RC23738 (W0503-053) March 9, 2005
Computer Science

IBM Research Report

The Semantic Analysis Workbench (SAW): Towards a Framework for Knowledge Gathering and Synthesis

Anthony Levas, Eric Brown, J. William Murdock, David Ferrucci
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

The Semantic Analysis Workbench (SAW): Towards a Framework for Knowledge Gathering and Synthesis

Anthony Levas, Eric Brown, J. William Murdock, and David Ferrucci
IBM TJ Watson Research Center
PO Box 704
Yorktown Heights, NY 10598, USA
{levas, ewb, murdockj, ferrucci }@us.ibm.com

Keywords: Search and Retrieval, All Source Intelligence

Abstract

We describe a tool called the Semantic Analysis Workbench (SAW) that integrates a broad set of analysis and search functions to support the Unstructured Information Management (UIM) lifecycle. We discuss the SAW and its application to an intelligence scenario and articulate the requirements for an underlying framework that supports Knowledge Gathering and Synthesis (KoGS) tasks.

1. Introduction

Unstructured information is the fastest growing source of current, high value information available. However, extracting relevant content from large volumes of multimodal data and representing it in forms required by search and mining applications that can be used by Intelligence Analysts is a challenging problem. Analysts need tools and techniques to help focus data collections to manageable sets containing relevant content. Lightweight analytics, such as that required by keyword search, may be appropriate for large sets of data. Deep analytics, such as that required for fact and deductive search, need to work on smaller volumes of information to be effective. While incrementally seeking a rich set of facts and justifying evidence, analysts need to prudently apply analysis to unstructured information at different phases of the knowledge gathering and synthesis task.

In this paper we describe a tool called the Semantic Analysis Workbench (SAW). It was built using the UIMA SDK (SDK 2004), a Java implementation of the IBM Unstructured Information Management Architecture (UIMA) (D. Ferrucci and A. Lally 2004 and 2004b). The SAW integrates a set of analysis and search functions to support 1) development of analytics, 2) configuring and running analysis on collections of data, and 3) exploring the results using different search paradigms each requiring incrementally complex analysis; these include keyword and semantic search as well as fact search for viewing entities and their relations. We will discuss the SAW and its application to an intelligence scenario and articulate the requirements for an underlying framework to support knowledge gathering and synthesis tasks.

2. Overview

The success of UIM applications hinges on the availability of a set of tools and methodologies that can be used to rapidly develop components for analysis of artifact and for searching the analysis results. UIMA has an associated framework called the UIMA SDK, which is designed to support the development and deployment of analysis in UIM applications. The SAW application is a graphical tool that supports building, configuring and deploying UIMA analytics, as well as providing a basis for query and delivery of information to the end user. Together these software tools provide a powerful assembly of function for analysis and search in UIM applications.

2.1 UIMA

There are five major UIMA components that will be described here. They are called Collection Reader, Common Analysis System (CAS), Analysis Engine, CAS Consumer and Collection Processing Engine (CPE).

The Collection Reader is the first component in the UIMA processing pipeline, and is responsible for preparing the unstructured artifacts passed to it for use downstream. This may include stripping unwanted header information and removing extraneous tags or characters that are not needed for analysis. It inserts this unstructured artifact into a UIMA data component known as a CAS. The unstructured artifact is referred to as the *Subject of Analysis* (SofA) for that CAS. The CAS travels through the sequence of Analysis Engines, getting analyzed in turn by each Analysis Engine it encounters.

An *Analysis Engine* is the processing component in the pipeline, that processes and adds structured information (in the form of annotations) to each CAS analyzed. It does this by analyzing the SofA, as well as other structured information that has been added to the CAS by analysis engines earlier in the pipeline. Analysis Engines are arranged in a specific sequence to perform the processing required to extract the desired structure and do not retain state from any CAS processed previously.

CAS Consumers can retain state across CASs and produce results over the entire collection. A CAS Consumer

developer is free to build whatever structure they feel is appropriate to aggregate collection-level results. They are often used for processing and/or storing structured information in persistent storage such as a search index, a knowledge base, or a database.

The Collection Processing Engine (CPE) is the container that aggregates all the components into a single processing component that essentially defines the complex processing pipeline. The CPE is defined by a declarative description of the configuration of all the individual elements as well as their respective connectivity. It is important to note that there is declarative information for each of the UIMA components describing required inputs and outputs and many other relevant pieces of information. This declarative information is used to dynamically assemble the required software components into an executable form that is deployed in a UIM application.

UIMA provides a powerful architecture for reusing and combining components to create CPEs that can handle very complex processing. It is designed to be scalable in deployment and has been successfully employed both inside and outside IBM. This section provides a very cursory overview of the major UIMA components. Those interested in a comprehensive description of UIMA are encouraged to see the following publications (D. Ferrucci and A. Lally 2004 and 2004b).

2.2 The SAW

The SAW is an application that integrates a broad set of tools that are used throughout the development process of UIM applications. Figure 1 illustrates the support that the SAW provides for managing this life cycle. There are three orthogonal activities here: 1) Develop Analysis, 2) Configure and Run Analysis and 3) Explore Results.

To support the *Develop Analysis* phase, the SAW launches the Java Eclipse platform (Eclipse 2003) loaded with the UIMA SDK, along with a set of specialized plugins. Developers use this IDE to write and test analytics that are used in UIM applications.

The *Configure and Run Analysis* phase is supported by providing access to the Collection Processing Engine editor tool, a graphical tool for building CPEs. This tool allows analysts to select, parameterize, and sequence components, as well as run the analysis. The CPE is used to analyze a set of unstructured artifacts that have been aggregated into a collection, and as a result produces structured information repositories.

The *Explore Results* phase is supported by a rich set of search capabilities that are used to query the structured information derived from the previous phase. The SAW provides four major search capabilities, each contained in its own tabbed pane in our graphical user interface. They are: *Keyword Search*, *Semantic Search*, *Entity Search* and *Fact Search*.

The *Keyword Search* function uses search engine technology to access artifacts through simple keyword types of queries. These queries are applied against search indexes produced during the analysis of artifacts. The artifacts matching the search query are presented in a hit list, in a manner similar to most search engines available today.

In *Semantic Search*, the words of each document have been annotated with semantic types, or tags. These semantic tags are then the keys that are encoded in the search index, as well as the actual words themselves. Queries can be made against these semantic tags to locate documents of interest. Note that tags can be used by other tags to form more complex semantic structures. The Semantic Search index is queried by using an XML fragment language (Carmel et. al. 2002).

The SAW provides a tool that allows users to graphically compose an XML query fragment by selecting semantic types presented in a list derived from the set of all known types found in the collection. In addition a user can build nested structures using this editor to compose complex queries.

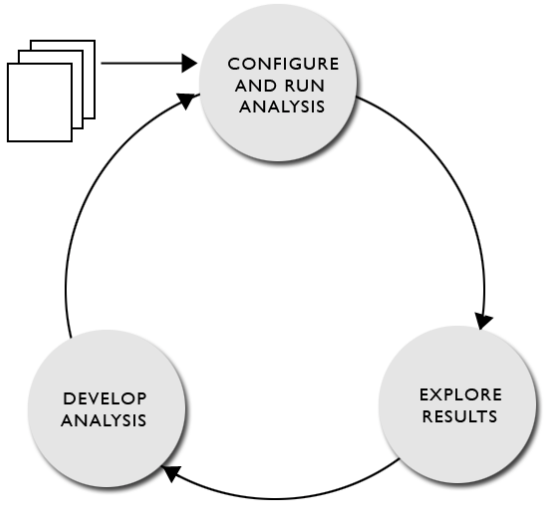

Figure 1. UIM lifecycle supported by SAW

Semantic Search provides a very powerful search capability returning artifacts based on semantic category. For example, this capability can be used to focus or narrow a search e.g. to disambiguate between artifacts referring to the *Person* Kennedy, the *Airport* Kennedy, or the *Facility* Kennedy (hotel). In other situations, it can be used to cast a broad search query, returning all artifacts that contain a particular tag. For example, <*PERSON*> </*PERSON*> would return all documents that include at least one *Person* annotation. We will demonstrate how narrowing and broadening are very effective search techniques for different situations in our example below.

Entity Search and *Fact Search* operate at a different level. Specialized analytics, in this case, have produced annotations at the Referent Layer. This level uses annotations from prior analytics to produce structured information for the *entities* and *relations* discovered in the arti-

facts undergoing analysis. Entities can be thought of as individual things that exist in the data - for example, the specific person John F. Kennedy, or the specific country, the United States. Relations are semantic constructs that relate entities. For example, [*John F. Kennedy* **President** *United States*] encodes the fact that Kennedy was at one time the president of the US.

Coreference analysis relates mentions in a document that have different forms for the same entity. For example, *IBM*, and *International Business Machines* should resolve to the same entity. Pronominal reference can also be resolved. For example, in *"John went home. He took the bus"*, both "John" and "he" refer to the same entity, requiring these facts be associated with the specific entity *John*. In addition, coreference must span documents, enabling facts to be collected for an entity across many documents. During the analysis phase, we extract entities and relations and encode their structured information in a repository called the Extracted Knowledge Data Base (EKDB). The EKDB is a relational database and can be queried using a standard SQL API. The graphical user interface provided by the SAW, however, shields the user from this level and allows queries to be easily created without requiring knowledge of SQL. We will illustrate many of the features described here in the example below.

3. Example Scenario

We will illustrate the various capabilities of the SAW by working through an example information extraction and analysis scenario as if we were an intelligence analyst. For the scenario we will use the document collection from the ARDA sponsored NIMD program (http://www.ic-arda.org/Novel_Intelligence/). The collection includes 250 relatively short documents comprising four types: FBI reports, CIA reports, telephone call intercepts, and nuclear smuggling abstracts from the Center for Nonproliferation Studies. The nuclear smuggling abstracts are real, while the rest are fabricated. We have also added ten locally generated (fabricated) documents that are similar in nature to the FBI and CIA reports, for a total of 260 documents. The fabricated data describes various activities surrounding a number of terrorist plots. No single document describes the entire plot, but by piecing together information across a number of documents, one can uncover the plot in a fair amount of detail. Our task is to uncover the plot.

3.1 First Iteration

We begin the plot with the following information. Recent network chatter has identified an individual and an organization involved in the plot. The chatter has been spotty and noisy, so all we know about the individual is that the person's first name is "Muhammed", and all we know about the organization is that its name contains the word "Al".

Given this information, a typical starting point is to index the available data using a text search engine and then run keyword queries to find relevant documents. Building a search index capable of supporting keyword queries requires minimal text analysis, with tokenization and sentence boundary detection the only mandatory processing.

In the SAW, we invoke the Collection Processing Engine configuration tool (called the CPE GUI) and identify the components required to index the document collection. This includes a Collection Reader capable of reading the document collection, an Analysis Engine that tokenizes and detects sentence boundaries, and a CAS Consumer that extracts tokens and sentence boundaries from the CAS and builds the search index.

After running the configured Collection Processing Engine (CPE), we navigate to the Search tools, select the Keyword Search function, and configure it to run against the index we just built. Given that we are looking for a person with the name "Muhammed", we enter the keyword query "Muhammed", as shown in figure 2. The search engine returns a hit-list and we view the top ranked document with the Document Viewer.

Figure 2. Keyword Search using the SAW

The Document Viewer, as illustrated in figure 3, shows the document with the query terms highlighted. We immediately see that the document matches the query (i.e., it contains the term "Muhammed"), but it does not describe a person named "Muhammed". Rather it mentions the "Muhammed Book Company". Although the document matches our query term, it is not relevant to our intended query.

We view the second document on the hit-list and discover the same situation, so we move on to the next document. After viewing several documents, we still have not found a relevant document, so we decide to

search using the other piece of information we have, namely that the plot involves an organization with the word "Al" in its name.

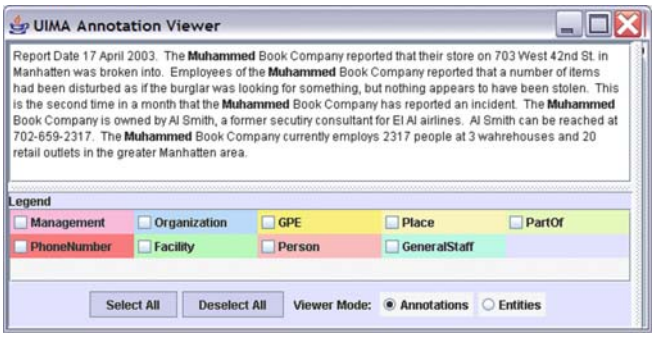

Figure 3. Document viewed with the Annotation Viewer

We enter the keyword query "Al" and view the documents on the hit-list in a similar fashion. The top ranked document matches the query (i.e., it contains the term "Al"), but it does not describe an organization with the word "Al". Rather, the document describes a person named "Al". Viewing several documents on the hit-list again fails to return a relevant document.

In both cases, we are suffering from query term ambiguity—the query terms have multiple meanings in the document collection and the simple keyword search interface gives us no way to distinguish which meaning we want. Often this results in far too many documents for an analyst to view. Our first attempt at analyzing the document collection does not yield an effective search application that allows us to uncover the plot. Using the SAW, we return to the CPE configuration step and see if we can apply more sophisticated text analysis to the problem.

3.2 Second Iteration

In the first query, we are looking for a person. In the second query, we are looking for an organization. Both of these concepts are entities, which can be detected by a Named Entity Recognizer. As such, we decide to incorporate a Named Entity Recognizer into our text analysis pipeline. In addition to tokens and sentences, this will identify semantically meaningful concepts in the documents and allow us to use a *semantic search engine* to search the documents. Using the CPE tool, we add a Named Entity Recognizer to the analysis and select a CAS Consumer that will build a semantic search index.

After processing the data we return to the SAW Search tools and this time select the Semantic Search function. Now, rather than enter simple keywords, we can select semantic concepts from a list of available concepts (generated based on the set of entities and relationships detected by the Named Entity Recognizer) and build a semantic search query. Returning to our first query, we start by selecting the semantic concept *Person* using the Semantic Search panel of the SAW, as shown in figure 4. We then further constrain the query to find a person with the name "Muhammed". In the XML fragment query language supported by the semantic search engine, the query is written as: <*Person*>*Muhammed*</*Person*>. This query yields a much shorter hit-list. We view the top-ranked hit as before and this time we find a document that mentions the person "Muhammed bin Harazi", which not only matches the query term, but also is relevant to our query. Figure 5 shows this document using the Annotation Viewer, in this case highlighting the concept *Person* throughout the document.

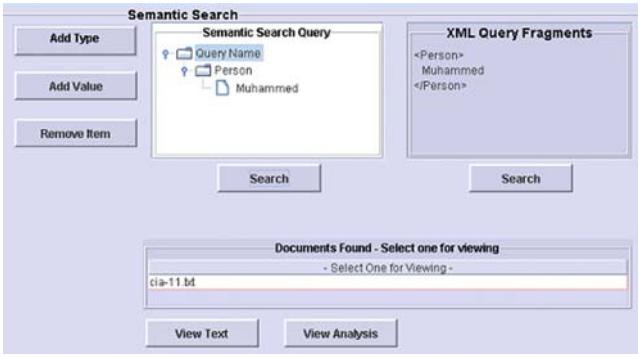

Figure 4. The Semantic Search panel

We then proceed to our second query and build a semantic search query looking for an organization with the word "Al" in it. The query is written as: <*Organization*>*Al*</*Organization*>.

Figure 5. Document with *Person* annotations

Again, this query returns a much shorter hit-list and the top ranked document mentions "Al Qaeda"—a relevant document.

Reading through a few relevant documents from these two queries we start to gather information but still don't see a clear plot. We do discover, however, that some of the documents in our collection describe phone calls made by Muhammed bin Harazi. This suggests that we should perform more detailed analysis on phone calls. Ideally we would automatically identify phone calls described in the document collection and index them as semantically meaningful entities. Using the SAW, we return to the CPE configuration tool and look to add an-

other analysis engine that identifies phone calls. In this case, we don't have a component in our portfolio that satisfies this requirement. We must enter the third phase of the SAW methodology and develop a new analytic.

3.3 Third Iteration

Selecting "Develop Analysis" in the SAW takes us to the analytic development environment where, using the UIMA SDK, we develop two new Analysis Engines: one that identifies phone numbers and one that identifies phone calls, modeled as phone call *relationships* between two phone numbers. With the introduction of relationships into our analysis, we can consider other search and query applications that go beyond keyword and semantic search. In particular, we can construct an Extracted Knowledge Database (or EKDB) that stores relationships between entities and provides a query interface that allows us to search over these relationships, which we call *facts*.

Figure 6. Facts with the *MadePhoneCallTo* relation

The SAW provides a complete environment for developing, testing, and debugging analytics. When we are satisfied that our new analytics are operating correctly, we return to the CPE configuration tool and integrate them into the analysis pipeline. To build the EKDB, we incorporate the appropriate CAS Consumer into the CPE. The EKDB provides a powerful query capability over information assembled and merged over the entire document collection. To fully exploit this, we decide to add additional analysis engines that identify additional named entities and relationships, an analysis engine that reconciles annotations made by multiple analysis engines, and CAS Consumers that perform cross-document co-reference resolution. This last set of components links multiple mentions of the same concept (possibly using different surface forms, called variants) to a single entity and canonical form.

After processing the document collection with our relatively complex CPE, we navigate to the Search Tools, select the Fact Search function, and configure it to point to the EKDB we just built. The Fact Search query interface allows us to search for facts in the database by constraining various aspects of the facts that we seek. In particular, we can specify the domain, the relationship, or the range of the fact, or some combination of the above. For the domain and range, we can specify just the annotation type, or further select a specific instance of that type.

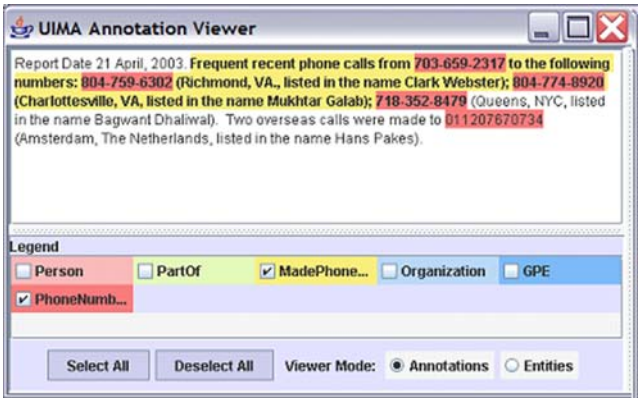

Figure 7. Document with *MadePhoneCallTo* annotations

For example, to see all of the phone calls that the system identified in the document collection, we specify "MadePhoneCallTo" as the desired relationship and leave the domain and range unspecified. This search returns a list of all of the phone calls in the EKDB. Selecting a particular phone call causes the interface to display more details about that fact along with a list of documents that contain that fact (i.e., provide evidence to support the fact) as shown in figure 6. If we view one of the documents in the Document Viewer (figure 7), we see the query fact highlighted in the document along with the participating annotation types and can easily confirm that the phone call fact was correctly extracted from the source document.

Exploring phone calls is a powerful way to connect individuals in the document collection. We can also start with a particular individual and search for all facts that involve that individual. This time, we'll select *Person* as the semantic type for the fact domain, and then further constrain the domain to be the individual "Muhammed bin Harazi". We'll leave the relationship and range unspecified. The resulting fact list immediately shows that Muhammed bin Harazi is involved with the Talihan. Selecting that fact (figure 8) further reveals that the entity Muhammed bin Harazi has other variant forms in the

document collection, including "Abdul Ramazi". If we view the document that supports this relationship, we see that the text explicitly states that Muhammed bin Harazi uses the alias Abdul Ramazi. The aggregate analysis engine that we ran on the document collection was able to detect this relationship and record the link in the EKDB, thus providing a significantly more powerful search capability. By resolving multiple variants to a single canonical form, the analysis allows us to discover important connections between the different variant forms and to query at the conceptual level rather than the lexical level.

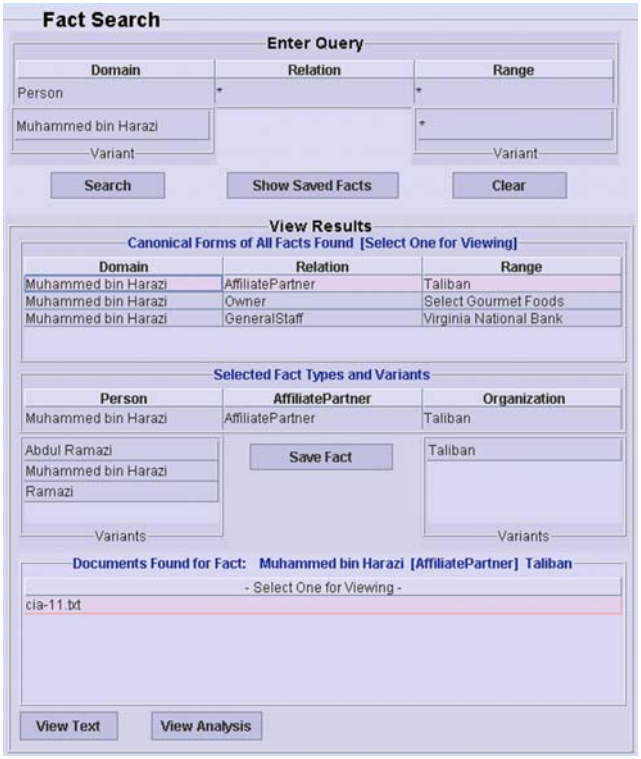

Figure 8. Facts with *Person* [Mohamed bin Harazi]

3.4 Summary

This scenario demonstrates a typical situation faced by an intelligence analyst at the start of an information gathering task. Very little is known about the available data and the information contained within it, leading to a relatively shallow and high-level initial approach. As more information is discovered, the analysis applied to the data must be tailored and refined based on the information need and the characteristics of the data. Deeper, more sophisticated analysis then leads to richer, more powerful query applications.

The SAW naturally supports this iterative, incremental approach to data analysis by integrating tools to easily assemble, configure, and run a CPE, develop new analytics, and explore results with a variety of search and query applications.

4. Conclusion

We have shown that the SAW provides a powerful set of capabilities applicable to the UIM lifecycle. It has proven to be an effective tool for developers of UIM applications as well as for those interested in just viewing the results of analysis. We have used the SAW extensively as a tool for demonstrating UIMA and the capabilities of the underlying analytics as well as a pedagogical tool in many tutorials. The SAW is our first prototype application integrating a broad set of tools that support the UIM lifecycle. This work has motivated us to begin thinking about a framework we call KnOwledge Gathering and Synthesis (KoGS), that bridges the UIMA Layer and the Application Layer above, enabling the rapid development and integration of a wide array of UIM applications.

Our future work in developing the KoGS framework will focus on a few key areas. Management of results of analytics, such as Semantic Search indexes and KnowledgeBases has emerged as a key requirement. Issues, such as, naming; storing; merging new information; removing unwanted information; declaring the type systems, the analytics and the CAS Consumers used in creation; re-useable graphics widgets for visualization, and a host of other concerns need to be addressed to effectively manage analysis results in UIM applications. Management of collections of artifacts is also very important. Similar issues, such as creating artifact collections, naming them, merging them, pruning them, etc. are especially important. Managing the analysis session is also an important requirement. Allowing an analyst to keep and browse a history of his activity, along with fruitful results and provenance information, as well as dead ends encountered can help him in this process, as well as allow sharing this information with colleagues. The three areas above are just a few of the areas that we will begin to explore as we develop the KoGS framework.

References

D. Ferrucci and A. Lally 2004a. Building an example application with the Unstructured Information Management Architecture. *IBM Systems Journal* **43**, No. 3, 455-475.

D. Ferrucci and A. Lally 2004b. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* **10**, No. 3-4, 327-348.

D. Carmel, N. Efraty, G. Landau, Y. Maarek, Y. Mass, 2002 An Extension of the Vector Space Model for Querying XML Documents via XML Fragments, ACM SIGIR'2002 Workshop on XML and IR, Tampere, Finland.

Eclipse 2003, The Eclipse Platform Technical Overview, http://www.alphaworks.ibm.com/tech/uima

SDK 2004: The Unstructured Information Management Architecture. http://www.alphaworks.ibm.com/tech/uima