Takoma Park, MD",
    "profile_image_url":"http://a1.twimg.com/.../foo.jpg",
    "created_at":"Tue, 16 Feb 2010 03:57:28 +0000",
    "from_user":"GoodTwitty",
    "to_user_id":null,
    "text":"RT @owillis: Mullah Abdul Ghani Baradar: Top Taliban Military
     Commander Captured http://bit.ly/9u6Ekk #p2 #tcot #obama",
    "id":9169738444,
    "from_user_id":33803270,
    "geo":null,
    "iso_language_code":"es",
    "source":"&lt;a href=&quot;http://bit.ly&quot;rel=&quot;
     nofollow&quot;&gt;bit.ly&lt;/a&gt;"
    }
  ]]>
</twitter-message>
```

Figure 12. Archived Twitter message.

The NetKernel module maps the file location to a Uniform Resource Identifier (URI),[†] which is associated with the message entity via the *messageURI* property. When a Twitter message is displayed in the RDS Investigator, the user can click on this URI and another NetKernel module will retrieve the text of the message that is then displayed (see figure 13). The message will contain a hyperlink to the author's Twitter page, as well as to any other Twitter users or articles that are referenced in the message. The Twitter message archive can be searched by using the RDS Twitter Client's *archive* tab. Any message matching the search term will be displayed similarly to the search results previously shown in figure 9.

[*]Developed by 1060 Research (http://www.1060research.com/netkernel); "NetKernel is a software platform that implements the Resource Oriented Computing (ROC) abstraction."

[†]For more information, see <http://en.wikipedia.org/wiki/Uniform_Resouce_Identifier>.



name	Tweet20224808345
class	OSINT
containsPerson	http://twitter.com#WSJ
hasAuthor	http://twitter.com#askcharlock
messageURI	http://zulu:8080/messages/twitter/20224808345
dateOfInformation	Tue Aug 03 10:16:45 EDT 2010
messageType	Tweet

@bridgettwagner
RT @InsiderOnline Drilling moratorium will cost economy $2.8 billion, says study http://bit.ly/bNyG9 #energy #OilSpill #tcot

Figure 13. RDS investigator showing Twitter dataset.

2.3.3 Dual-Use Potential

Although Twitter data were originally sought for developmental purposes, the RDS capability to very quickly conduct graphic analysis on Twitter datasets has sparked interest from a pure, social-network-analysis perspective. Clearly, the RDS could be used for other purposes besides enhancing situation awareness in asymmetric warfare. As of 15 September 2010, Twitter is growing by 370,000 new users per day and has 160 million users.[3] Events of national or international significance, such as natural disasters, economic and technology news, as well as politics, sports, and entertainment can trigger a torrent of tweets. This information could be of use to various organizations tasked with services ranging from emergency response to peace keeping in crowded venues to gauging acceptance of new technology.

2.3.4 Scalability

The RDS uses *betweenness*, a popular network centrality metric for determining an individual's importance in a given social network. A prerequisite calculation for determining *betweenness* is to calculate all shortest paths from each node in a network to every other node. Currently, the RDS uses unweighted links, so the path length of a route from a node *A* to a node *B* is simply a count of the number of hops from node *A* to node *B*. The route finder of the RDS was originally

[3]Wilhelm, A. The Next Web. Live from the Twitter HQ. http://thenextweb.com/socialmedia/2010/09/14/live-from-twitter -hq/ (accessed 3 November 2010).

written as a recursive algorithm using the classic divide and conquer[*] design paradigm. This was very efficient for small graphs of up to a few hundred nodes. However, recursive algorithms make heavy use of the stack, which becomes a significant drain on resources during the calculation, possibly starving the garbage-collected, heap-based memory[†] of the Java virtual machine (JVM) of the necessary storage for the result.

The RDS first calculates an adjacency matrix (A), which is an N-by-N matrix of Boolean values, where N is the number of nodes in the graph. Each value in the matrix (A_{uv}) is initialized to *false* and will be set to *true* during the adjacency matrix calculation if the corresponding source node (N_u) is directly linked to the corresponding target node (N_v). All RDS route finder implementations use this matrix as an optimization in determining connectivity between nodes. Additionally, the RDS stores the product of the route finder as an N-by-N matrix of route vectors (RV), where N is the number of nodes in a graph. Each route vector (RV_{uv}) is a collection of shortest routes from the source node (N_u) to the target node (N_v). These matrices can require significant space for large graphs (over 1000 nodes), especially the RV matrix, since each route vector can contain many routes. After the route finder completes, a connectivity matrix (C) is calculated that is similar to the adjacency matrix, but a cell is *true* if there is either a direct or indirect path between the associated nodes. This connectivity matrix is used to speed up graph layout (arrangement of nodes on the display). Therefore, due primarily to these three matrices, space is at a premium.

Another problem with recursive algorithms is that they tend to limit themselves to a single thread of execution. The RDS recursive route finder calculated a route from a source node to a target node by first looking at all nodes adjacent to the source node. If any of these neighbor nodes are the target node, a solution is found. Otherwise, the algorithm is called recursively on each of these neighbor nodes (each neighbor becoming the source node in the recursion). Since the process of recursively searching for the target node calculates routes between other nodes as a side effect, these sub-routes can be used to fill in the result matrix. The coherency of the graph is exploited, but the algorithm cannot easily be rolled out as a loop (where each iteration is independent) and is therefore a poor candidate for parallel processing.

Subsequently, the route finder was rewritten to be an iterative solution based upon Dijkstra's algorithm.[‡] It was hoped that this algorithm would serve for unweighted edges (by assuming a weight of unity), and later when the RDS supports weighted edges, it would continue to serve well. During testing with the Ali Baba dataset, however, on a graph on the order of 1000 nodes with high adjacency (a high edge-to-node ratio), the route finder became bogged down. The code was then instrumented to show the queue sizes as they changed during the route finder

[*]For more information, see <http://en.wikipedia.org/wiki/Divide_and_conquer_algorithm>.

[†]For more information, see <http://en.wikipedia.org/wiki/Dynamic_memory_allocation>.

[‡]Dijkstra's algorithm is highly regarded for solving the single-source, shortest path problem for graphs with non-negatively weighted edges (http://en.wikipedia.org/wiki/Dijkstra's_algorithm).

execution, and it was apparent that in unweighted graphs with lots of connections, Dijkstra's tree pruning is not effective (many alternate nodes with the identical path cost), so queues become very large. This not only increases the number of iterations necessary in the search, but memory usage could be prohibitive.

Because of this, a breadth-first search algorithm was implemented[4] in RDS to use for computing routes where edges are not weighted. This has been the most efficient algorithm to date. The time required to render a graph of the entire Ali Baba dataset (over 1300 nodes) has been reduced from on the order of 30 min to >1 min.

Now that algorithm speed is scaling well, memory limitations have become the issue. When the route finder, which is running in a Java applet, computes a result matrix for a large graph with high adjacency, the storage exceeds the limits imposed by the Java plugin on 32-bit Windows operating systems. The largest size allowed for the memory heap in the Java browser plugin (the JVM used to run applets) is 1/4 of system memory. However, 32-bit Windows can only address around 4 gigabytes of system memory, which results in the Java plugin setting the maximum heap size to 1 gigabyte. To mitigate this, the Java objects used to store routes in the route finder were augmented to undergo a self-packing process so that storage is minimized. This increased the size of the graph that can be processed but not by a comfortable margin. Perhaps the easiest fix would be to upgrade machines to use 64-bit Windows (64-bit Java is also required), and this is planned for early FY11. This will allow the maximum heap size in the Java plugin to be increased to 1/4 of the physical memory in the machine. A longer-term solution will most likely be to move the route finder to a back-end computing cluster with parallel processing capability so that limits will not be dictated by the computation power and available memory on the user's desktop.

3. Conclusion

This report has detailed work performed under A2SF. The scope of the effort expanded from the original goals of the program, and the modifications to the ARL RDS will provide the means to address the additional goals of cloud computing and threaded execution.

The use of the Twitter API alleviated the need for an external extraction capability to populate the RDS knowledgebase, a fundamental limitation with the RDS. With this limitation removed, the RDS can now be developed and tested with a variety of data content. The new API provides the RDS with data of unlimited sizes, providing more realistic-sized graphs for stress testing performance issues.

[4]Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. *Introduction to Algorithms*; 2nd ed.; MIT Press: Cambridge, MA, 2001; p 531.

The Ozone Widget interface moves the RDS from the DAF to the FEF and Global Graph architecture, which is the future architecture for DCGS-A. The capability to process larger datasets provides better realism from the limited datasets developed under STEF. Use case and proof-of-principle testing shows promise that the RDS can be used successfully to track individuals and events through the knowledgebase to discover unknown relationships. ARL also has a number of datasets. These datasets have differing characteristics and provide the basis for various purposes, such as software development, stress testing, and algorithm validation. Future work includes extending the library of fusion algorithms, continued acquisition and evaluation of databases, focused research in ontologies for rigorous social network analysis metric computation from semantic data, and Global Graph implementation and experimentation.

NO. OF COPIES	ORGANIZATION
1 (PDF only)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC HRR 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB RDRL CIM L 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB RDRL CIM P 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB RDRL D 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
1	DIR USARL RDRL CIM G (BLDG 4600)

NO. OF COPIES	ORGANIZATION
1	DIRECTOR US ARMY RESEARCH LAB RDRL CII B BROOME 2800 POWDER MILL RD ADELPHI MD 20783-1197
2	DIRECTOR US ARMY RESEARCH LAB RDRL CII A L TOKARCIK R WINKLER 2800 POWDER MILL RD ADELPHI MD 20783-1197
2	DIRECTOR US ARMY RESEARCH LAB RDRL CII B M HOLLAND R HOBBS 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB RDRL CII D S YOUNG 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	MITRE RDRL CII D D WALSH 4812 EXECUTIVE DR GAINESVILLE VA 20155
	<u>ABERDEEN PROVING GROUND</u>
2	US ARMY CERDEC I2WD RDER IWP D PORTER SR 6003 COMBAT DR ABERDEEN MD 21005
28 (27 HC 1 CD)	DIR USARL RDRL CII C M THOMAS B BODT E BOWMAN F BRUNDICK J DUMER T HANRATTY C HANSEN E HEILMAN M MITTRICK (15 CPS, 1 CD) G MOSS A NEIDERER J RICHARDSON H ROY