Partnering Enhanced-NLP with Semantic Analysis In Support of Information Extraction

Hisham Assal
CAD Research Center
Cal Poly State University
San Luis Obispo, CA
hassal@cadrc.calpoly.edu

John Seng
Computer Science
Cal Poly State University
San Luis Obispo, CA
jseng@calpoly.edu

Franz Kurfess
Computer Science
Cal Poly State University
San Luis Obispo, CA
fkurfess@calpoly.edu

Emily Schwarz
Computer Science
Cal Poly State University
San Luis Obispo, CA
eschwarz@calpoly.edu

Kym Pohl
CDM Technologies, Inc.
San Luis Obispo, CA
kpohl@cdmtech.com

Abstract

Information extraction using *Natural Language Processing* (NLP) tools focuses on extracting explicitly stated information from textual material. This includes *Named Entity Recognition* (NER), which produces entities and some of the relationships that may exist among them. Intelligent analysis requires examining the entities in the context of the entire document. While some of the relationships among the recognized entities may be preserved during extraction, the overall context of a document may not be preserved. In order to perform intelligent analysis on the extracted information, we provide an ontology, which describes the domain of the extracted information, in addition to rules that govern the classification and interpretation of added elements. The ontology is at the core of an interactive system that assists analysts with the collection, extraction, organization, analysis and retrieval of information, with the topic of "terrorism financing" as a case study. User interaction provides valuable assistance in assigning meaning to extracted information. The system is designed as a set of tools to provide the user with the flexibility and power to ensure accurate inference. This case study demonstrates the information extraction features as well as the inference power that is supported by the ontology.

Categories and Subject Descriptors I.2.7 [**Artificial Intelligence**]: Natural Language Processing – Language parsing and understanding, text analysis.

Keywords *Ontology; Natural Language Processing; NLP; Information Extraction; Modeling; OWL; Intelligent Analysis; Semantic Analysis.*

1. Introduction

For a long time, access to relevant documents was a critical impediment for people trying to obtain information on a topic of interest for them. Paper-based documents require the availability of a physical instance of a document, involving the transport of documents with the corresponding costs, delays, and risk factors. Computers and networking infrastructure provide nearly instantaneous access to a huge repository of documents via the World Wide Web and search engines offer support in locating documents that are likely to contain relevant information. The task of examining these documents, extracting relevant pieces of information, and assembling them into a coherent framework, however, still requires significant human involvement, leading to the "information overload" bottleneck. This paper describes the approach our group pursued in the creation of a system that supports humans whose task it is to collect information about a particular domain, analyze that information, assemble it into a coherent framework, and possibly draw conclusions and recommend actions based on knowledge derived from the collected information.

In our context, *information extraction* (IE) refers to the use of computational methods to identify relevant pieces of information in documents generated for human use, and convert this information into a representation suitable for computer-based storage and processing [Wimalasuriya & Dou, 2010]. IE is often implicitly constrained to text-based documents, although in principle it can be applied to other types such as images, videos or audio recordings as well.

Examining text documents for the occurrence of words is very straightforward for computers, and search engines like Google demonstrate the success of this approach. For IE, however, the goal is to identify meaningful chunks of information, which requires the selection of relevant pieces of texts (words or phrases), and the conversion into a computer-suitable format. Since natural language is ambiguous, redundant, and contextual, the task of identifying and extracting relevant pieces of information is very challenging for computers.

Natural language processing (NLP) refers to the use of computation methods to analyze and process spoken or written statements in a language commonly used by humans. Such methods are applied from different angles. At the *syntactic level*, grammatical rules are used to determine the basic building blocks of text, such as sentences, words, and the roles they play in a given piece of text. At the *semantic level*, the meaning of words, phrases, sentences and documents is determined. At the *pragmatic level*, the context is taken into account as well to determine the most suitable interpretation. Syntactic analysis is relatively straightforward from a computational perspective, but not sufficient to determine the meaning of a text fragment or document; ambiguity, for example, can drastically change the information conveyed in a sentence. Semantic analysis relies on a common interpretation between the creator (writer) and the consumer (reader) of a document. One approach to establish a common interpretation relies on ontologies as frameworks that define the core terminology in a domain and specify the relationships between words. Contextual aspects can be explicitly specified (e.g. through rules), incorporated into a domain-specific ontology, or derived from additional information about the documents and how they are used. Statistical approaches in NLP can overcome this interpretation problem to some degree, and are sometimes combined with the structural analysis methods that rely on rules specifying the grammar of the language.

For humans, speaking the same language is the basis for such a shared interpretation. They use the same principles to construct sentences, although they may not explicitly know the rules of the underlying grammar. For effective communication, they should have a shared vocabulary, meaning that there is a set of words for which the conversation partners have an interpretation, ideally, the same or at least compatible interpretations for one particular word. Contextual information, such as the current location, task, domain of interest, or emotional state of the conversation partner, is considered in cases where multiple interpretations are possible. During the act of reading or listening, humans automatically analyze and interpret language blocks, resolving ambiguities and try to arrive at a coherent interpretation of a document or conversation. Such a holistic treatment of natural language is very challenging for computers. Current systems often combine rule-based approaches, ontologies, and statistical approaches with reasonable success for situations where vocabulary, language structure, or context is constrained. Participants in the Loebner Prize contest, a refinement of the Turing Test, can engage in conversations on limited topics in such a way that it is difficult for an observer to determine if a conversation partner is human or computer [Loebner Prize, 2010].

Our system aims at the extraction of meaningful pieces of information from wide sets of documents, their integration into a coherent framework, and the derivation of new knowledge. One critical assumption is the existence of such a coherent framework for the domain under consideration. An ontology is a formalized representation of such a framework, and serves multiple purposes in our context. First, it makes explicit the knowledge of humans about the domain. Second, it ensures that the interpretation of critical terms is consistent within the group or organization that utilizes it. Third, it spells out important relationships between those terms. Associated with an ontology can be a set of axioms, which capture the very basic, generally accepted statements about a domain. The flexible use of relations in ontologies allows dynamic, multiple classification of entities. While these properties of ontologies already allow for fairly powerful reasoning, our system also incorporates components for reasoning that are external to the ontology.

On the NLP side, ontologies are the vehicle to provide a semantic framework for the interpretation of sentences and documents, enabling the conversion of statements available in natural language into a representation suitable for computers. For the IE task, an ontology helps in deciding which pieces of information may be relevant, and how they are incorporated into the already existing knowledge repository. The combination of axioms and a flexible hierarchical structure provides a strong basis for reasoning and analysis of the information captured in the repository. Ontologies have a natural visual representation as a graph where nodes represent concepts and links relationships between concepts, and thus serve as a powerful information retrieval method by following interesting relationships.

Ontologies provide major support for several aspects of our system, such as the explicit representation of domain knowledge, interpretation of text, the analysis of documents, and the identification and retrieval of stored information. However, they are difficult and cumbersome to built, may not be available for some areas of interest, and do not capture the full understanding that humans have. The use of ontologies can also become computationally very expensive.

Our overall system is structured as a community of collaborative agents, which share a common objective, but have their own capabilities and the autonomy to make their own decisions. Each agent offers a set of services, and in turn may rely on services provided by other agents.

While we experimented with different domains, our main testing ground is the extraction of information about terrorism financing from publicly available documents on the Web, with news agencies, newspapers, various organizations, and individuals as sources. Clearly the analysis, interpretation, and integration of extracted information is very critical here, and justifies the use of significant human and computational resources. Extracted information can be extremely valuable, and augmenting the capabilities of human analysts may enable them to find overlooked "nuggets" of information, or to draw conclusions they would not have come to otherwise. Obviously the choice of this domain also has some drawbacks: Many relevant documents are not publicly available, and understanding the environment and context in which such analysts work is limited by security constraints [Flynn et al., 2010]. Nevertheless, we believe that the domain of terrorism financing is suitable for the demonstration of our system, and later we will present a usage scenario from this domain.

Computer-based IE and NLP methods have become "good enough" to assist humans with the processing of large quantities of text-based documents. They can identify the occurrence of particular text pieces (words, phrases) in documents, allowing the analysis of simple statements about entities, events, and actions, and the comparison of identified entities against other documents [OpenCalais, 2010].

The performance of these methods, however, in general is not good enough to leave the task of information extraction completely to computers, especially in fields where the combination of different pieces of information, possibly from a wide range of sub-fields or related areas, is critical. Thus the goal of our project is to demonstrate that computer-based methods for information retrieval, combined with natural language processing and ontologies, enable analysts to deal with significantly larger sets of documents and obtain better results than they would with conventional methods. Our system provides a reusable, configurable platform that can be adapted to specific domains and tasks with moderate effort. It is intended for interactive and collaborative use, and relies on configurable components for document search and retrieval (e.g. Web crawlers), domain-specific language aspects (e.g. rules to identify entities in the domain such as names or specific terms), visual presentation of entities and their relationships (e.g. as topic/concept maps, or ontology visualizations), observation of activities and intermediate results during the extraction and analysis process, and traceability of results to the source documents.

2. Ontology-based Information Extraction and Tools

This section covers the various implementation differences in previous ontology-related information extraction systems. The survey by [Wimalasuriya & Dou, 2010] provides an excellent overview of variations among ontology-based information extraction systems. We describe the difference in implementations along four dimensions as categorized by [Wimalasuriya & Dou, 2010]: information extraction implementation, ontology usage, ontology extraction specificity, and natural language data source.

The first and probably most significant variation in implementation is how the information extraction is performed in a system. Information extraction itself is a developing field and can be performed using a combination of techniques.

The first information extraction technique is to use regular expressions to match phrases in natural language text. These regular expressions are often constructed by a domain expert to perform matches on phrases as they appear in actual text. This approach is tedious, but can often yield high quality results. Another information extraction technique is that of gazetteer list. A gazetteer list is a list of known terms and phrases as they exactly appear in text. When text contains a named entity that matches an element in the gazetteer list, then the named entity is extracted. A third approach is to use a machine learning classifier to classify natural language text as relevant information. This approach uses training data (commonly human annotated text) to train a machine learning classifier to learn how information appears in sentences based on some feature set (e.g. part of speech tags, word position, or capitalization).

The second of four implementation differences is how the ontology is used in the system. Some systems use the ontology as user input which a human has pre-defined. This assumes all extracted information items must fit into some portion of the defined ontology. Another approach is to have the system dynamically define the ontology as it processes the natural language input. Such a system would create new objects in the ontology as they are identified at run-time.

The third implementation difference is what portion of an ontology a system can extract. An ontology information extraction system can potentially extract classes, properties of classes, and relationships between classes. Ontology-based information extraction systems can vary on the level of details for a class that is extracted.

The final variation among information extraction systems is in the source of the natural language data that is processed by the systems. Some systems will use a text source available from a Web site, while other systems require the text to exist in a particular file format.

Research in the information extraction field has been motivated in the past by two research competitions. In the past, the Message Understanding Conference [MUC] was a DARPA sponsored event held from 1991-1997. The event required participants to demonstrate systems performing various information extraction tasks. The Automatic Content Evaluation [ACE] program was a NIST sponsored event held from 1999-2008.

In terms of the tools that are available, there are several toolkits that target the development of natural language processing applications [LingPipe, 2010; OpenNLP, 2010; GATE, 2010; UIMA, 2010; ClearTK, 2010]. Two commonly utilized frameworks are GATE (General Architecture for Text Engineering) and UIMA (Unstructured Information Management Architecture). These frameworks provide services and workflows which simplify the construction of NLP applications. For our work we utilize the UIMA architecture. This framework provides facilities such as annotations, chaining of text-level annotators, and an overall modular development environment.

3. General Approach

The objective of this project is to build a platform for extracting a large volume of information from multiple sources on the Internet and provide a context, within which intelligent analysis can be performed. To achieve this goal, we combine existing NLP tools together using the UIMA framework and partner the enhanced version of NLP with a domain ontology which describes the complex relationships that exist in that domain and some of the basic rules of inference, which describe the necessary conditions for classification of objects and relationships. We emphasize the role of the user in this approach to provide assistance in interpreting the extracted information, review the automated inference and make any changes to enhance the quality of information.

The architecture of this system is service-oriented (SOA), as shown in Figure 1. All the functional components are implemented as Web services and hosted on an application server. The decision to architect the information extraction capability in a service-oriented manner was twofold. First, the extraction of information appeared to be an activity that would be invoked by external components as a part of some larger business process and done by a wide variety of users and on a potentially repetitive basis. As such, it appeared appropriate to deploy the information extraction capability as a whole in the form of a discoverable and subsequently invocable service. Second, since the information extraction capability is comprised of a number of sub capabilities each of which may be a candidate for replacement with newer, more capable components, it seemed advantageous to carry this service-oriented concept into the internal architecture as well. It was anticipated that doing this would produce a test bed-like environment where internal capabilities were substantially decoupled and primed for potential replacement or introduction of additional, complimentary capabilities.

The current service implementation includes the following:

- Web crawler service to access sources on the Internet and re-

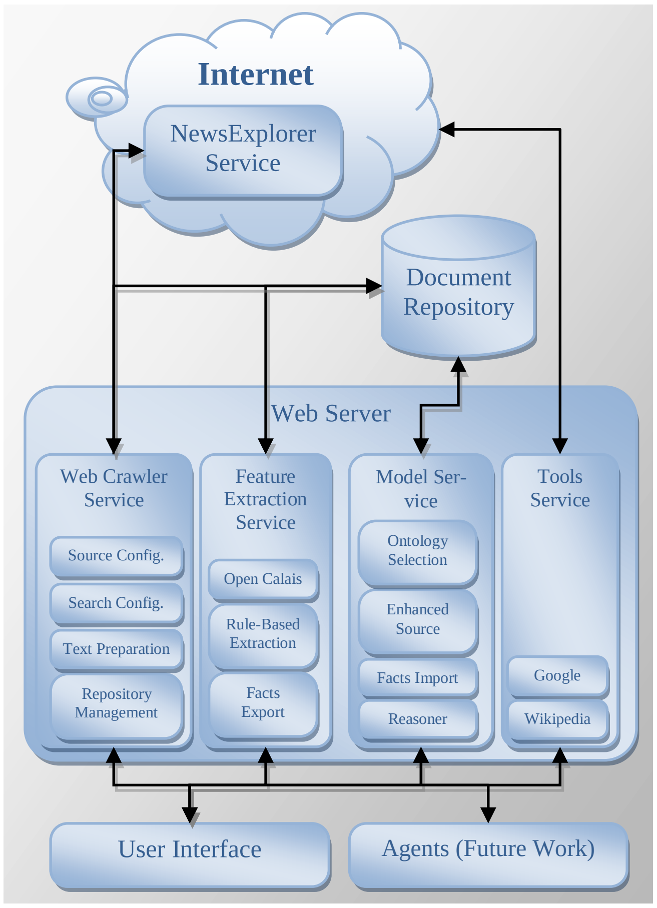


Figure 1. General Architecture

trieve articles and other material from Web sites, clean it up (e.g. remove HTML tags) and store it in the document repository.

- Feature extraction service to process the textual information, extract objects of interest, as defined by the ontology, and classify some of the extracted objects.
- Model service to provide ontology access and manipulation functionality. The Web Ontology Language (OWL) [OWL 2009] is selected to build the context model. It supports dynamic, multiple classification, which is important in order to describe a person, for example, as both a 'banker' and a 'terrorist supporter'. It also supports decidable logic, which makes the inference possible. The model service utilizes a reasoner, which is an embedded inference engine. For persistence of model objects we use an RDF [RDF/XML 2004] triple store.
- Tools service to provide additional tools to help the user interpret the extracted information and check the validity of the automated inference. The tools allow for more complex extractions, such as multi-term NER. Tools also help refine and aid in the classification of extracted objects, using encyclopedic services (e.g. Wikipedia).

In addition to the Web services, there is a document repository, which holds the raw textual information as well as the results of NLP extractions. The repository is accessible to all the services, as well as the user interface.

The basic concept of the user interface design is to allow the user to be involved at any step of the process. It is important to have the user input and integrate it into the intermediate results.

The user interface provides tools for the user to access the services, configure them to perform according to the task at hand and provide input when the user decides to intervene.

The UI is Web-based, so that it can be run from any location and support collaboration. The UI offers support for configuration of information sources, setting of various selection criteria, user-initiated extraction of basic (i.e., explicit) concepts, user-initiated exporting of NLP extractions into semantic model. The user can also edit the system configuration elements, such as the extraction rules and classification lists.

The reasoning process is triggered by the user and can be configured easily on the UI. The user can select an ontology for reasoning and can load multiple ontologies to support complex inference.

The presentation of the inference results can be done in a tabular form as well as on a graph, depicting the concepts and their relationships. The results are editable and in the final design can be integrated into the on-going inference process.

The UI also offers access to a set of services that provide tools to enhance the interpretation of the extracted information.

4. Use Case Overview

We present a use case here to illustrate the benefits of enhancing the traditional NLP capabilities with ontology-based inference capability. The use case supports intelligence analysis of terrorist activities based on information extracted from news sources as well as intelligence reports. The assumption here is that different pieces of information appear in different documents from multiple sources using related but not identical terms, and that their combination yields an interesting or important new piece of information. Information may be obtained from a number of sources, with overlapping data items, e.g. a person or an organization may be mentioned in multiple news articles. All the statements are stored in the repository, including any redundant information. When statements are imported into the ontology as facts, a check on information is performed against the current state of the ontology. Information about the same object (person, organization, location, etc.) is merged together to ensure information integrity.

The use case examines information about people, organizations, and meetings. The relationships among these three types of objects are also examined, e.g. person's membership in organization, meeting between two people, etc.

The scenario for the use case is as follows:

Purpose:

- To identify the possibility that a suspected terrorist group (or individual) is planning an attack

Explicitly extracted information:

- Membership in a known (or suspected) terrorist group
- A person employed by a financial institution
- A meeting, taking place between the members of the terrorist group and the banker

Inference:

- The terrorist group is planning to transfer a large amount of money through the banker
- A terrorist plot may be in the making

This simple use case illustrates the extraction of information from multiple sources (the person's membership in the terrorist group may come from one source, while the meeting between this person and the banker may come from another). It also illustrates the use of an ontology to build a context for the extracted information and the use of the ontology inference capability to create new information.

5. NLP Environment

This section covers the implementation and design of the natural language processing server portion of this project. The NLP environment implements the Web crawler service and the feature extraction service, as shown in Figure 1. The NLP server architecture consists of a combination of very general external Web resources as well as domain-specific features. By utilizing both external Web resources and features tailored to a unique domain, we are able to extract natural language features which may be missed by a more broad-based approach while maintaining an acceptable level of generality.

From a high-level perspective, it is the job of the NLP server to download news articles from the Internet and extract relevant information from key phrases present in the articles. The extracted phrases, which we call *assertions*, are then sent to the semantic model. The NLP server acts directly with the GUI application and therefore, the user can directly send requests to the NLP server via SOAP Web service calls from the user application.

The following subsections outline the server structure. The next subsection describes what criteria are used to acquire news articles for processing. Subsequent subsections cover: the task of cleaning the HTML for a Web article, how natural language features are extracted from our system, and the network implementation of the NLP server along with a description of the exposed API.

Article Acquisition

In a high-level view of our system, the NLP portion takes as input natural language text and produces assertions regarding the relevant domain. Our goal is that the natural language input may come from a variety of sources: Examples of online sources include: major news Web sites, online forums, and blogs. In addition, static documents may be utilized as well. These may include various technical reports. In our current implementation, the source of natural language content in our system is news articles gathered from the Internet. News articles are written for readers who may not have extensive background knowledge on a topic and therefore the articles will often list people's full name and titles. This is advantageous for detecting the political and corporate positions of various people.

We use an online news article collection Web site NewsExplorer [NewsExplorer, 2010] developed by the European Commission's Joint Research Centre. The Web site performs statistical analysis of various news articles that are released throughout the day. Some sample features of the NewsExplorer Web site are the ability to cluster articles by common terms, country of origin, and primary article topic.

For our work, we utilize a subset of the topics that NewsExplorer supports. Our graphical application currently allows downloading of articles that are categorized under one of the following topics: Security, TerroristAttack, Conflict, Energy, and AlternativeEnergy.

Article Cleanup

The NewsExplorer Web site provides Web links for each of the articles that it tracks. In order to download an article, the article itself must be cleaned of any HTML tags and content not relevant to the news article. For this task, we use the HTML cleaning functionality of the AlchemyAPI NLP Web service [AlchemyAPI, 2010]. Given a Web link of an article, this service returns the raw text of the news article void of any HTML markup tags or any other advertising content. Once an article has been cleaned, a local copy is cached for future use.

Natural Language Feature Extraction

Our system utilizes two methods of extracting information from natural language text: OpenCalais and our own rule-based technique.

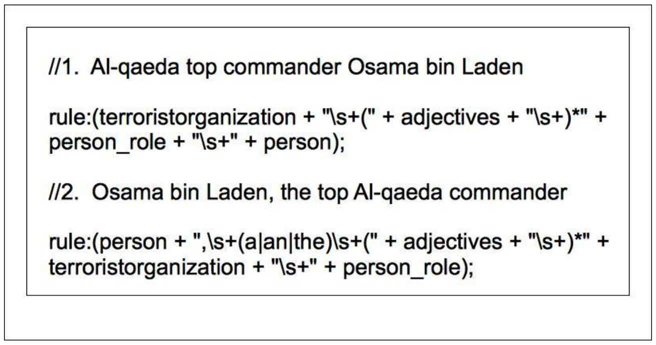
OpenCalais [OpenCalais, 2010] is an information extraction service provided by Thomson Reuters. The service takes natural language text and returns a document with various labeled entities in an RDF format.

For our current implementation, we use only a subset of the entities that are detected by OpenCalais. The entities that are stored are `PersonCareer`, `PersonCommunication`, `PersonRelation`, and `FinancialInstitution`.

`PersonCareer` is a relationship that states a person's full name along with an institution that the person is affiliated with. The institution may be a company, political organization, or other institution. This `PersonCareer` relationship does not necessarily specify an employee relationship.

`PersonCommunication` relations exist between two or more persons. The communication can be an actual communication event, visit, or meeting. The status of a `PersonCommunication` can be either announced or planned. An announced communication occurs when the communication occurred or allegedly occurred. A planned communication is one that will happen in the future.

In addition to the entities that OpenCalais detects, we implement a rule-based detection system, which implements matching based on regular expressions. As an example application of our case study, we focus on detecting the names of terrorists and their membership with particular terrorist organizations. Figure 2 shows some sample extraction patterns that we look for in order to associate a particular person with a terrorist organization. Our regular expressions patterns are based on a list of known terrorist organizations. We allow for minor spelling differences in the names of the organizations and assume if the spelling is within a threshold of character differences then the names are a match.



```
//1.  Al-qaeda top commander Osama bin Laden

rule:(terroristorganization + "\s+(" + adjectives + "\s+)*" +
person_role + "\s+" + person);

//2.  Osama bin Laden, the top Al-qaeda commander

rule:(person + ",\s+(a|an|the)\s+(" + adjectives + "\s+)*" +
terroristorganization + "\s+" + person_role);
```

Figure 2. Sample Extraction Rules

The figure shows two example rules. Before each rule, a sample phrase that matches against that rule is listed. In each rule, there are keywords that are used to match types of words in a phrase. For example, the keyword `terroristorganization` matches with a list of known terrorist organizations that is read from a configuration file. The keyword `adjectives` represents a list of known common adjectives that are used in the context of terrorist members and leaders. The `person` keyword refers to the name of a person. We use the detection of persons from OpenCalais. `Person_role` represents various roles or positions that a person can take (e.g. member, leader, director). Other standard regular expression symbols are used for matching whitespace, articles, and punctuation.

As the assertions are extracted, they are inserted into an intermediate model using Jena [Jena, 2010]. This Jena model maintains occurrence frequency for each assertion as well as the pertinent information for the assertion (e.g. terrorist organization name, person name, or communication status). Once the assertions are stored in an intermediate model, they are then transferred to the semantic model Web service. This information is stored in an RDF file that is explained in the semantic model section of the paper.

We ran across some limitations of the OpenCalais Web service. OpenCalais does perform co-reference resolution with pronouns, but has difficulty when the reference does not utilize a personal pronoun but instead using a plain noun to refer to an object. For example, a paragraph may refer to 'the president' several times without refer to the person's actual name. In this case, OpenCalais will have difficulty in making the connection with the particular reference to the president's actual name.

Another limitation of OpenCalais is that the relationships it extracts are limited to certain pre-defined relationships that exist in the RDF schema. Although the schema is quite extensive, there are certain instances where it is more powerful to implement a domain-specific relationship extraction technique - like our rule based technique. In our approach, we used a rule-based system to extract information regarding terrorists. This is a quite domain specific extraction and could not be extracted by OpenCalais.

NLP Web service Implementation

The NLP Web service is implemented as a SOAP service that the end-user application can invoke. There are two primary components of the NLP service: a thin wrapper layer and the backend server module. This section describes both service modules.

The thin wrapper layer is a Tomcat Web service that provides a Web interface to the user application. This layer provides an API which allows applications to perform various NLP related and file system related tasks. This layer communicates with the backend NLP server over a socket-layer protocol.

Backend NLP Server Implementation

The backend NLP server module is the component of the NLP server which performs all of the primary work. It is threaded to handle requests from multiple requests.

There are two primary service calls which the backend server handles: `BatchDownload` and `RunBatch`. `BatchDownload` is an API call which invokes the backend server to download a number of articles.

`RunBatch` calls the OpenCalais Web service for each of the documents that was downloaded using `BatchDownload`. OpenCalais returns an RDF file containing the extracted relationships. This RDF file is cached locally for future use and the assertions within a particular RDF file are compiled into a statistics list of assertions for a given batch.

6. Semantic Environment

To extend the solution into the realm of semantic analysis, the enhanced NLP environment described above is mated to the second component of this two-part approach, the semantic environment. This additional environment consists of a Web Ontology Language (OWL)-based context model that is managed within a Jena-based platform equipped with an embedded reasoner. The resulting model and inference capability is collectively exposed as an invocable Web service.

The following section presents the representational paradigm selected to represent the respective domain ontology together with applicable inference logic. Within this discussion, several key modeling facilities are highlighted and related to the information extraction problem.

The OWL Representational Paradigm

At the heart of the semantic platform is the context model, or ontology, which contains the concepts necessary to describe and reason about the extracted information fragments produced by the enhanced NLP capability. Particular attention was given to the selection of the appropriate representational paradigm. This paradigm must provide a somewhat relaxed view of classification along with native support for embedding inference logic within the very concepts that it semantically relates to. Following investigation into several more traditional paradigms, we decided upon OWL as the representational language and execution platform suitable for this research.

The Web Ontology Language, or OWL as it is more commonly known, is a semantic markup language. The primary purpose of OWL is to facilitate the publishing and sharing of ontologies across the World Wide Web (WWW). OWL is intended to be used by software applications that need to process Web-based content in a meaningful manner. That is, OWL-based content is designed to be machine-interpretable.

A typical OWL environment consists of several key components that support both model development as well as subsequent execution. A typical OWL execution platform consists of a triple-store where model content is persisted, a reasoner capable of inferring additional concepts based on embedded logic, and a query engine used to ask questions regarding model content. Together, these components form a cohesive platform for the development and execution of semantic content.

Perhaps the most important component of any OWL environment is the reasoner and as such warrants a more detailed description of its tasks. As the name implies, the main function of this component is to essentially *reason* about a given OWL model and its associated content. More specifically, an OWL reasoner processes class definitions, individuals, and embedded rules in an effort to accomplish two primary objectives. The first of these tasks is to identify any logical inconsistencies existing within the model definition and its use. As the platform produced by this research supports user-driven importation of ontologies applicable to the target analysis domain (e.g., intelligence, logistics, command and control, etc.), this feature can be used to verify decidability and logical consistency of such models. The second task performed by the reasoner is to identify any additional knowledge that can be automatically inferred based on the logic embedded within the model definition in conjunction with associated content. This additional knowledge can include subsumption and association relationships and the adjustment of classification of various scenario content. Although only beginning to emerge within the timeframe of this research, some reasoners have the ability to not only infer additional content, but to retract previously inferred content whose truth can no longer be established (i.e., truth maintenance) [BigOWLIM, 2010]. This is a crucial feature for any practical application of automated reasoning as establishing what is no longer true is as important as the initial inference that produced it.

Above and beyond the components comprising its typical platform, the OWL representational paradigm supports several very powerful concepts that are thoroughly exploited by the information extraction process. These concepts provide the flexibility to represent as of yet unclassifiable extractions as well as to repeatedly adjust the definitions of those that can be classified.

Multiple Classification

Multiple classification is the ability for something to be simultaneously defined under two or more classifications. This is a very powerful capability and has significant implications on the manner in which representational models are developed. Unlike traditional, more rigid modeling paradigms where inheritance must be employed as a means for the specialization of concepts, OWL modelers enjoy a much more flexible environment without concern for relating classifications in order to support a single entity exhibiting features defined across multiple classifications. Once qualification rules are embedded within class definitions, the management of exactly which classifications are appropriate at any given time can be conveniently offloaded onto the OWL reasoner.

Dynamic Classification

Dynamic classification is the ability for the classification of something to be adjusted throughout time. In contrast to the traditional approach of re-instantiating an entity under a new classification, dynamic classification offers the means to preserve referential integrity by maintaining the existence of the original entity by only changing its type information. This capability goes hand-in-hand with multiple classification in creating a dynamic environment where extracted facts can effectively mutate throughout their lifecycle as additional knowledge is encountered. Like management of multiple classification, determining exactly what classification(s) an OWL object qualifies for at any point in time is typically the responsibility of the OWL reasoner.

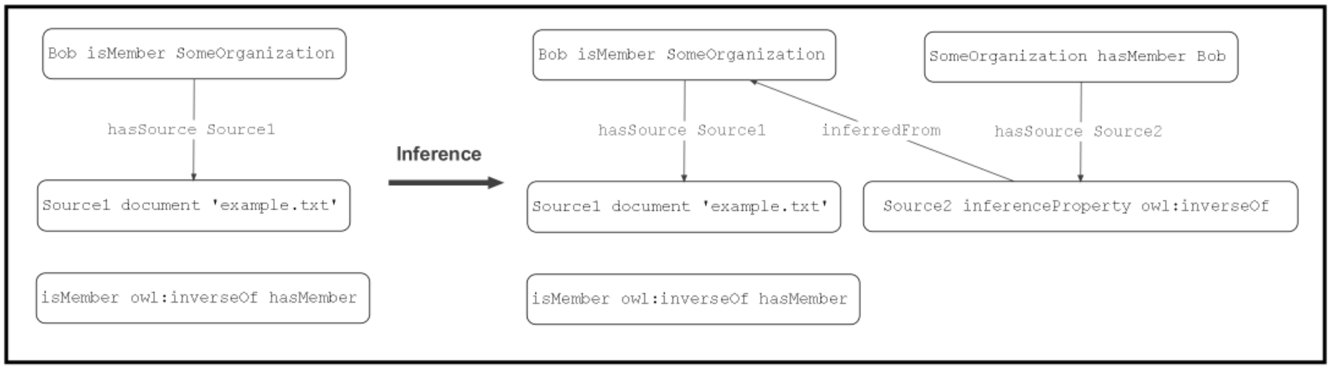

Bob isMember SomeOrganization
hasSource Source1
Source1 document 'example.txt'
isMember owl:inverseOf hasMember
Inference
Bob isMember SomeOrganization
SomeOrganization hasMember Bob
hasSource Source1
inferredFrom
hasSource Source2
Source1 document 'example.txt'
Source2 inferenceProperty owl:inverseOf
isMember owl:inverseOf hasMember

Figure 3. Sample Inference

Open World Assumption (OWA)

Traditional database systems operate under a set of assumptions in order to enable the query engine to return meaningful responses. These suppositions include the *closed world assumption*, the *unique name assumption* and the *domain closure assumption*.

The closed world assumption states that if a statement cannot be proven true, given the information in the database, then it must be false. The unique name assumption states that two distinct constants designate two different objects within the given universe. The domain closure assumption states that there are no other objects in the universe than those designated by constants within the database.

These assumptions were reasonable in a world where a database represented all the information available about a given domain and no external information sources were needed to perform the functions of any database application. Since this time, however, the Internet has become a major source of information with the effectiveness of many applications being based on access to external information from sources that may be unknown at the time of application design. This has required a different kind of knowledge representation, capable of dealing with the *openness* of the Internet. The *open world assumption* was adopted to allow for the relaxation of the constraints imposed by the closed world assumption. Along with the open world assumption, the other two assumptions were consequently also relaxed.

Under an open world assumption, all things are possible unless asserted otherwise. This is in stark contrast to traditional decision-support paradigms where the volume and extent of considerations is limited to what is explicitly asserted to be true about the world at any given time. Although operating under an open world assumption has implications on model development, it is primarily model usage and interpretation that is affected since logic operating across such a model must refrain from assuming too much.

Unconstrained Composition of Features

Being able to describe the characteristics of something in a manner unbound by available definitions can be very helpful when dealing with evolving knowledge, such as in the information extraction problem domain. With environments equipped with such flexibility, information extractions can be characterized and continuously re-characterized in a manner unconstrained by current classification(s) or known concepts. Working in conjunction with dynamic classification, as something's characteristics evolve or otherwise change, so may its alignment with available definitions. As described earlier, determining exactly what changes in classification are appropriate is typically the responsibility of the reasoner.

The complementary partnership between this set of representational concepts is one of the most powerful aspects of the OWL paradigm. Collectively, these mechanisms support a progressive and fluid understanding of extracted information fragments within a dynamic and constantly evolving context.

Semantic Model Service

Our implementation of the semantic environment is the semantic model service (as referred to as the "model service" in this section). This SOAP Web service contains the ontology-based semantic processing for the system. From a high level perspective it is the role of the model service to take in assertions from the NLP extraction and derive new knowledge via inference over those assertions in combination with a domain specific OWL ontology. To accomplish this task the model service controls the importation and persistence of NLP extractions and an OWL ontology into a triple store. The model service is then able to execute reasoning over the triple store and query its contents.

In order to produce new facts which can be traced to their original source the rules used for reasoning must be enhanced beyond those typical for OWL inference. The structure of the facts within the semantic repository also needs to be modified to contain source information. In the next section we will discuss the added reasoning and source references as implemented in our data model and in the following sections discuss the implementation of the model service itself.

Source Enhancements for Reasoning

For a real world scenario such as our central use case seeks to represent, the reasoning and expressiveness of OWL is stretched. Our dilemma is similar to that expressed by [Vrandecic et al., 2006]: a bare fact stored as a triple cannot express information about its source, reliability, or how it was derived. This information about the *context* of a fact is as important as the fact itself in our solution.

To express this context each fact within our semantic repository is annotated with a source object. This object can be of several types and contain varying properties. A source object for facts imported from the NLP extraction is of type `NLPAssertionInfo` and contains information such as the source document and the place in the document where the extraction occurred. This annotation could be applied through the use of named triples, named graphs, or RDF reification. In our case we use RDF reification, since it can be used with any type of triple store.

However, our needs go beyond simply annotating facts. New facts that are inferred will need source data as well. A new fact's source is the inference itself. That inference as well as the facts used in the inference need to be made available to the user. Because of this we have taken the rules for typical OWL inference and extended them to create proper sources for inferred facts with type `SemanticAssertionInfo`. Figure 3 demonstrates an asserted fact with source, an OWL restriction, and the result of inference over that data. Note that depending on the level of traceability needed `Source2` could also have a property linking it to the OWL restriction as well.

This technique radically increases the size and complexity of rules. We have yet to discover its suitability for large datasets. It is possible that systems that deal with reification naively with named triples will allow for better scalability. Future work on the model service includes documenting more specific needs for our triple store and rule engine regarding performance requirements and facilities for advanced reasoning.

Service Structure and Implementation

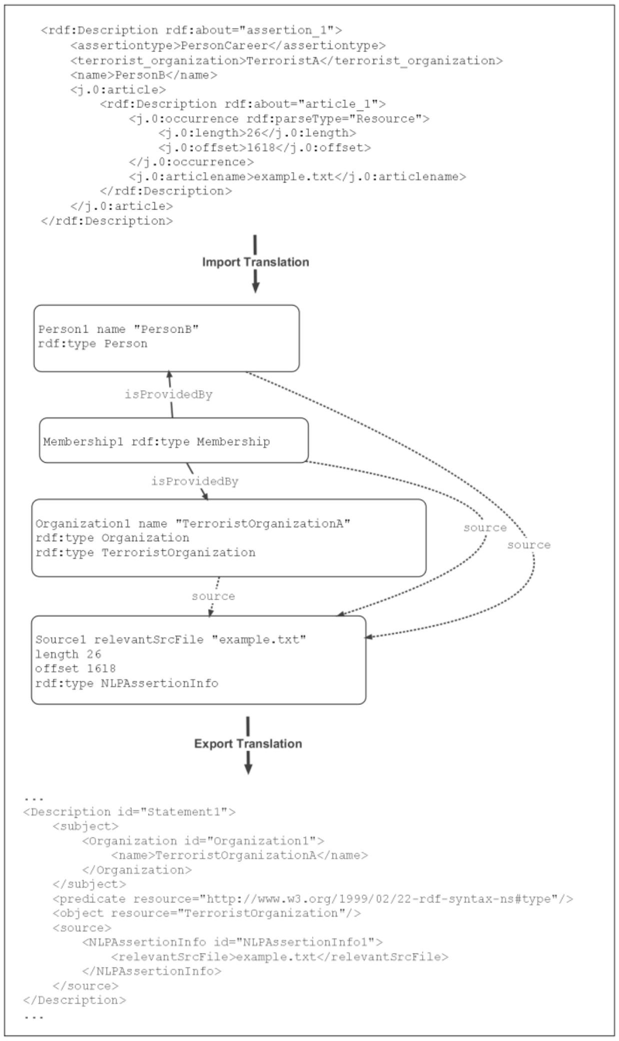
The model service has three internal components responsible for ontology management, import translation, and querying and exporting data. In this section we will briefly explain each component's purpose and key features.

The ontology management component interacts with and maintains the internal triple store. It is also responsible for triggering the reasoner and persisting the resulting inferred facts. Our triple store is implemented in Jena. The triple store contains both the OWL domain model and facts received from the NLP extraction. After the reasoner is run the triple store also contains inferred facts. Because the triple store is persistent new facts can be constantly added to the existing data to trigger further inference.

Import translation is the process by which *assertions* from the NLP output become *facts* within the semantic model environment. This translation is necessary because the NLP model and the semantic model use different ontologies to describe entities. While it is theoretically possible that the NLP and the semantic model could use the same ontology, it is impractical for several reasons. From a reusability standpoint the NLP should not have to be configured to the current ontology. For example, two instances of the system might have similar data sources and extraction needs (thus the same NLP model) but different reasoning and ontology needs (thus different semantic models). From a practical standpoint it makes sense to have the NLP output its assertions in formats close to their sources with minimal processing and translation since it doesn't have, nor should it have, the semantic tools available to do more complex translations that the system might require.

Figure 4 shows the process of import translation. The NLP output is based around the concept of an assertion, a single instance of NLP extraction. This extraction contains the extracted information as well as the source. This extraction represents an individual who is a member of a terrorist organization. The import translation takes this assertion and breaks it into its component parts. The parts are the existence of a person, the existence of a terrorist organization and the membership object that represents their relationship. Each of these component parts (in fact, every triple that comprises them) as tied through RDF reification to the source object. The import translation also makes sure that new entities are checked against existing entities of the same type and name. If a match is found, then new assertions will be connected to the existing entities thus unifying data over multiple sources.

At this time user interaction with the import mechanism is strictly through processing of the NLP extraction, however in the future it is likely that users will desire to add facts to the model directly. Therefore, the import translation component is likely to implement translations from a user-friendly schema into the model schema in the future.



```
<rdf:Description rdf:about="assertion_1">
    <assertiontype>PersonCareer</assertiontype>
    <terrorist_organization>TerroristA</terrorist_organization>
    <name>PersonB</name>
    <j.0:article>
        <rdf:Description rdf:about="article_1">
            <j.0:occurrence rdf:parseType="Resource">
                <j.0:length>26</j.0:length>
                <j.0:offset>1618</j.0:offset>
            </j.0:occurrence>
            <j.0:articlename>example.txt</j.0:articlename>
        </rdf:Description>
    </j.0:article>
</rdf:Description>
```

```
Person1 name "PersonB"
rdf:type Person
```

```
Membership1 rdf:type Membership
```

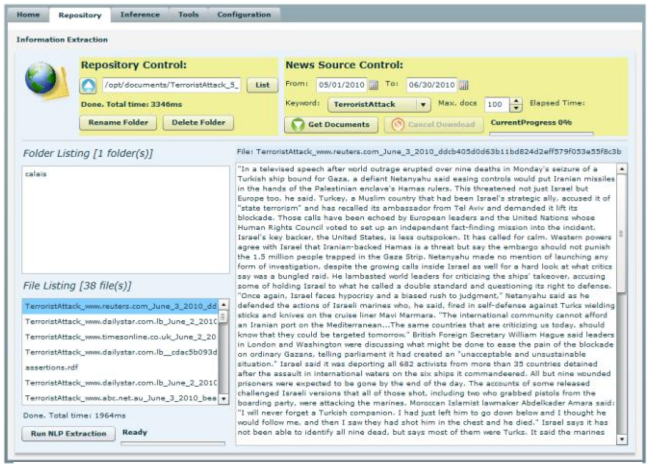
```
Organization1 name "TerroristOrganizationA"
rdf:type Organization
rdf:type TerroristOrganization
```

```
Source1 relevantSrcFile "example.txt"
length 26
offset 1618
rdf:type NLPAssertionInfo
```

```
...
<Description id="Statement1">
    <subject>
        <Organization id="Organization1">
            <name>TerroristOrganizationA</name>
        </Organization>
    </subject>
    <predicate resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
    <object resource="TerroristOrganization"/>
    <source>
        <NLPAssertionInfo id="NLPAssertionInfo1">
            <relevantSrcFile>example.txt</relevantSrcFile>
        </NLPAssertionInfo>
    </source>
</Description>
...
```

Figure 4. Import/Export Translation Process

On the opposite side of the import process is the export component. This component queries the model for desired facts and translates them to a format that can be served to clients needing the information. We can see this final translation in Figure 4. It only shows a part of the export, a single statement that identifies `TerroristOrganizationA` as a `TerroristOrganization` and the source, from which information was extracted. The complete export can contain hundreds of such assertions including relationships (such as `memberOf`) across objects. Currently the export process uses hard coded queries to create an output RDF. The output RDF is modified with regular expressions to create an XML document which is more palatable to the user interface client. In order to serve data to a more diverse range of clients the export translation process will need to be expanded to support general querying and possibly Extensible Stylesheet Language Transformations (XSLT) as well.


Home | Repository | Inference | Tools | Configuration

Information Extraction

Repository Control:

/opt/documents/TerroristAttack_5_ List

Done. Total time: 3346ms

Rename Folder | Delete Folder

News Source Control:

From: 05/01/2010 To: 06/30/2010

Keyword: TerroristAttack Max. docs 100 Elapsed Time:

Get Documents | Cancel Download | CurrentProgress 0%

Folder Listing [1 folder(s)]

calais

File Listing [38 file(s)]

TerroristAttack_www.reuters.com_June_3_2010_dd
TerroristAttack_www.dailystar.com.lb_June_2_2010
TerroristAttack_www.timesonline.co.uk_June_2_20
TerroristAttack_www.dailystar.com.lb__cdac5b093d
assertions.rdf
TerroristAttack_www.dailystar.com.lb_June_2_2010
TerroristAttack_www.abc.net.au_June_3_2010_bea

Done. Total time: 1964ms

Run NLP Extraction | Ready

File: TerroristAttack_www.reuters.com_June_3_2010_ddcb405d0d63b11bd824d2eff579f053e55f8c3b

"In a televised speech after world outrage erupted over nine deaths in Monday's seizure of a Turkish ship bound for Gaza, a defiant Netanyahu said easing controls would put Iranian missiles in the hands of the Palestinian enclave's Hamas rulers. This threatened not just Israel but Europe too, he said. Turkey, a Muslim country that had been Israel's strategic ally, accused it of "state terrorism" and has recalled its ambassador from Tel Aviv and demanded it lift its blockade. Those calls have been echoed by European leaders and the United Nations whose Human Rights Council voted to set up an independent fact-finding mission into the incident. Israel's key backer, the United States, is less outspoken. It has called for calm. Western powers agree with Israel that Iranian-backed Hamas is a threat but say the embargo should not punish the 1.5 million people trapped in the Gaza Strip. Netanyahu made no mention of launching any form of investigation, despite the growing calls inside Israel as well for a hard look at what critics say was a bungled raid. He lambasted world leaders for criticizing the ships' takeover, accusing some of holding Israel to what he called a double standard and questioning its right to defense. "Once again, Israel faces hypocrisy and a biased rush to judgment," Netanyahu said as he defended the actions of Israeli marines who, he said, fired in self-defense against Turks wielding sticks and knives on the cruise liner Mavi Marmara. "The international community cannot afford an Iranian port on the Mediterranean...The same countries that are criticizing us today, should know that they could be targeted tomorrow." British Foreign Secretary William Hague said leaders in London and Washington were discussing what might be done to ease the pain of the blockade on ordinary Gazans, telling parliament it had created an "unacceptable and unsustainable situation." Israel said it was deporting all 682 activists from more than 35 countries detained after the assault in international waters on the six ships it commandeered. All but nine wounded prisoners were expected to be gone by the end of the day. The accounts of some released challenged Israeli versions that all of those shot, including two who grabbed pistols from the boarding party, were attacking the marines. Moroccan Islamist lawmaker Abdelkader Amara said: "I will never forget a Turkish companion. I had just left him to go down below and I thought he would follow me, and then I saw they had shot him in the chest and he died." Israel says it has not been able to identify all nine dead, but says most of them were Turks. It said the marines

Figure 5. Repository Control Interface

7. User Experience

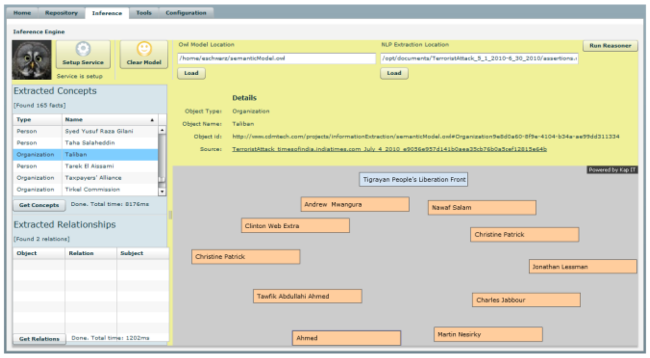
The design of the user experience is focused on providing the user with a set of tools, which can be configured at run-time, depending on user needs, to perform different tasks. The accuracy levels of existing NLP tools make user involvement essential in providing meaning and guiding both the extraction and the inference processes. The application is developed as a Web application for availability to multiple users and easy access.

The information source can be a given repository location on the Internet, or can point to an information aggregation Web service, such as "NewsExplorer". Information collection Web services can be used by simply setting the application address in the address box. The overwhelming volume of information requires that information sources be searched for documents satisfying certain search criteria. Currently, the search criteria include a date range and a set of keywords. Search criteria will expand to include additional filters, such as search-by-organization, search-by-person, search-by-event. The collected information is downloaded to a local repository for further processing. At this point, the user has the tools to manage the repository by checking the downloaded documents for relevance, deleting some documents, moving documents to other folders, or download more documents.

Once the documents in the repository are ready for processing the user can start the extraction process on a selected folder. The extraction service creates an extraction file for each document in the folder, then consolidates all extractions in one file, eliminating redundancies and resolving references. The user can run the extraction process on many folders at the same time, in which case a separate extraction session will be created for each folder. The user can also stop any extraction session before it is completed, and decide whether to keep the extracted information so far or delete it.

The user can run the inference engine on any set of extracted concepts to see what insight can be gleaned from the current document set. After setting up the service, the user can select an ontology to load. The selected ontology reflects the current interest and the required analysis. The user can clear the current ontology and load another one to perform a different analysis. Once an ontology is loaded the user can load extracted concept files, which are used for the analysis, then run the reasoner to automatically perform the analysis. The analysis results may include the creation of additional objects and/or relationships.

The analysis results are presented in tables as well as in a graph, depicting the extracted concepts and their relationships.


Home | Repository | Inference | Tools | Configuration

Inference Engine

Setup Service | Clear Model

Service is setup

Owl Model Location: /home/eschwarz/semanticModel.owl Load

NLP Extraction Location: /opt/documents/TerroristAttack_5_1_2010-6_30_2010/assertions. Load

Run Reasoner

Extracted Concepts

[Found 165 facts]

Type	Name
Person	Syed Yusuf Raza Gilani
Person	Taha Salaheddin
Organization	Taliban
Person	Tarek El Aissami
Organization	Taxpayers' Alliance
Organization	Tirkel Commission

Get Concepts | Done. Total time: 8176ms

Extracted Relationships

[Found 2 relations]

Object	Relation	Subject

Get Relations | Done. Total time: 1202ms

Details

Object Type: Organization

Object Name: Taliban

Object id: http://www.cdmtech.com/projects/informationExtraction/semanticModel.owl#Organization9e8d0a60-8f9e-4104-b34a-ae99dd311334

Source: TerroristAttack timesofindia.indiatimes.com July 4 2010 e9056e957d141b0aea35cb76b0a5cef12815e64b

Figure 6. Inference Engine Interface

The inferred objects/relationships are included in the presentation and marked as 'inferred'.

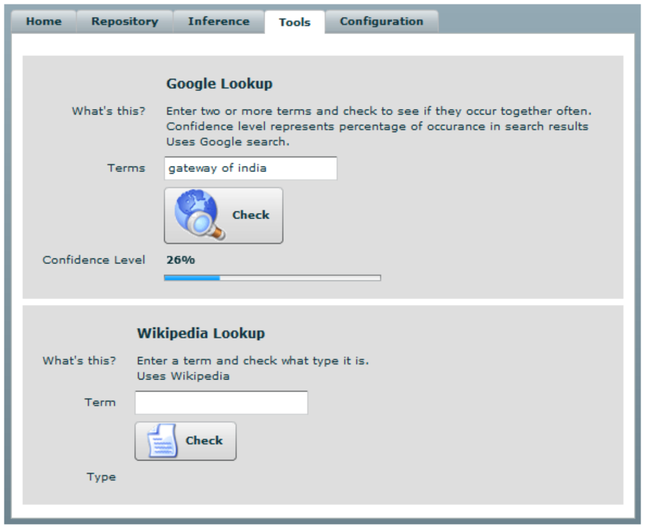
The user can load multiple extraction files and perform the analysis on all of them, or load the files one-at-a-time, and perform the analysis incrementally. The user can also clear the current analysis and start over with a new set of extraction files.

In addition to the NLP extraction and the inference engine, the application offers tools to assist the user in the analysis, by finding out more information about specific concepts or identifying the types of some concepts that may have been missed by the extraction process.

One tool determines whether two or more words appear together as a single term. This tool is based on Google search. For example, to find out whether the 'gateway of India' is a landmark or just an expression, the tool performs a search on Google using the keywords: gateway of India, then examines the first two hundred results to find out how often the words appear in the given sequence.

Another tool is based on Wikipedia search, to find out whether an extracted concept is a place, organization, person, or another type that exists in the ontology. This tool takes advantage of the structure of a Wikipedia page, which provides basic information in a structured format for a large collection of concepts.

The ability for the user to configure certain aspects of the system is an integral part of the user experience in a collaborative intelligent system. The user experience in this application includes


Home | Repository | Inference | Tools | Configuration

Google Lookup

What's this? Enter two or more terms and check to see if they occur together often. Confidence level represents percentage of occurance in search results Uses Google search.

Terms: gateway of india

Check

Confidence Level: 26%

Wikipedia Lookup

What's this? Enter a term and check what type it is. Uses Wikipedia

Term:

Check

Type

Figure 7. Additional Support Tools

the ability to configure some aspects of the system to suit the user needs for a given task. On the NLP extraction side, the user can control the NLP server and stop it and start it at any time during execution. The user can also control the batch jobs that are running on the server and stop any job or delete it. In addition, the user can view the extraction rules and edit them to introduce new rules. The internal lists, which are used by the system to determine the specific types of organizations, can also be edited. These lists assist the extraction server in determining terrorist organizations, financial institutions and other types of organizations.

8. Conclusion

The current state of NLP offers some tools to extract information from a collection of documents and recognize basic entity types in that information. In order to perform intelligent analysis on the extracted information, a context has to be provided along with some guidelines for the analysis. We combine existing NLP tools with an OWL ontology to provide a rich environment for performing intelligent analysis. The ontology provides the context and the structure of knowledge in the given domain with the rules, which govern the classification and membership of objects in concepts in the ontology, and provide basic inference mechanism for creating new knowledge. The existing NLP tools extract information from a collection of documents and identify the basic types for entities in the extracted information. Mapping then takes the extracted basic entities and populates the ontology. The inference mechanism runs on the ontology objects and may create new objects or relationships, thus expanding the ontology. The overall system incorporates these tools as collaborative agents that provide and utilize services through a Web server based on Tomcat and Apache. The user interacts with the system through a Web interface that provides assistance with the management of sources and documents, and offers support with the extraction, analysis, and retrieval of information contained in and derived from the repository.

References

AlchemyAPI. Orchestr8, http://www.alchemyapi.com/, visited 8/10/2010.

Apache Clerezza. Apache Software Foundation – Apache Incubator, http://incubator.apache.org/clerezza/, visited 8/10/2010.

Apache UIMA. Apache Software Foundation. http://uima.apache.org, visited 8/10/2010.

Jem Rainfield. BBC World Cup 2010 dynamic semantic publishing. http://www.bbc.co.uk/blogs/bbcinternet/2010/07/bbc_world_cup_2010_dynamic_sem.html ,/ visited 7/15/2010.

BigOWLIM. Ontotext AD, http://www.ontotext.com/owlim/big/, visited 8/10/2010.

ClearTK. Center for Computational Language and Education Research (CLEAR), University of Colorado, Boulder. http://code.google.com/p/cleartk/, visited 8/10/2010.

Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. Journal of Information Science, June 2010, 36: 306-323.

GATE. University of Sheffield, Computer Science Department. http://gate.ac.uk, visited 8/10/2010.

Jena. Hewlett-Packard Development Company, LP, http://jena.sourceforge.net/, visited 8/10/2010.

LingPipe. Alias-i, http://alias-i.com/lingpipe/, visited 8/10/2010.

Loebner Prize in Artificial Intelligence Web page, http://www.loebner.net/Prizef/loebner-prize.html, visited 8/10/2010.

Michael T. Flynn, Matt Pottinger, Paul D. Batchelor. Fixing Intel: A Blueprint for Making Intelligence Relevant in Afghanistan. Center for New American Security, Voices from the Field, January 2010. http://www.cnas.org/node/3924

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni. Open Information Extraction from the Web, IJCAI 2007.

NewsExplorer. Europe Media Monitor, http://emm.newsexplorer.eu/, visited 8/10/2010.

OpenCalais RDF. Thomson Reuters. http://s.opencalais.com/1/pred/, Visited 8/10/2010.

OpenCalais. Thomson Reuters. http://www.opencalais.com, Visited 8/10/2010.

OpenNLP. http://opennlp.sourceforge.net/, Visited 8/10/2010.

OWL Specification. W3C, http://www.w3.org/TR/OWL2-overview/,October 2009.

Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Proceedings of the 2nd International Workshop on Meta-Modelling, WoMM 2006, October 12-13, 2006, Karlsruhe, Germany. LNI 96 GI 2006, ISBN 978-3-88579-190-4.

RDF/XML Syntax specification. http://www.w3.org/TR/REC-rdf-syntax/, February 2004.

Stanford Named Entity Recognizer. The Stanford Natural Language Processing Group, http://nlp.stanford.edu/software/CRF-NER.shtml, Visited 8/10/2010.

Stuart Hendren and John Yesberg. Qualifying Triples and Selective Reasoning - OWL Experiences and Directions, Fifth International Workshop - co-located with ISWC 2008, Karlsruhe, Germany, October 26-27, 2008.

Vrandecic, D., Völker, J., Haase, P., Tran, D., Cimiano, P.: A metamodel for annotations of ontology elements in OWL DL. In Proceedings of the 2nd International Workshop on Meta-Modelling, WoMM 2006, 109-123, Brockmans, S., et al., eds., 2006

Witte, R., Krestel, R., Kappler, T., and Lockemann, P. C. (2010). Converting a historical architecture encyclopedia into a semantic knowledge base. IEEE Intelligent Systems, 25:58–67.