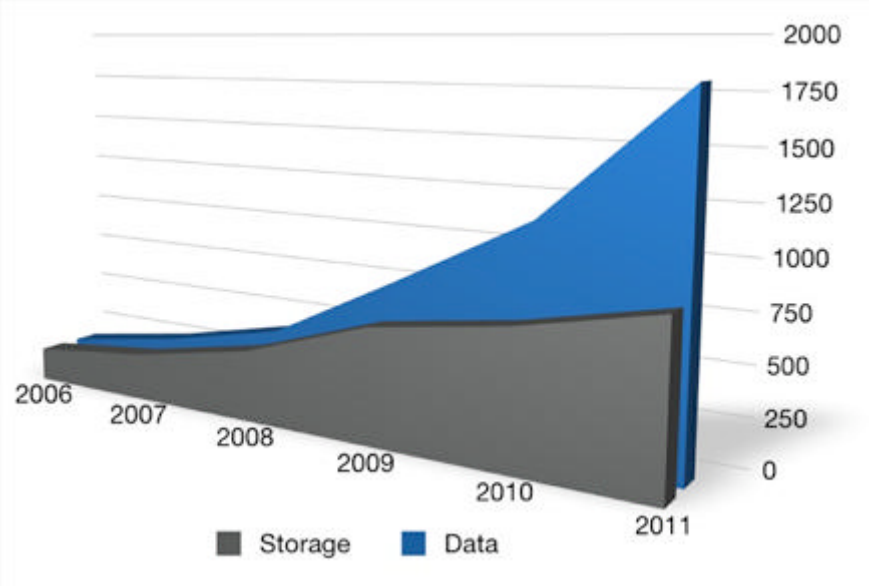
Synthesys™

White Paper

May 2010

Executive Summary

One of the biggest challenges facing the Information Technology market today is how to effectively deal with the explosion of new information, with the vast majority of this information in an unstructured form. It took the Internet from 1995 until 2006 to reach the milestone of 100 million web sites. It took only another two years to reach the 200 million mark[1]. The Digital Universe study, conducted by IDC on behalf of EMC, claimed digital information reached 0.8 Zettabytes – one Zettabyte equals a trillion gigabytes – last year and predicted this number would grow 44 fold to 35 Zettabytes by 2020. This is further evidenced by the approximately 126 million blogs on the internet, and the 90 trillion emails that were sent just in 2009[2]. While the computer industry has developed multiple paradigms for dealing with structured data, the technology for effectively managing unstructured data is still in its infancy.



Global information created and available storage shown in Exabytes (Source: IDC)

Users today are overloaded with too much information; so much, in fact, that it is often very difficult to discern the relevant information amidst the noise. Thus, much of the important information contained within the unstructured data remains untapped and underutilized.

Digital Reasoning Systems, Inc. (DRSI) was founded on the principle that there is order inherent in all languages that can be discovered and mathematically modeled. In other words, we view the problem as being closer to a semiotic (i.e. study of signs and symbols, and their interpretation) problem than a "model-fitting" problem. Fundamentally, DRSI views language as a signal. By developing algorithms that understand these models, the result is software that learns the meaning within the text similar to the manner in which humans do, by examining how terms are used in context.

As text is ingested into our system, it is mathematically encoded words into components of a signal. This facilitates the detection of patterns in the signal more easily by our patented[3] technology.

---

[1] http://news.netcraft.com/archives/web_server_survey.html

[2] http://mail2web.com/blog/2010/01/email-stats-200/

[3] See U.S. Patent Number 7,249,117 - "Knowledge discovery agent system method," granted July 24, 2007

Our technology has its foundations in cognitive science, psychology, philosophy of language, and multiple artificial intelligence (AI) approaches. It does not require pre-modeling of the data, but instead builds a model from the data itself. Our approach is unique because we have a bias against trying to leverage a priori models against the data (i.e. exhaustive extraction or ontology type models). In other words, patterns and relationships emerge from the data itself and do not require specific "knowledge models" to identify them, thus saving considerable time and effort by circumventing the need to create new models for every new genre of data.

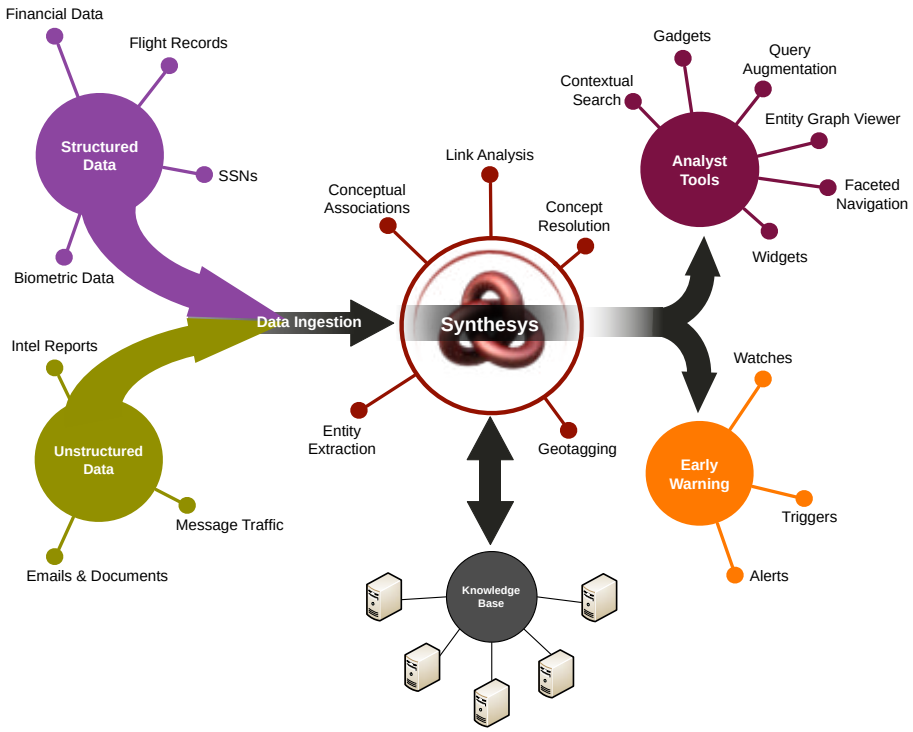
The technology finds relationships between key concepts across very large amounts of unstructured data, giving the system a more human-like quality, learning what words mean based on how they are used, not because they are morphologically related. The value of this approach is the ability to reveal critical hidden information that might otherwise be missed by other analytical techniques.

These analytics, coupled with advanced visualization techniques, allow users to pivot between key concepts and events, effectively summarizing the important information contained in massive amounts of unstructured data. Thus, the important information can be quickly identified, refined, and routed to the appropriate users in a timely fashion.

DRSI's technology is called Synthesys™ and is available as a product in the form of a Software Development Toolkit (SDK), or as a web service (Synthesys™ Server). This enables the power of Synthesys™ to be easily integrated into new applications.

Synthesys™ Functional Overview

Synthesys™ provides a platform for unstructured data analytics, bringing together a diverse set of technologies into a seamless set of services with a rich Application Programming Interface (API). Synthesys™ takes unstructured text as input, uses entity extraction with strong semantic relationship analysis to generate abstracted knowledge objects. These objects (people, places, connections, etc.) can then be used to understand and analyze what's important to you in the data.



As shown above, Synthesys™ ingests structured and unstructured data and performs various analytical functions as it encodes and indexes the data. This “encoded/analyzed” data is designed to be distributed on a massive scale, supporting a variety of analytical tools and alert mechanisms.

The following sections describe the data structure used to store the encoded data and abstract knowledge objects (Knowledge Base), the data ingestion process, and the major types of analytics provided by Synthesys™.

Synthesys™ Knowledge Base (KB)

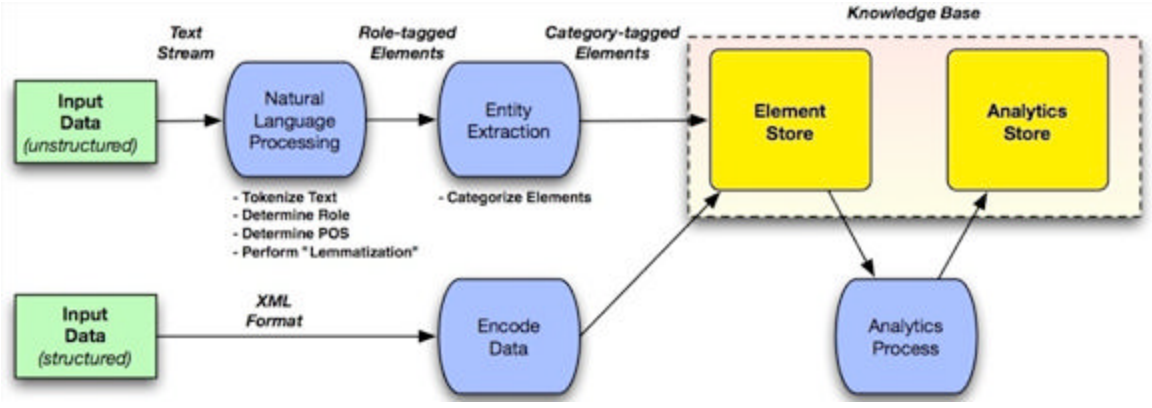
One of the major challenges encountered when creating and managing entity-centric information for large corpora of unstructured data is the inability of most existing database architectures to scale to the large volume of entities and inter-entity relationships. After a great deal of research using traditional tools such as relational database engines and text search libraries like Lucene without achieving the required scale and performance , DRSI has developed an entity storage architecture called a Knowledge Base (KB) that not only performs at scale for both data insertion and querying, but also provides the flexibility of a “pluggable” modular architecture that can accommodate a variety of analytics and Natural Language Processing (NLP) engines. It is implemented in Java, providing machine and operating system independence.

The KB is a combination of persistent storage and intelligent data caching that enables the rapid storage and retrieval of text documents and related meta data. This includes the text content of messages, the categorized individual tokens and semantic token groups (i.e. lemmas) comprising those messages and meta data such as properties, relationships and events. This combination of rich metadata and intelligent indexing supports powerful search and rapid retrieval capabilities, in addition to advanced analytical functions.

Synthesys™ Data Ingestion

All data (structured and unstructured) ingested by Synthesys™ is stored in a KB. The ingestion process is different for structured data vs. unstructured data. Structured data is stored in the KB directly in an encoded form, while unstructured data undergoes more intensive processing.

Synthesys™ accepts two formats of input data, UTF-8 text format for unstructured data and a specified XML format for structured data. There are plenty of readily available transform tools, supporting a wide variety of file formats, all of which can be used with Synthesys™ to transform data into either of these two formats.



Unstructured Data Ingestion

Synthesys™ performs three primary analytics functions as data is ingested and stored in the Knowledge Base. They are the Natural Language Processing (NLP), the entity extraction process, and several advanced analytics processes.

DRSI has done a good deal of research in the area of Natural Language Processing and entity extraction, experimenting with both rule-based and statistical model approaches. We chose a statistical model approach because it provides high-quality feature identification without the need for labor-intensive rule tuning efforts as required by rule-based methods. Statistical models are easier to train and they can easily adapted to a variety of genres of data.

The first step in the unstructured ingestion process is to "tokenize" the input stream of text into terms to be processed by the NLP. A term is typically a word, punctuation mark, or special character. Each term is then examined in order to determine its role (i.e. entity, predicate, number, punctuation, etc). The next step is to assign each term a part of speech (POS), such as proper noun - singular, adjective, adverb, etc. The last step in the NLP process uses the role and POS information to determine if any adjacent terms should be co-joined into a single "element" because they represent the same entity. Common examples might be co-joining "John" and "Smith" into a single element "John Smith", or co-joining titles, such "President of the United States". This process is referred to as "lemmatization" or "chunking" in Synthesys™.

The output of NLP is an "element", which is defined as one or more terms that has an assigned role. Elements are also sometimes referred to as entities. Synthesys™ treats elements as the smallest unit of data that is categorized and processed by the advanced analytics to determine conceptual relationships across the corpus of data.

The next step in the ingestion process is the categorization of the role-tagged elements produced by the NLP. This is called entity (or element) extraction. Each element is examined to determine if it should be classified as belonging to one of the categories defined by the statistical classification process. Synthesys™ comes with a set of defined categories. However, you can also create additional categories by marking up training files with representative examples of the elements of new categories.

At this point, the KB contains the ingested files in an encoded format, each element tagged with role and category information. The element data store contains the tables that define the input messages, the contexts making up each message, and the elements making up each context.

Finally, advanced analytic functions are run on the ingested data to find conceptual relationships and to resolve concepts. Concept resolution is the process of rolling up all the concepts that refer to the same entity into a global representation of that concept. This is explained later in the analytics section.

Structured Data Ingestion

Structured data is ingested using a different mechanism. The structured data must be defined by an XML file (i.e. ingestion of other data base structures not supported) and can take the following data forms:

- concepts (i.e. describes an entity - person, place, or thing)
- properties
- events
- relationships.

A concept describes an entity (i.e. person, place, or thing) through associated properties. For example, if the concept described a person, some of the associated properties might be height, weight, eye color, hair color, bank account #, etc.

Properties are expressed as name : value pairs describing attributes that can be associated with any element or concept defined in the system (e.g. "weight : 215"), whether structured or unstructured. Properties can also be used to augment other analytics processing, such as concept resolution. They can also be used to track data back to its source using a "pedigree" value.

Relationship data consists of the name of the relationship and a list of two or more concepts and/or events that share that relationship. For example, a relationship might be “classmates”, containing a list of people (concepts) that shared this relationship. Concepts can be related to events or other concepts, events to concepts or other events, etc.

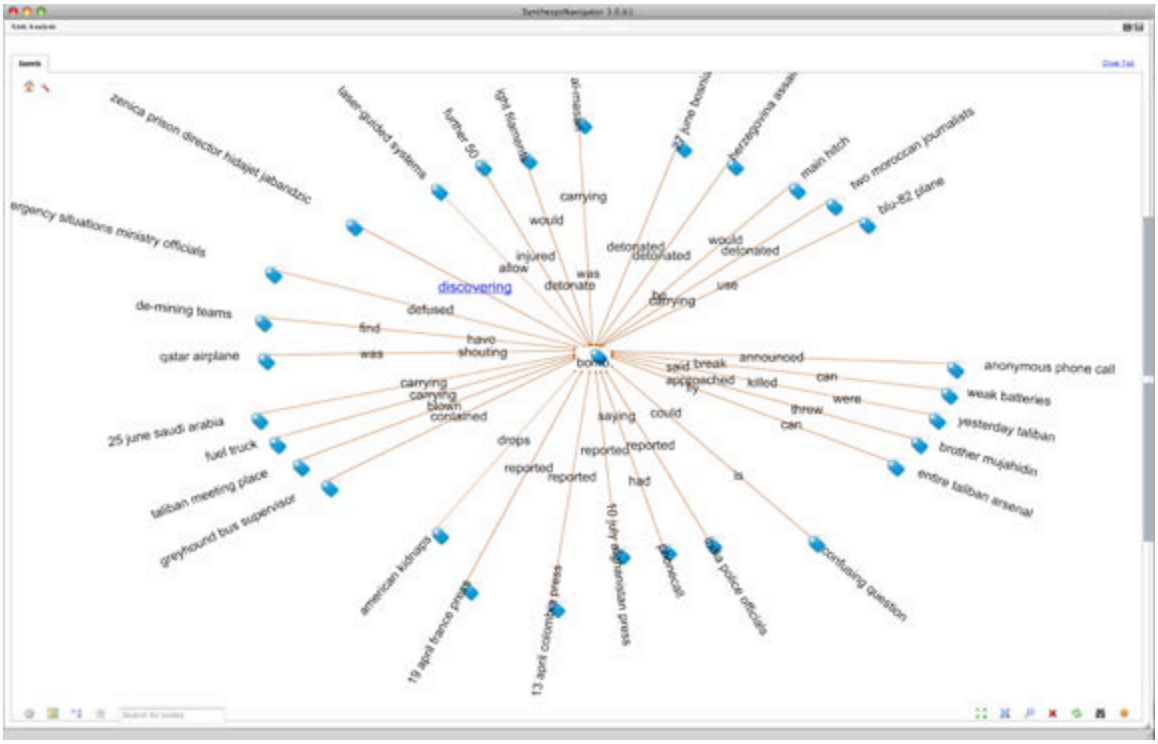
Events are simply activities with a temporal and spatial data component. A variety of methods for specifying time and location are supported in Synthesys™. These include specific time specifications with a start / end value, or a more fuzzy specification, such as “Spring”. Location can be specified by a latitude / longitude specification, or simply a name of a location.

Synthesys™ Analytics Functions

There are four key advanced analytics functions provided by Synthesys™. They are:

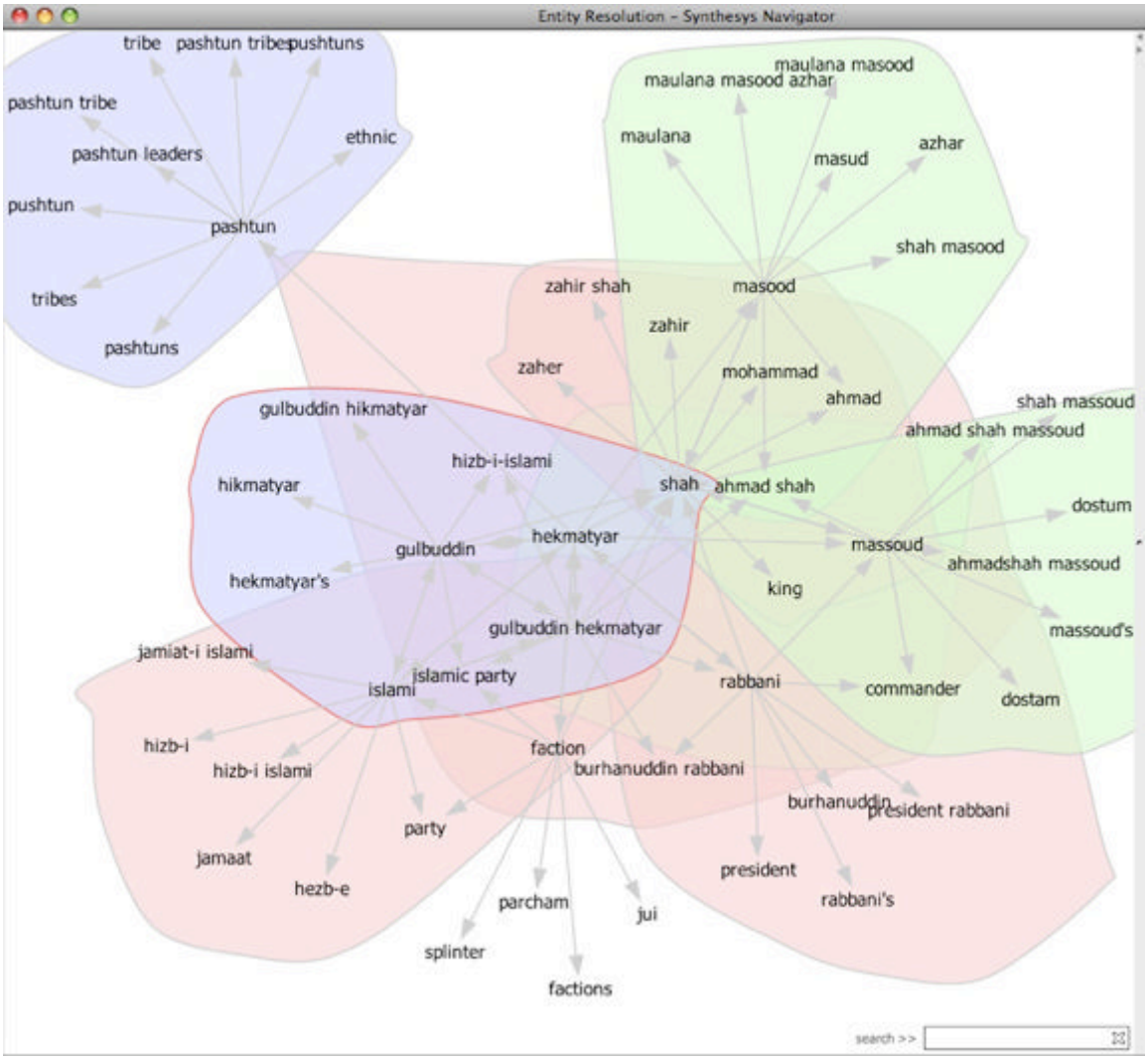
- Link Analysis
- Associative Net (determining conceptual associations between key concepts in the data)
- Concept Resolution
- Geo-reasoning (i.e. tagging named locations with their respective latitude/longitude geo-coordinates).

The Link Analysis function determines which elements are in close proximity to a selected element, and the linkage that ties them together, and generates graphs that represent the type of relationship present in the data. Co-occurrence relationships are extracted from unstructured data and depicted visually in co-occurrence graphs, which show what elements are referred to in the same contexts. The example below shows a link between “zenica prison director hidajet jabandzic” and “bomb” of “discovering”. The context containing this link is “on zenica prison director hidajet jabandzic *discovering* a bomb planted under his car”.



Conceptual associations (i.e. Associative Net) entail treating the text as a signal, with the impact of each element's interaction with every other element being mathematically measured, consisting in part of intelligent dimensionality-reduction on a contextual interaction (or co-occurrence) distribution of elements, resulting in a compact representation of each element's interaction (or signature) with every other element.

Using advanced statistical tests, insignificant or incidental relationships are pruned out, creating a high recall/high precision associative capability. This function can also be used to identify all the various spelling variations of a person's name, or associating a person's name to a nickname used to refer to the same person in a different context. The following shows how this capability can be used in rolling up similar and highly related names – in this case variations and related entities to the Afghan insurgent "Gulbuddin Hekmatyar":



Another interesting aspect of this function is the ability to specify a person, and find other people not directly related to the specified person, but that share similar characteristics. For example, specifying the head of state of a country might return heads of state for other countries.

Concept Resolution involves finding and rolling up all the elements in both structured and unstructured data that define the same entity into a "virtual global entity". It can also be described as a maximally-determined set of properties across a collection of structured and unstructured

elements. The concept resolution process leverages information from the Associative Net (i.e. conceptual associations) process, categorization, property values, and relationships to make this determination.

The ability to resolve concepts is very important to any analysis process, since it could skew the results if a set of different entities defining the same concept are treated as separate entities. An example of a specific use case that illustrates the usefulness of this function is name vetting. The accurate analysis of a person's activities depends on the ability to identify all references for that specific person as the same "entity", regardless of how that person is referred to in the text (i.e. different name spelling, use of nicknames, etc). It also insures that all the information related to a specific entity in any of the structured tables are correctly correlated.

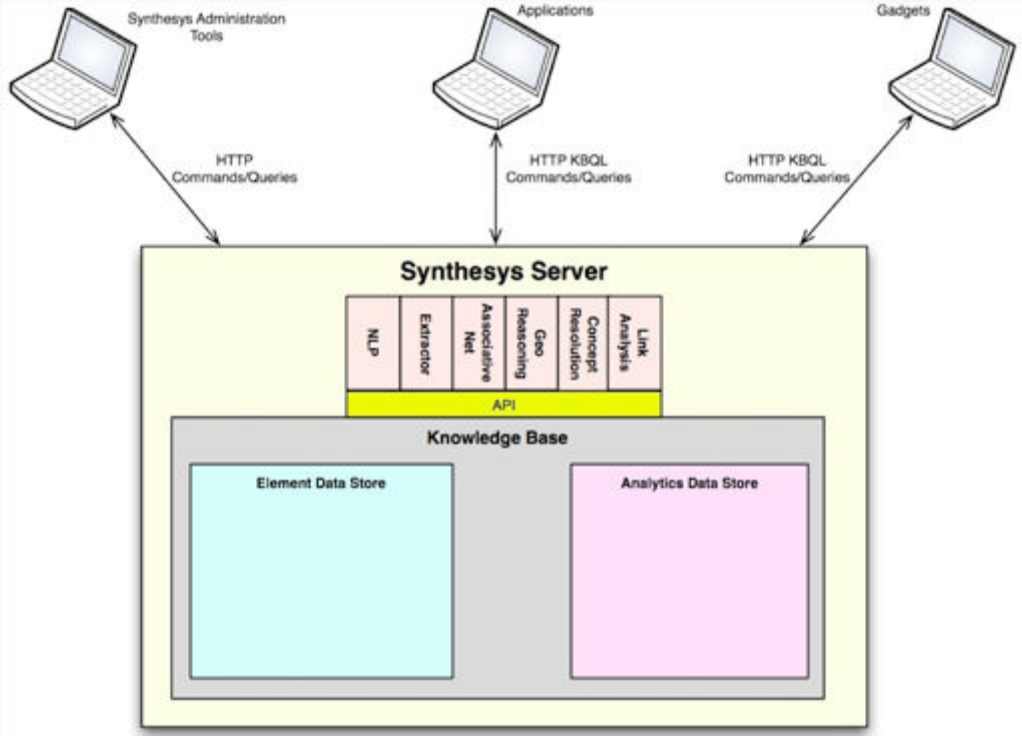
There is also an optional "geo-reasoning" function that directs Synthesys™ to tag named locations found in the text with their respective geo-coordinates (i.e. latitude and longitude). These values are stored in the data as properties belonging to the element representing the named location. This process is performed during the unstructured data ingestion processing.

Synthesys™ Integration Tools

As mentioned in the executive summary, Synthesys™ is a true SDK, providing two integration options. You can integrate Synthesys™ directly into your program through a set of Synthesys™ JAR files and a Java API, or instantiate Synthesys™ as a web server.

While the SDK option is fully supported, DRSI strongly recommends the Synthesys™ Server deployment option because it offers the greatest amount of flexibility, scalability, and performance for large enterprise environments, as well as a set of convenient administrative tools.

A simple Graphical User Interface (GUI) - based installation tool is provided to install Synthesys™ Server. In addition, other tools are available to administer instances of Synthesys™ Server, creating and managing Knowledge Bases, and a category training tool that provides an easy mechanism to mark up example files for use by Synthesys™ to create its statistical entity extraction models.



Synthesys™ Server is designed as a modular (or plug-in) architecture, providing the flexibility to specify alternate analytics tools for certain functions such as NLP, entity extraction, conceptual relationships, etc, using the server administration tool.

The Synthesys™ Server API consists of two primary URL forms. They include a set of Knowledge Base queries (KnowledgeBase Query Language (KBQL)), and module commands that provide access to many of the modules that comprise Synthesys. These URL forms all have a similar format structure, shown below:

`http://host:port/synthesysserver/query` - KBQL query requests

`http://host:post/synthesysserver/modules/`*`module-name?query-strings`* - module command requests

KBQL is a query format developed by DIgital Reasoning Systems, based upon the MQL specification which is published as a part of the Freebase project to serve as a JSON-based query language for the Freebase database schema. For web developers using JavaScript, JSON is trivially transformed into JavaScript objects so it is particularly convenient for work in which a browser-based user interface is involved. Because the Freebase project's MQL usage isn't mapped out as a formal language, there is no set schema to be designed against.

Standard KBQL requests may include operations for the standard CRUD (Create, Read, Update, Delete) operations. Currently, the KBQL query engine supports create and read query types for the most common object types, with limited support for update queries for specific object types. Support for specific delete query types is planned for future implementations.

Since the KBQL commands are built on a JSON (JavaScript Object Notation) based query language, it is more intuitive to use JSON objects to specify the input parameters for this command form. The input parameters are placed into a JSON object, and then passed to the server in the request body. The result is passed back to the client in a JSON object with the same format as the one passed in the request body.

JSON syntax is used for specifying KBQL queries because it is fairly intuitive and there are JSON libraries available in most every programming language. JSON is similar to XML in that they are both “self-describing” (i.e. the values are named, and thus human-readable), hierarchical in nature in that you can describe values within other values, and have parsing support built into many different languages. A JSON object can consist of name/value pairs, arrays, and other objects.

KBQL queries are executed by passing the text of the query in a JSON object as “post data” with the URL specifications. These queries are based on the assumption that you specify the known fields of a data structure to act as a template for the result set you are seeking, and leave placeholders for the fields that you would like information returned. For example, to query for a specific element in the system, you could ask for it by it id value as follows:

```
{
        "knowledge_base" : "oef",
        "type"           : “element”,
        "id"             : “906238099457”
}
```

The results of this query, however, would not return any new information, other than validating that the specified element exists in Element Index. This is because KBQL queries work under the the principle that you specify what you know, leave placeholders for the data that you want returned, and the system "fills in" the template you've constructed with the results from the query engine. For example, running the same query but leaving placeholders for the "text" and "role" fields of the element will return the "text" and "role" fields are filled in:

```
{
        "knowledge_base" : "oef",
        "type"           : "element",
        "id"             : "906238099457",
        "role"           : null,   /* "null" is the placeholder for primitive
                                      data types such as strings and numbers*/
        "text"           : null
}
```

will return the following:

```
{
        "knowledge_base" : "oef",
        "type"           : "element",
        "id"             : "906238099457",
        "role"           : "ENTITY",
        "text"           : "foo"
}
```

Synthesys™ Server also provides cursor support for KBQL queries that might return large amounts of information. Cursors allow a user to fetch large result sets from the server in pages. The pages are fetched sequentially, and after all of them have been returned, the cursor is closed and the results discarded. This feature is especially important for insuring acceptable performance in wide area network (WAN) enterprise environments

Examples of the types of queries supported by KBQL include search functions for specified elements, categorized terms, contexts, properties, events, and messages, along with faceted navigation support for filtering these search results. Also included are a variety of relationship, event, and concept queries, as well as Link Analysis queries that return graphical results in a graphML format easily rendered by a variety of visualization tools. The flexibility afforded by KBQL allows programmers a wide latitude in how to integrate the analytics capabilities provided by Synthesys™ into their applications.

Synthesys™ Server is specifically designed to meet a variety of processing requirements, with configurations ranging from small, single server deployments capable of supporting millions of documents, all the way up to a large distributed, clustered environment capable of supporting billions of documents, using commodity hardware to scale to any size data or concurrent user requirements.

Summary

Both the public and private sectors are awash in information today, the majority of which is unstructured. The underlying problem is not a lack of data, but rather the inability to understand the data already collected. Agencies, organizations and businesses simply can't connect the dots that exist across their data. Without better tools, there is a very real danger that non-obvious, critical concepts and relationships will be missed. Failure to discover and understand these facts and relationships across the large and complex information sources can have tragic consequences.

DRSI believes that Synthesys™ provides a flexible, scalable solution to this problem, providing real-time analysis of structured and unstructured data.

About Us

Founded in 2000, Digital Reasoning® Systems is a privately held company based in Franklin, Tennessee. Born out of the belief that some day all software would learn, the company spent the first five years of its existence conducting extensive research into the nature and types of machine-learning systems, their inherent limitations and their potential application to difficult problems such as natural language processing and unstructured data analytics (UDA).

It was during the initial research phase that Digital Reasoning Systems founder Tim Estes created the concepts behind Digital Reasoning, which is the company's patented breakthrough in machine-learning and natural language processing.

Digital Reasoning Systems provides UDA solutions that enable organizations to reduce costs as well as increase revenue and customer satisfaction by mining and analyzing the over 80% of data that is unstructured and currently untapped.

Digital Reasoning Systems is a rapidly growing firm that is intent on becoming the leader in the UDA market with best-of-breed solutions and unwavering customer focus.

For more information, contact us at:

Digital Reasoning Systems
730 Cool Springs Blvd.
Suite 110
Franklin, TN 37067
615.370.1860 p
615.370.1865 f

sales@digitalreasoning.com

www.digitalreasoning.com

Synthesys™ White Paper

Published by Digital Reasoning Systems, 730 Cool Springs Blvd., Suite 110, Franklin, TN 37067

Published in the United States of America.

Marketing Department
Digital Reasoning Systems, Inc.
730 Cool Springs Blvd., Suite 110
Franklin, TN, 37067.

Email: marketing@digitalreasoning.com