To appear in Real World Semantic Web Applications, V. Kashyap and L. Shklar, Eds., IOS Press, 2002.

Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content

Brian Hammond, Amit Sheth*, Krzysztof Kochut*
Semagix, Inc.
Also, LSDIS Lab, Computer Science, University of Georgia

Abstract. Traditionally, automatic classification and metadata extraction have been performed in isolation, usually on unformatted text. SCORE Enhancement Engine (SEE) is a component of a Semantic Web technology called the Semantic Content Organization and Retrieval Engine (SCORE). SEE takes the next natural steps by supporting heterogeneous content (not only unformatted text), as well as following up automatic classification with extraction of contextually relevant, domain-specific (i.e., semantic) metadata. Extraction of semantic metadata not only includes identification of relevant entities but also relationships within the context of relevant ontology.

This paper describes SEE's architecture, which provides a common API for heterogeneous document processing, with discrete, reusable and highly configurable modular components. This results in exceptional flexibility, extensibility and performance. Referred to as SEE modules (SEEMs), which are divided along functional lines, these processors perform one of the following roles: restriction (determine the segments of the input text to operate upon); enhancement (discover textual features of semantic interest); filtering (augment, remove or supplement the features recognized); or outputting (generate reports, annotate the original, update databases, or other actions).

Each SEEM manages its configuration options and is arranged serially in virtual pipelines to perform designated semantic tasks. These configurations can be saved and reloaded on a per-document basis. This allows a single SEE installation to act logically as any number of Semantic Applications, and to compose these Semantic Applications as needed to perform even more complex semantic tasks.

SEE leverages SCORE's unique approach of creating and using large knowledge base in semantic processing. It enables SCORE to provide flexible handling of highly heterogeneous content (including raw text, HTML, XML and documents of various formats); reliable automatic classification of documents; accurate extraction of semantic, domain-specific metadata; and extensive management of the enhancement processes including various reporting and semantic annotation mechanisms. This results in SCORE's advanced capability in heterogeneous content integration at a higher semantic level, rather than syntactical and structural level approaches based on XML and RDF, by supporting and exploiting domain specific ontologies. This work also presents an approach to automatic semantic annotation, a key scalability challenge faced in realizing the Semantic Web.

Keywords: semantic applications, document enhancement platform, semantic metadata, automated metadata extraction, automatic classification, semantic content integration, semantic annotation

1. Introduction

Today, many organizations have access to a broad variety of information resources, including collections of internal documents, data repositories, and external public and subscription-based content. Collectively referred to as enterprise content, these information resources occur in a wide variety of formats and digital media. In order to truly take advantage of enterprise content, the next generation of enterprise content management systems require novel, advanced capabilities.

Making information resources searchable is a step in the right direction, but to get the most out of these resources, users must be able to access information through mechanisms that blend seamlessly into the workflow and applications they use. The value of internal resources may be significantly enhanced when coupled with open source information on the World Wide Web, private networks, and the deep web.

Additionally, it would be highly valuable to establish links between semantically related internal and external documents. For this, documents must be converted from simple text buckets to collections of meaningful information that can be collated with web resources. In other words, traditional document management needs to be enhanced with knowledge management and semantic technology.

By combining classification with syntactic and semantic metadata extraction, the SCORE (Semantic Content Organization and Retrieval Engine) Enhancement Engine (SEE) component has been designed to do just that. The process of automatic classification puts the user in the ballpark, and contextual metadata extraction puts him in the seat; or, in the parlance of the World Wide Web, classification puts the user in the web ring, and metadata puts him on the web page.

Systems that can be built around this type of functionality are as diverse as the documents with which they work. SEE is a system designed with flexibility and integration issues as high priorities from the ground up. SEE's implementation centers around three primary concepts: core document service, modular processing, and superior configurability.

Core document processing tasks, such as format normalization (e.g., tag versus text) and linguistic analysis (e.g., sentence boundaries), are needed to address the issue of format variety. Modular processing and configurability address the wide variety of workflows into which SEE must be integrated, as well as providing a mechanism through which classification and metadata extraction may be coupled.

This allows for the extraction of domain-specific metadata from raw text (Figure 1). First, the classification technology determines the category for a document and hence determines the domain of discourse. Then semantic metadata particular to the domain is targeted and extracted. This includes specific named entity types of interest in the category (such as “CEO” in “Business,” “Chipset” in “Technology,” or “SideEffects” in “Pharmacology”) as well as category specific, regular expression-based knowledge extraction. This domain-specific metadata can be regarded as semantic metadata, or metadata within context. It is possible to associate multiple domains with a document and extract different semantic (domain-specific) metadata for that document. It is also possible to take more than one pass of the process shown in the figure. The classification and extracted metadata forms the basis of semantic annotation of the documents. Although currently an XML based format is used for annotation, in future RDF and OWL can be easily supported.

The automatic extraction of semantic metadata from documents which have not been previously associated with a domain is a unique feature of SEE. In essence, this transports the document from the realm of text and mere syntax to a world of knowledge and semantics in a form that can be used for computation.

By providing a framework for working with the relevant knowledge in a document rather than the text, SEE allows for a new breed of semantic applications to be implemented in one easy-to-use, extensible package. HTML documents can be moved from the World Wide Web to the Semantic Web [1] through the process of semantic annotation that attaches automatically extracted semantic metadata to text.

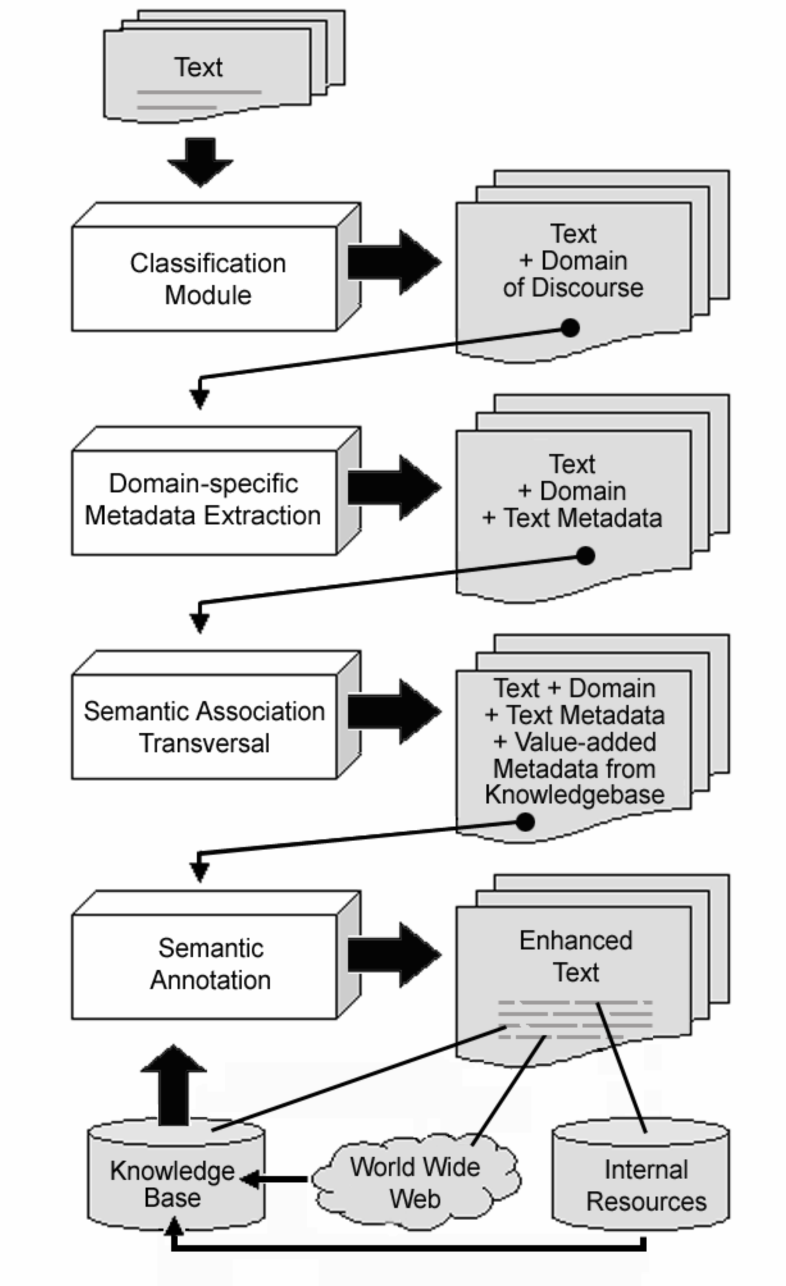

Text
Classification Module
Text + Domain of Discourse
Domain-specific Metadata Extraction
Text + Domain + Text Metadata
Semantic Association Transversal
Text + Domain + Text Metadata + Value-added Metadata from Knowledgebase
Semantic Annotation
Enhanced Text
Knowledge Base
World Wide Web
Internal Resources

Figure 1: Through the use of automatic classification and contextually relevant KnowledgeBase information, domain specific metadata can be extracted from a document, enhancing the meaning of the original and allowing it to be linked with contextually heterogeneous content from multiple sources.

To summarize, the unique features of SEE are: core document services with a modular functionality; the use of automatic classification to allow for domain-specific metadata (semantic metadata) extraction; and a simple, unified format for configuration of modules, which allows a single installation of SEE to be used for an immense variety of applications.

This chapter is organized as follows. Section 2 provides a relevant overview of the SCORE technology, of which SEE is a component. Section 3 provides an overview of SEE. Section 4 discusses core document services SEE supports. Section 5 discusses four types of SEE modules (SEEMs) that support respective semantic tasks. Section 6 discusses configuration and implementation features. Section 7 discusses Virtual Machines, which support basic Semantic Applications, such as domain specific metadata extraction and integration of internal and external knowledge sources, by combining various SEEMs. Section 8 provides additional details on several SEEMs and an example of automatic semantic annotation, which includes identification and extraction of contextually relevant relationships. Section 9 discusses related work, and Section 10 offers conclusions.

2. Background: SCORE Technology

Ontologies play a central role in most semantic technologies. They form the basis for syntactic and semantic metadata, which can be used for annotating or tagging content. The content's context determines which semantic metadata to extract. Automatic classification technology helps select the context by classifying documents into one or more categories and extracting or inferring semantic metadata corresponding to one or more contexts.

We have divided ontology into two related components: the definition component, called the WorldModel and the assertional component called the Knowledgebase. As with the specification for ontologies, the WorldModel and Knowledgebase definition process involves domain-specific expertise as well as an understanding of eventual application requirements. While some clustering techniques can provide initial input to semi-automate this process [2, 3], it cannot generally be completely automated if high-quality results are needed. Also the techniques that rely on learning from structured data [3] are not particularly practical.

The Knowledgebase reflects the subset of the real world for which a semantic application is created. As such, it is an important part of the solution. It enables the extraction of value-added semantic metadata, such as the ticker symbol "INTC" when a document only mentions the company "Intel." It also provides the framework for semantic associations.

In addition to these two components, SCORE provides a query-processing system. A comprehensive suite of APIs uses this query-processing capability to support rapid development of semantic applications such as search, directory, personalization, syndication, and custom enterprise applications [4].

The operation of a SCORE technology-based system involves three independent activities, as illustrated by the sections separated by the gray lines in Figure 2 [5]. Defining the WorldModel and Knowledgebase is the first activity (Figure 2, top right). Knowledge extraction agents manage the Knowledgebase by exploiting trusted knowledge sources. Different parts of the Knowledgebase can be populated from different sources. Various tools help detect ambiguities and identify synonyms. Commercial deployments of SCORE can be expedited with a predefined WorldModel and Knowledgebase.

Content processing comes second (Figure 2, left). This includes classifying and extracting metadata from content. The results are organized according to the WorldModel definition and stored in the Metabase. Knowledge and content sources can be heterogeneous (XML, RDF, static and deep Web pages, database, or documents in various formats), internal or external to the enterprise, and accessible in Push (content feeds or database exports) or Pull (Web sites or database queries) modes.

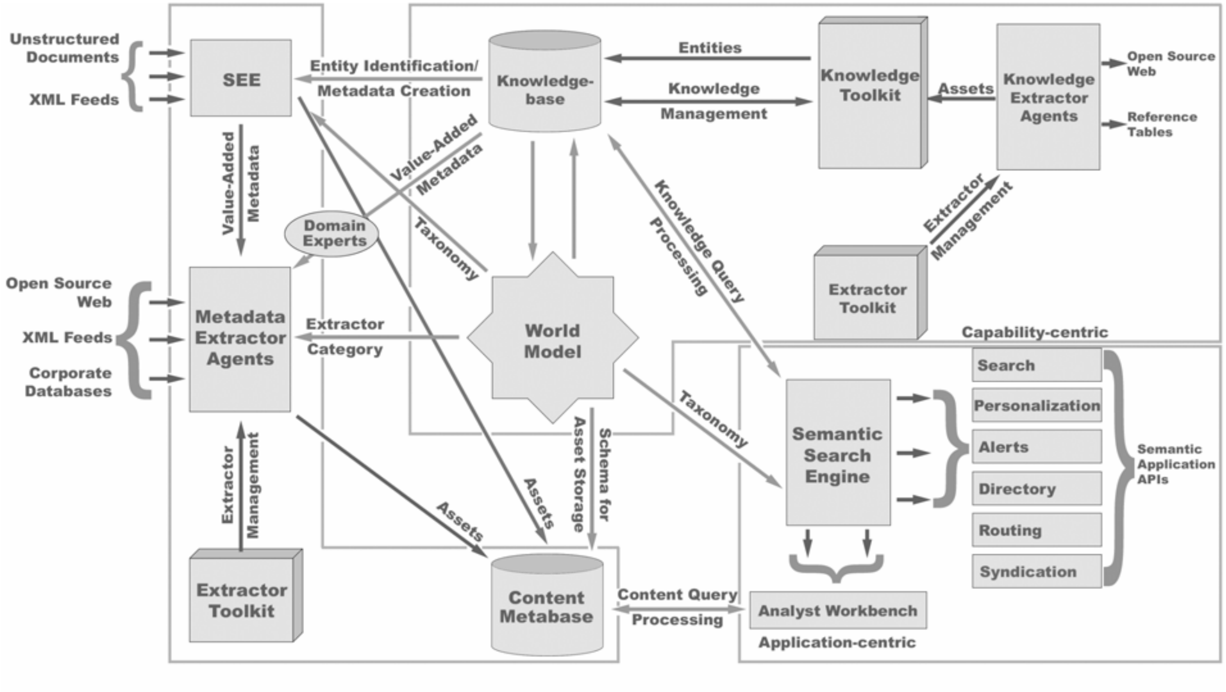


Figure 2: ***SCORE system architecture. The three activities bounded by gray lines cooperate through XML-based knowledge and metadata sharing.***

Support for semantic application comes last (Figure 2, bottom-right and top left). The Semantic Search Engine (SSE) processes semantic queries and returns results in XML from the Metabase. An API for building traditional and customized applications facilitates GUI creation. SEE provides a modular framework for domain-specific metadata extraction through the use of automatic classification techniques.

3. SEE Overview

SEE provides a variety of services and functions as well as a framework for implementing semantic applications.

A semantic application may involve a number of inter-related *semantic tasks*. Examples of semantic tasks include text classification, metadata extraction, identification of semantic associations, and enrichment of a document by annotating it with pertinent semantic information. The SEE framework allows for an easy integration of such semantic tasks in order to achieve a more complex objective. As an example, such semantic tasks may be organized into a pipeline to create a large semantic application in which text documents obtained from a network live feed are first classified into a node in taxonomy. Subsequently, entities relevant to the determined category of the document (in the Knowledgebase) are identified. Furthermore, relationships existing among them are uncovered, and other entities that may be important, though not explicitly present in the document, are identified via known relationships. Ultimately, the original document is enriched by suitable annotations, including the category of the document, recognized entities, identified relationships and other relevant entities. A detailed description of some of the semantic tasks is presented in Section 8.

The core concepts behind building systems based on SEE include:

Core Document Services: A modern document system must be able to handle XML, HTML, raw text and various other document formats[1]. The text segments must be decomposed into chunks, such as words, phrases and sentences. Access methods for going from character to word to sentence position are provided to maintain maximum flexibility. Text annotation methods allow for new data to be added easily to existing documents. This allows SEE module implementers to concentrate on implementing text algorithms without the hassle of basic parsing and character offsets.

SEE Modules (SEEMs): Each text-processing task performed in the system is implemented as a SEEM. The functionality provided usually falls into one of four major divisions: restriction (determine the segments of the input text to operate upon); enhancement (discover textual features of semantic interest); filtering (augment, remove or supplement the features recognized); or outputting (generate reports, annotate the original, update databases, or other actions).

These modules are then arranged into pipelines through which incoming documents flow. As a document passes through the modules, text segments of semantic interest are identified. These segments of text are referred to as DocumentFeatures. DocumentFeatures may also be attached to a document based on features found or other criteria such as automatic classification.

When Entities are discovered in a document, other Entities can be attached to the document through the traversal of semantic associations. These traversals can be single-step (for example "Linux" has "created by" relationship with "Linus Torvalds") or may follow user-defined paths (such as "Microsoft" has a "competes with" relationship with "Sun Microsystems" which has a "created" relationship with "Java.") This allows for connections to be created between entries in the Metabase by exploiting domain-specific knowledge. This represents a unique functionality of SEE and SCORE.

Modular Configurations: All SEE modules use a common configuration API that produces XML output, which provides information about the use, type and acceptable values for its options. By using XSLT, different aspects of this configurability can be exposed via a GUI to the end user. These configurations can also be saved on the SEE server to be later loaded by name on a per-document basis. In addition to the module options, their order in the pipeline is also stored. Since SEE is HTTP based, configuration commands can be added to hyperlinks to facilitate custom annotation.

Virtual Engines: Each configuration of SEE can be thought of as a discrete functional unit. This allows a single installation to be used by scripts as though it were in fact several distinct semantic applications. This facilitates the creation of systems that perform advanced semantic processing, such as the extraction of category metadata extraction based on automatic classification, the linking of HTML documents to various internal and external resource by metadata type and other applications which can benefit from performing additional Semantic processing based on semantic data. The effective use of semantic feedback is a unique and powerful property of SEE.

[1] We assume use of format converters such as Verity's Keyview [http://www.verity.com/keyview/] and Stellent's Inso/Outside In [http://www.stellent.com] to help obtain text from proprietary document formats.

More detailed information about core enhancement modules is provided in the remaining sections to demonstrate the functionality and usage of the system. An example of the HTML annotation with semantic metadata is also given.

4. Core Document Services

The importance of the World Wide Web as a content source makes it imperative that the system be able to handle documents that are in HTML. The emergence of the Semantic Web mandates that SEE also be able to support XML (and in the future, RDF) as well. In SEE, before any module processes a document, it must first pass through three parsing phases: format parsing, word parsing and sentence/phrase parsing.

Format parsing is primarily concerned with the separation of tags from text. Despite their similarity and common ancestry, HTML is much more problematic to work with than its younger cousin, XML. Though at first glance this may sound like a non-issue because it is an operation so commonly performed, the parsing of HTML represents a number of challenges.

Ironically, the problem seems to have arisen from the aspirations of HTML parser writers in the past to make life easier for those who deal with HTML. In the beginning HTML browsers, such as Mosaic, were fairly strict and would complain about small mistakes in HTML authorship. At that point in time, most HTML documents were still written by humans directly rather than through authoring tools, so simple errors were a common source of annoyance for web surfers.

As browsers improved their ability to deal with partially malformed HTML, authors have become less and less concerned with fixing mistakes, or were perhaps simply unaware of their existence. All of these factors have led to very nice HTML parsing in browsers but also to many rather nasty quirks for the implementers of new parsers. By contrast, XML is much easier to work with because it imposes tighter constraints on well-formedness.

During the next two parsing phases, SGML entities representing ampersand, quote and other punctuation symbols must be resolved and noted. They must be resolved so that their true meaning as punctuation can be maintained, and they must be noted so that character offsets can be adjusted accordingly.

This can be best illustrated by a brief example: “AT&amp;T is a telecommunications company.” The text logically references “AT&T”, which is four characters in ASCII, but is written using eight. This becomes an issue when it comes time to identify DocumentFeatures and later when it is time to perform semantic annotation.

Once those portions of the document that are markups are identified, the next step in the document preparation phase is to segment the text into individual words. Words are always broken up by white space and often by punctuation. However, there are cases in which punctuation is considered integral to the word itself. Some examples include: O’Connor, AT&T, and Matthews-Morgan. This results in a list of word tokens that is then passed into the sentence/phrase analyzer.

When identifying DocumentFeatures, some false positives can be avoided by using end-of-sentence and end-of-phrase information. SEE currently uses a number of heuristics based primarily on punctuation and capitalization to detect breaks corresponding to sentence and phrase boundaries. Though currently designed for English syntax, the SEE sentence/phrase analyzer is being enhanced to support other languages; this is made easier by its object-oriented design. This can be contrasted with systems that use table-driven parsing.

5. SEE Modules

Once a document has been prepared, it is sent to the SEE Pipeline. Each Pipeline is composed of a number of SEEMs. Control flows from SEEM to SEEM with different parts of the semantic tasks performed by different modules. The individual modules perform discrete tasks that usually fall into one of the following roles: *restriction, enhancement, filtering* and *outputting*.

Restriction Modules: The purpose of modules that perform restriction tasks is to identify the portion or portions of a document that should be enhanced. Many web pages, for example, have sidebars with text unrelated to the main story. To address this issue, there is a SEEM which attempts to identify the primary text. It then informs SEE to ignore the rest of the document for enhancement purposes. Similarly, often only certain elements of an XML document should be examined.

In the interest of efficiency, SEEMs that restrict are typically placed in the pipeline **before** the built-in sentence/phrase boundary detection component. This is because there is no point in performing such analysis just to ignore it.

Restriction modules are unique in that they are almost the only SEEM which needs to be concerned with the original document format to fulfill their task.

Enhancement Modules: As might be expected from the name, SEEMs that fall into the enhancement category are the métier of SEE. The primary purpose of such modules is to identify DocumentFeatures that occur in the original text.

Usually, a DocumentFeature is a matched region of text placed within a semantic context. Examples of such DocumentFeatures include the proper names of people, places and things (named entities); monetary sums, percentages and dates (unnamed entities); and phrases.

There are other kinds of DocumentFeatures not directly associated with a particular section of the document. Examples of such features include classification results, relationships between named Entities found in the text, and Entities that may be added to the text by contextual rules based on Entities found in the input document. Filter or output modules usually add the latter. These DocumentFeatures represent metadata for the document that are the basis of semantic annotation.

Filtering Modules: The primary purpose of filtering modules is to modify the set of DocumentFeatures generated by enhancement modules. These modules can be thought of as the "decision makers" of the SEE. One of the most important purposes of filtering is the removal of semantic ambiguity. In SEE, this can occur when a single region of text matches multiple instances, such as "John Smith".

By using relationships between Entities found in the text and knowing the domain of the document through classification, SEEMs can make reasonable decisions about which of many potential Entities to keep and which to discard. Filter modules may also add additional DocumentFeatures based on other recognized features, such as adding the name of a company when its CEO is mentioned in a document.

Output Modules: One or more output modules produce the final result of a trip through the SEE pipeline. Output typically takes the form of report generation or annotation of the original document. In the former, a new document is created with information about the

DocumentFeatures found in the input document. Reports might be in XML, plain text or SQL, depending on the application.

By contrast, in the case of annotation, the original text is augmented with semantic metadata added in line with the sections of text with which it is associated. In the case of HTML, this may take the form of new hyperlinks to repositories containing detailed information-recognized Entities or, in the case of classification, possibly links into the spot in a directory containing similar documents. The WorldModel or one of its components (for example, a subtree rooted in the "Sports" node of the WorldModel) in SCORE may be used as the domain ontology.

6. Modular Configurations and Other Implementation Issues

Each SEEM is compiled into a separate Dynamically Shared Object (DSO) file. Each time SEE starts, it loads and initializes each SEEM from its corresponding DSO file. If a SEEM fails to initialize, the problem is logged, and initialization of SEE continues. In the case of a faulty or misconfigured module, the rest of the system will still remain available. Compiling a SEEM into a separate executable file results in many advantages: additional functionality can be added to the existing installation in a straightforward, extensible way; bugs can be fixed simply by swapping out modules; and new functionality can be added by extending a C++ class, which helps reduce development time for extending functionality.

Due to the plug-and-play nature of SEEMs, it is necessary to provide a common mechanism so that new modules can be added to an existing installation without affecting the current SEE or requiring it to be upgraded. To this end, all SEEMs make use of a common configuration API. This API produces XML that describes the purpose of the module as well as information about its options. The description of each option specifies its purpose, value type (string, integer or boolean), range of accepted values, and the default and current value of all of the module variables.

When SEE processes a request to send its current configuration, it returns an XML description that includes a list of saved configurations, the modules available, and the modules currently in the pipeline. Then SEE calls a method for each SEEM which produces its tasks-specific XML description. This allows SEE modules to be largely self-documenting and easily configured. Configuration of SEE is highly customizable by using XSLT. Dynamic GUIs can be created on a per-user basis so that novice users need not be overwhelmed by options they do not want to consider or do not need to change. Additionally, the GUI can be implemented to closely match the look and feel to which users have become accustomed, adding to the comfort level.

Configurations can be saved on the server with SEE for later use. These configuration files can be referenced by name and invoked on a per document basis. Once a configuration is loaded, the state of SEE and its modules remains set until another configuration file is loaded or a configuration command sent. Because SEE saves a copy of the last valid configuration loaded to disk and re-reads it on initialization, it retains its state even in the event of a catastrophic system failure, such as a power outage.

The use of named configuration files also allows a simplified interface consisting of a drop-down menu box, where users can select canned configurations for different purposes. In this way, SEE can support a wide range of applications as middleware, where the end user does not even need to be aware of the full range of options available to be configured. More importantly however, since the configurations can be invoked on a per document basis, and because SEEMs can implement such a diverse range of semantic functionality, this provides a mechanism whereby a single SEE can be used for completely different purposes. Logically, each configuration can be thought of as a different application instance running under SEE.

7. Virtual Engines

By thinking of each configuration of SEE as a distinct functional unit, the utility of SEE is dramatically expanded. Not only can SEE perform a wide variety of semantic tasks, but through the use of scripts, these SEE application instances can be chained together to create pipelines for semantic processing. Since each step in the pipeline is a semantic application, the meaning of the metadata can be maintained throughout the entire process. This allows the creation of systems that can make decisions and take action within the context of meaning rather than data. This allows for semantic application beyond annotation, where semantics can play a role in the decision making process for automated agents.

When a document comes into SEE, it is first identified as a collection of words and sentences. Then, semantic metadata is recognized and added in the form of DocumentFeatures. At this point, the document has been augmented with knowledge. To make the most of this augmentation, applications must have the semantic savvy to exploit it. To illustrate this concept, consider the following scenarios.

Domain Specific Metadata Extraction: The *RightNow News Service* has a stream of documents coming in from reporters around the globe. The documents are short, two or three paragraphs of text each, and fall into one of twenty possible categories. The goal is to classify and extract category specific metadata for each incoming document.

Metadata to be extracted is based on the relevant domain specific ontology. For documents that fall into the financial category, the system needs to extract dates, monetary amounts, companies, CEOs and ticker symbols. If a document is classified as baseball, the names of teams, players, managers and scores should be identified. For other categories, domain specific metadata can also be discovered.

Because SEE integrates classification and metadata extraction, an application like this can be implemented very quickly. Quite often, companies and groups interested in classification and per category metadata will have amassed large collections of documents that have been classified already. These corpora can be used to train the automatic classification subsystem of SEE.

Integrating Internal and External Knowledge Resources: Despite the fact that *TransMeta Investment World* (TMIW) analysts have access to an extensive collection of in-house financial data, many of them still find external web resources extremely useful. Always on the lookout for competitive advantage, TMIW wants a way to combine these two sets of resources.

HTML documents can be annotated with links to a local Knowledgebase for Companies, a real-time streamer for stock symbols and hyperlinks to *biography.com* for company executives. SEE can accomplish this contextual annotation by processing a given web page multiple times with slightly different configurations controlling the type of Entities recognized each time.

Because SEE can switch gears on a per document basis, a single engine could be used to perform both of these completely different tasks. Since they are based on HTTP, SEE configuration commands can be “embedded” in hyperlinks, even in hyperlinks generated by SEE HTML annotation.

8. Selected SEE Modules in Detail with Examples

Since the primary purpose of SEE is to link documents with knowledge, the SEEMs that perform enhancement are perhaps the most interesting. A module known as the eMDExEnhancer recognizes named Entities. Dates, percentages and monetary amounts are matched by the eRegexpEnhancer. Potentially interesting regions of text are located by the ePhraseEnhancer for later analysis. Automatic classification is done by the eClassifyEnhancer.

The eMDExEnhancer

The Knowledgebase in SCORE keeps track of named Entities such as people and places [5]. The eMDExEnhancer is the module responsible for matching regions of text with entries in the Knowledgebase.

This is done using a number of pattern matching rules specific to various types of Entities. Person names, for example, are usually stored in Knowledgebase in the canonical form of: "LAST_NAME, FIRST_NAME MIDDLE_NAME*(, SUFFIX*)?".

In order to accomplish name recognition, this form must be mapped to a more natural language flavored version. Aliases, also called nicknames, must be considered as well.

The eMDExEnhancer makes a number of passes over each document. In the first pass, only the most probable matches are retained. So in a case like: "John Black is a man. Black wears pants," the first pass would identify "John Black" in the first sentence but not "Black" in the second. Once this stage is completed, probable references such as "Black" are then associated with their most likely Entities from stage one.

Obviously, this process if fraught with pitfalls, and a number of measures have been taken to avoid false positives. The use of sentence boundaries is one such example. Without this functionality, "John Black" would also be matched in following text: "I have a friend named John. Black umbrellas are very popular."

Despite these precautions, false positives do occur. Another problem is that the Knowledgebase may contain multiple Entities named "John Black." To deal with cases like these, SEE uses a number of different mechanisms.

Since the Knowledgebase tracks relationships between Entities, a scoring mechanism uses this information to try to winnow down the possible matches by analyzing the connection between the Entities found in the text directly and through a single intermediary Entity not mentioned in the text. The aggressiveness of this reduction can be controlled and configured.

Another approach that further illustrates the utility of combining classification with metadata extraction is to use the domain of the document from classification to help determine the best Entities. In order for this to be effective, Entities must be associated with the classification taxonomy. Alternatively, EntityClasses may be associated with the domains and hence Entities that fall within a given class. In this way, when a document is assigned to the domain of politics, politicians are extracted preferentially. This technique reaches it limits, however, when the Knowledge contains multiple Entities named "John Black" who are politicians.

Conversely, Entities found in a corpus can be replaced with the corresponding EntityClasses in an attempt to replace the instance with an abstracted reference. This is analogous to "word concepts" in the LexisNexis classification system [6]. By so doing, a collection of training documents can be rendered more general, since occurrences of "John Black" would then match any politician once the substitution was made. Since the eMDExEnhancer works off the Knowledgebase, this information can be easily maintained.

The eRegexpEnhancer

As described in the previous section, named Entities are stored in a canonical representation mechanism known as the Knowledgebase. Dates, percentages, monetary amounts, phone numbers, email addresses and URLs are also potentially of interest. In order to allow this sort of knowledge to be exploited, the eRegexpEnhancer uses user-defined regular expressions to match arbitrary text.

Regular expressions are a popular and powerful way to perform pattern recognition. SEE's document normalization mechanism allows for regular expressions to span lines in the input text and ignore issues of encoding such as ""&amp;", "&quot;", and "&lt;". Regular expressions have been used extensively in the past and are immensely popular in the programming community. By allowing the author of expressions to concentrate on defining the pattern without worrying about text format issues, their utility is expanded.

In this context, regular expressions can be thought of as "data scissors" that cut out meaningful pieces of documents, which can then be pasted into a semantically defined context. While in general this functionality is not unique [7], the ability to use regular expressions on a per category basis is.

The eRegexpEnhancer SEEM maps user-defined patterns to user-defined semantic types. Because it is a SEEM, this information is part of its configuration information. This means that sets of regular expressions mapping to domain specific semantic types can be implemented quite readily. For each domain, a user of the system can assign semantic significance to any data for which they can define a regular expression.

So, if a document is categorized as a story about the stock market, for example, certain keywords such as "buys," "sell," "downgrade," and "sharp upturn" could be mapped to "FinancialEvent." When coupled with the ability to match named Entities, such as the names of companies, stock exchanges and ticker symbols, this simple mechanism can be powerful. Implementing a system like this to watch popular financial web sites or news feeds could be done with minimal effort, because the semantic infrastructure is already in place. The ability to use email addresses, phone numbers and URLs as semantic features may also be useful.

The ePhraseEnhancer

The last of the modules that create DocumentFeatures mapping directly to text regions is the ePhraseEnhancer module. Its purpose is to locate interesting strings of words in the document corresponding to phrases.

Since SEE does not currently provide NLP functionality, such as parts-of-speech tagging [8], ePhraseEnhancer identifies phrases as lists of capitalized words possibly separated by a short list of common words such as "of" and "the." Though this could be implemented as a regular expression, the document facilities of SEE make it much more efficient to have a specialized module. The phrases found are interesting for a number of reasons and are worth considering for a moment.

The names of people are frequently identified by the ePhraseEnhancer, and since this may potentially overlap with named Entity discovery, any phrases that match the same text region already matched by another DocumentFeature are discarded. Phrases have the lowest priority of all DocumentFeatures because they are so general and have such a low semantic value. In the cases where they are retained, phrases may still match a named Entity that was not picked up. This occurs in a number of scenarios. This module can pick up entities not in the

Knowledgebase as well as alternative spellings and aliases for Entities that are. This can provide a handy mechanism for spotting shortcomings in the named Entity collection.

For a new installation starting from a fresh collection of documents with few Entities, the ePhraseEnhancer can be used to jump-start the Entity discovery process. Each file can be converted to a list of phrases and a TFIDF (term frequency / inverse document frequency) [9] value can be computed for each. By selecting the highest scoring phrases, probable Entities can be identified.

The eClassifyEnhancer

Automatic classification maps incoming documents to a domain in the taxonomy. In SEE, a committee of classifiers performs this classification. Numerous researchers have designed and implemented classification at a wide variety of academic [10] and commercial [11] institutions. After more than thirty years of research, there is still considerable debate and discussion over how this task can best be accomplished. In the document sets on which SEE's classification subsystem has been tested, varying from a test corpus such as Reuters-21578 and company internal portal with proprietary data and heterogeneous documents, there has been no single method which has always performed best. Instead of limiting classification to a single method, a committee approach has been taken which allows the text collection to dictate the classifiers used [12].

In addition to choosing a best-fitting collection of techniques, the classification committee is able to combine the results from a number of methods to arrive at a consensus about the probable category for a given document. As might be expected, this sort of approach is at its best when the classification techniques represented in the committee, function as differently as possible.

As the accuracy for the classifiers tends to vary for each document set, their results are weighted and combined as follows. First, the results for each classifier over all of the categories are scaled and normalized so that they fall within the range of zero to one. These per-classifier scores are then multiplied by a weight and summed to create a per-category score. Finally, this combined score is normalized with the score of one being assigned to the most probable category. These weights are determined at training time according to a variant of Larkey's and Croft's "Weighted Linear Combination" approach [13]. A genetic algorithm based approach is being developed to determine these weights as well [14].

Below are some results based on a subset of the Reuters-21578 text categorization test collection. Though the Lewis-Split (a well-known partitioning of the Reuters document set into training and test sets) used here details 144 separate categories, only a subset is shown here. The criterion used is that each category must contain at least 160 documents in its training set**.**

Table 1: The per-category, per-classifier accuracy rates for the Reuters document set where the number of training documents was at least 160 per category.

	acq	corn	crude	earn	grain	intst	money	ship	trade	wheat
Bay.1	64.95	0	7.98	2.02	1.39	1.52	1.69	1.15	0.85	1.45
Bay.2	99.58	0	3.72	15.35	1.39	1.52	2.25	0	0.85	1.45
HMM.1	97.22	89.29	82.45	95.13	81.94	79.55	87.08	77	92.31	92.75
HMM.2	97.77	85.71	78.72	98.62	85.42	63.64	90.45	47.1	87.18	92.75
HMM.3	97.22	85.71	78.72	98.62	85.42	62.12	88.76	47.1	88.03	91.3
Bay.3	99.58	0	3.72	70.4	1.39	1.52	2.25	0	0.85	1.45
HMM.4	0	8.93	1.6	0	6.25	1.52	2.25	5.75	2.56	5.8

Bay.5	0.14	73.21	3.19	0	33.33	6.06	3.93	9.2	5.13	34.78
KB	56.47	83.93	84.57	99.36	81.25	59.85	71.91	74.7	88.89	92.75
Bay.4	89.29	75	50	95.5	70.14	43.94	83.71	21.8	90.6	85.51
Combined	96.94	92.86	82.45	98.71	88.19	66.67	90.45	51.7	90.6	92.75

As can be seen in Table 1, the performance of the individual classifiers varies widely. Classifiers that produce poor results for a given set of documents are given lower weights in the combined score. The weights are computed by examining results on the training set. These weights are then used on the classifiers when the classification subsystem is tested.

Figure 3 shows results based on the Reuters-21578 text categorization test collection. In this test, we used a different threshold for the minimum number of documents per category in the training sets under the Lewis-Split, which produced the category counts at various thresholds. For each threshold, the categories meeting the minimum number of training documents served to train the classifiers, with tests then conducted on the corresponding test sets. The classification committee consistently outperformed the individual classifiers.

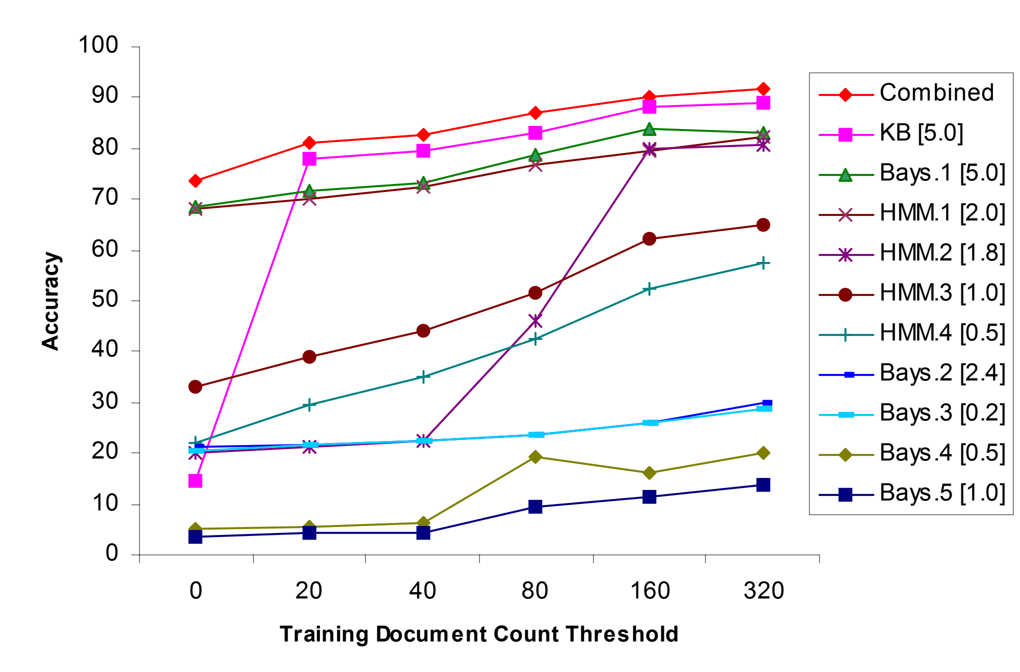


Figure 3: Accuracy rates for the individual classifiers on the Reuters set with various threshold values on the number of training documents required.

During this phase of operation, detailed information is available on a per-document basis. The document in the Reuters set labeled as 14859, reads:

AMATIL PROPOSES TWO-FOR-FIVE BONUS SHARE ISSUE
SYDNEY, April 8 - Amatil Ltd <AMAA.S> said it proposes to make a two-for-five bonus issue out of its revaluation reserve to shareholders registered May 26. Shareholders will be asked to approve the issue and an increase in authorised capital to 175 mln shares from 125 mln at a general meeting on May 1, it said in a statement. The new shares will rank for dividends declared after October 31. Amatil, in which B.A.T. Industries Plc <BTI.L> holds a 41 pct stake, said it does not expect to maintain its latest annual dividend rate of 29 cents a share on the enlarged capital.

All of the documents in the Lewis-Split have topic information associated with them. This particular document, like 75% of the Lewis-Split, is associated with a single category: earning. For this test, each document has been placed into a separate file.

Table 2: The per-category, per-classifier scores for a sample document along with the combined results.

```
/opt/shared/dox/reuters/limited_test/n_160/test/earn/reut.014.000858.sgm:
          bays1  bays2  bays3  bays4  bays5  hmm1   hmm2   hmm3   hmm4     kb  :  combined
     acq  0.335  1.000  1.000  0.000  0.080  1.000  0.579  0.696  0.300  0.230 :     0.632
    corn  0.016  0.000  0.000  1.000  0.000  0.000  0.027  0.059  0.940  0.153 :     0.035
   crude  0.062  0.078  0.056  0.921  0.000  0.167  0.194  0.317  0.680  0.076 :     0.245
    earn  1.000  0.315  0.281  0.684  0.919  0.655  1.000  1.000  0.000  1.000 :     1.000
   grain  0.020  0.026  0.019  0.973  0.000  0.108  0.149  0.257  0.740  0.384 :     0.191
interest  0.003  0.026  0.018  0.973  0.000  0.095  0.095  0.178  0.819  0.153 :     0.128
money-fx  0.024  0.078  0.054  0.921  0.000  0.118  0.179  0.297  0.700  0.230 :     0.224
    ship  0.047  0.026  0.017  0.973  0.000  0.038  0.000  0.000  1.000  0.230 :     0.000
   trade  0.000  0.052  0.034  0.947  0.000  0.029  0.135  0.238  0.760  0.230 :     0.168
   wheat  0.000  0.000  0.000  1.000  0.000  0.051  0.037  0.079  0.920  0.000 :     0.052
    earn   earn    acq    acq    acq   earn    acq   earn   earn   earn   earn :      earn
```

A result is considered to be correct if it matches any of the previously determined topics for a given document.

The first line is the name of the file being processed. The second line lists the names of the classifiers in the committee. The remaining lines give the name of the category and then the scores for each of the classifiers within the given category. The last line indicates the category predicted by the classifiers, with the first string representing the predetermined category and the last string giving the category determined by the committee. The guiding principles behind this format are ease of parsing and readability. When files are processed in batches, such as during the training and testing process, the entries are stored in a single file with blank lines separating the data.

Like SEEMs, the classifier committee is also modular. Classifiers can easily be configured to use stemming [15] and stop word lists. Generally speaking, however, these are enabled and disabled for all classifiers in the committee.

The string handling routines also contain a mechanism where certain words may be replaced with more generic versions. This list currently contains the following token or generalization: dollarAmountToken, percentageToken, floatingPointToken, monthToken, weekdayToken, yearNumberToken, timeToken, hrefToken, emailToken, largeAmountToken, numericToken, and nthToken.

This way the phrases “base run in the second inning” and “base run in the 7th inning” are both mapped to “base run in the nthToken inning.” Thus training documents can be made more general to help capture the essential elements of the category. This also means that words like “70%,” “1,000,” and “1997” can be used more effectively.

Since the committee is designed to use a collection of classifiers, it also can take advantage of external classification technology. A wrapper has been written so that the rainbow classifier, running in server mode, can also take part in the classification process [16]. Thus, even if SEE’s classification subsystem is not used, external classification technology can be seamlessly integrated into the system as a whole. This provides a way to leverage existing software infrastructures.

Semantic Annotation Example

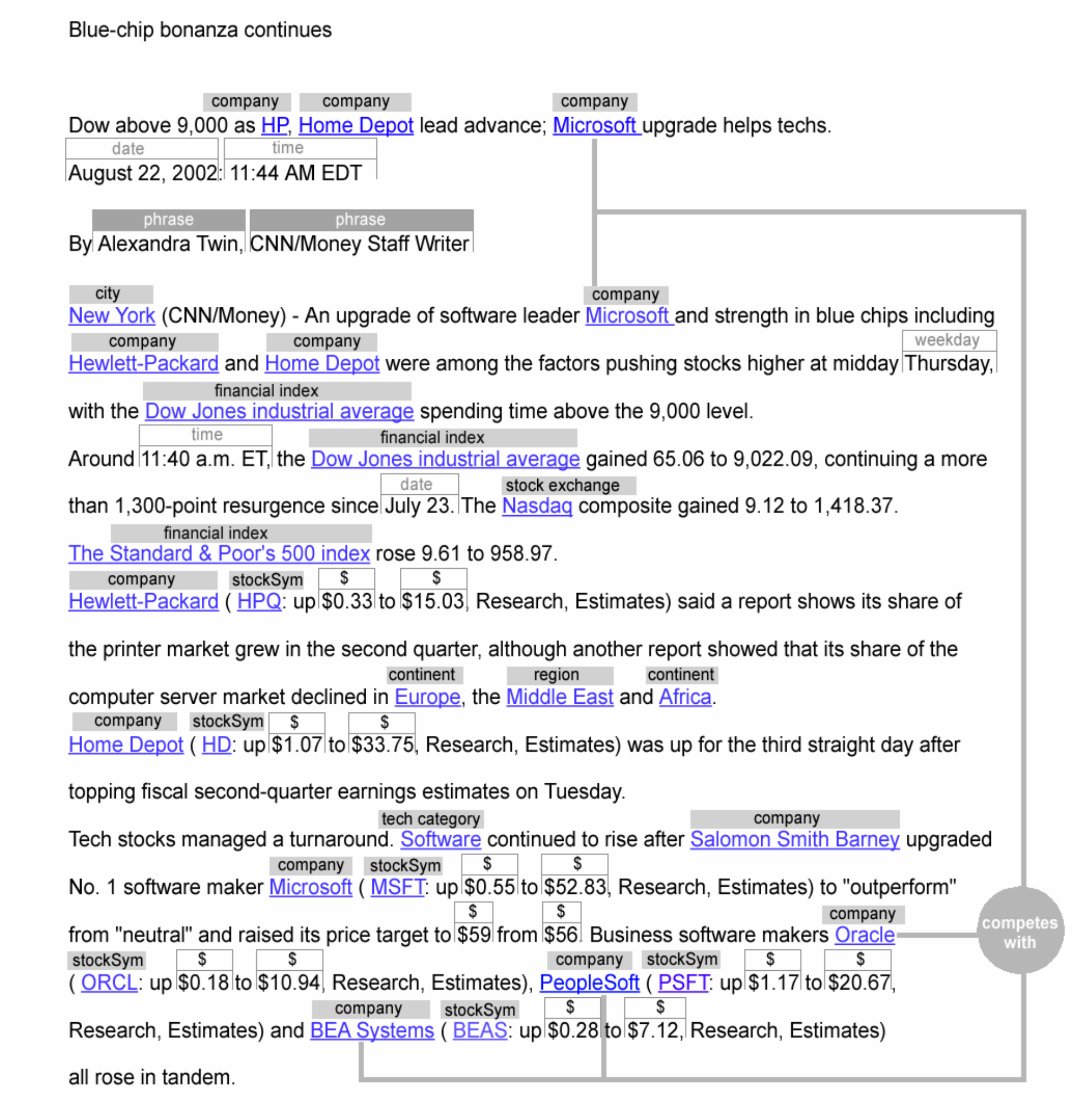
The document from which this snippet was taken originally appeared on the web at money.cnn.com on August 22, 2002. The names of the companies and their stock symbols were hyperlinked in the paragraph starting “Tech stocks managed.” SEE was not only able to uniquely identify entities, but also to find the relationships between them.

Here is the XML version of the enhanced text:

```
<EnhancedText>

<Classification domain=”Business”/>

Blue-chip bonanza continues

Dow above 9,000 as
    <Entity id="494805" class="company">HP</Entity>,
    <Entity id="501293" class="company">Home Depot</Entity>
lead advance;
    <Entity id="851165" class="company">Microsoft</Entity> upgrade helps techs.

<Regexp type="date">August 22, 2002</Regexp>:
<Regexp type="time">11:44 AM EDT</Regexp>

By
    <Phrase>Alexandra Twin</Phrase>,
    <Phrase>CNN/Money Staff Writer</Phrase>

<Entity id="4764" class="city">New York</Entity> (CNN/Money) - An upgrade
of software leader <Entity id="851165" class="company">Microsoft</Entity>
and strength in blue chips including
    <Entity id="494805" class="company">Hewlett-Packard</Entity> and
    <Entity id="501293" class="company">Home Depot</Entity> were
among the factors pushing stocks higher at midday
<Regexp type="weekday">Thursday</Regexp>, with the
<Entity id="720500" class="financialIndex">Dow Jones industrial average</Entity>
spending time above the 9,000 level.

Around <Regexp type="time">11:40 a.m. ET</Regexp>, the
<Entity id="720500" class="financialIndex">Dow Jones industrial average</Entity>
gained 65.06 to 9,022.09, continuing a more than 1,300-point resurgence since
<Regexp type="date">July 23</Regexp>.
The <Entity id="505367" class="stockExchange">Nasdaq</Entity> composite
gained 9.12 to 1,418.37. The
<Entity id="211452" class="financialIndex">Standard &amp; Poor's 500 index</Entity>
rose 9.61 to 958.97.

"Major indexes are up across the board," said
<Phrase>Peter Cardillo</Phrase>, director of
research at <Phrase>Global Partners Securities.</Phrase> "The
<Entity id="851165" class="company">Microsoft</Entity> upgrade certainly
helps, as does the absence of bad economic news. Also, oil prices are
retreating a little, the dollar is up and a little money is coming out
of Treasurys and into equities."

<Entity id="494805" class="company">Hewlett-Packard</Entity> (
    <Entity id="875349" class="tickerSymbol">HPQ</Entity>: up
    <Regexp type="money">$0.33</Regexp> to
    <Regexp type="money">$15.03</Regexp>,
    Research, Estimates
) said a
report shows its share of the printer market grew in the second quarter,
although another report showed that its share of the computer server
market declined
     in <Entity id="7852" class="continentRegion">Europe</Entity>,
    the <Entity id="7854" class="continentRegion">Middle East</Entity>
    and <Entity id="7848" class="continentRegion">Africa</Entity>.

<Entity id="501293" class="company">Home Depot</Entity> (
    <Entity id="511115" class="tickerSymbol">HD</Entity>: up
    <Regexp type="money">$1.07</Regexp> to
    <Regexp type="money">$33.75</Regexp>, Research, Estimates
) was up for the third straight day after topping fiscal second-quarter
earnings estimates on <Regexp type="weekday">Tuesday</Regexp>.

Tech stocks managed a turnaround.
    <Entity id="852744" class="techCategory">Software</Entity>
continued to rise after
    <Entity id="504523" class="company">Salomon Smith Barney</Entity>
upgraded No. 1 software maker
<Entity id="851165" class="company">Microsoft</Entity> (
    <Entity id="508968" class="tickerSymbol">MSFT</Entity>: up
    <Regexp type="money">$0.55</Regexp> to
    <Regexp type="money">$52.83</Regexp>, Research, Estimates
) to "outperform" from "neutral"
and raised its price target to
    <Regexp type="money">$59</Regexp> from
    <Regexp type="money">$56</Regexp>.

Business software makers
<Entity id="845065" class="company">Oracle</Entity> (
    <Entity id="508546" class="tickerSymbol">ORCL</Entity>: up
    <Regexp type="money">$0.18</Regexp> to
    <Regexp type="money">$10.94</Regexp>, Research, Estimates
),
<Entity id="496329" class="company">PeopleSoft</Entity> (
    <Entity id="508656" class="tickerSymbol">PSFT</Entity>: up
    <Regexp type="money">$1.17</Regexp> to
    <Regexp type="money">$20.67</Regexp>, Research, Estimates
) and
<Entity id="851574" class="company">BEA Systems</Entity> (
    <Entity id="508758" class="tickerSymbol">BEAS</Entity>: up
    <Regexp type="money">$0.28</Regexp> to
    <Regexp type="money">$7.12</Regexp>, Research, Estimates
) all rose in tandem.

</EnhancedText>
```

As can be seen in the XML above, each entity has a unique ID and a classification associated with it. Since the annotation is performed by a SEEM, the actual text of the tags can vary as needed. This can ease the integration of SEE into an existing system since it can be dropped in place and configured to yield custom input. In other situations, XSLT can be used to convert from one DTD to another or into HTML.

As shown in Figure 3, BEA Systems, Microsoft and PeopleSoft all engage in the "competes with" relationship with Oracle. When entities found within a document have relationships, we refer to the relationships as "direct relationships." Some of the direct relationships found in this example include: HPQ identifies Hewlett-Packard Co.; HD identifies The Home Depot; Inc.; MSFT identifies Microsoft Corp.; ORCL identifies Oracle Corp.; Salomon Smith Barney's headquarters is in New York City; and MSFT, ORCL, PSFT, BEAS are traded on Nasdaq.



Blue-chip bonanza continues

Dow above 9,000 as HP, Home Depot lead advance; Microsoft upgrade helps techs.

August 22, 2002: 11:44 AM EDT

By Alexandra Twin, CNN/Money Staff Writer

New York (CNN/Money) - An upgrade of software leader Microsoft and strength in blue chips including Hewlett-Packard and Home Depot were among the factors pushing stocks higher at midday Thursday, with the Dow Jones industrial average spending time above the 9,000 level.

Around 11:40 a.m. ET, the Dow Jones industrial average gained 65.06 to 9,022.09, continuing a more than 1,300-point resurgence since July 23. The Nasdaq composite gained 9.12 to 1,418.37. The Standard & Poor's 500 index rose 9.61 to 958.97.

Hewlett-Packard (HPQ: up $0.33 to $15.03, Research, Estimates) said a report shows its share of the printer market grew in the second quarter, although another report showed that its share of the computer server market declined in Europe, the Middle East and Africa.

Home Depot (HD: up $1.07 to $33.75, Research, Estimates) was up for the third straight day after topping fiscal second-quarter earnings estimates on Tuesday.

Tech stocks managed a turnaround. Software continued to rise after Salomon Smith Barney upgraded No. 1 software maker Microsoft (MSFT: up $0.55 to $52.83, Research, Estimates) to "outperform" from "neutral" and raised its price target to $59 from $56. Business software makers Oracle (ORCL: up $0.18 to $10.94, Research, Estimates), PeopleSoft (PSFT: up $1.17 to $20.67, Research, Estimates) and BEA Systems (BEAS: up $0.28 to $7.12, Research, Estimates) all rose in tandem.

Figure 3: When SEE recognizes an entity, knowledge about its entity classification and semantic associations becomes available.

Not all of the associated entities for an entity found in the text will appear in the document. Often, the entities mentioned will have one or more relationships with another common entity. In this case, some examples include: HPQ and HD are traded on the NYSE; BEAS, MSFT, ORCL and PSFT are components of the Nasdaq 100 Index; Hewlett-Packard and PeopleSoft invest in Marimba, Inc., which competes with Microsoft; BEA, Hewlett-Packard, Microsoft and PeopleSoft compete with IBM, Sun Microsystems and Apple Computer.

The use of semantic associations allows entities not explicitly mentioned in the text to be inferred or linked to a document. This one-step-removed linking is referred to as "indirect relationships." The relationships that are retained are application specific and are completely customizable. Additionally, it is possible to traverse relationship chains to more than one level. It is possible to limit the identification of relationships between entities within a document, within a corpus across documents or allow indirect relationships by freely relating an entity in a document with any known entity in the SCORE Knowledgebase.

Indirect relationships provide a mechanism for producing value-added semantic metadata. Each entity in the Knowledgebase provides an opportunity for rich semantic associations. As an example, consider the following:

Oracle Corp.

Sector:	Computer Software and Services
Industry:	Database and File Management Software
Symbol:	ORCL
CEO:	Ellison, Lawrence J.
CFO:	Henley, Jeffrey O.
Headquartered in:	RedWood City, California, USA
Manufactured by:	8i Standard Edition, Application Server, etc.
Subsidiary of:	Liberate Technologies and OracleMobile
Competes with:	Agile, Ariba, BEA Systems, Informix, IBM, Microsoft, PeopleSoft and Sybase

This represents only a small sample of the sort of knowledge in the SCORE Knowledgebase. Here, the ability to extract from disparate resources can be seen clearly. The "Redwood City" listed for the "Headquartered in" relationship above, has the relationship "located within" to "California," which has the same relationship to the "United States of America." Each of the entities related to "Oracle" are also related to other entities radiating outward. Each of the binary relationships has a defined *directionality (some may be bi-directional)*. In this example, *Manufactured by* and *Subsidiary of* are marked as *right-to-left* and should be interpreted as "8i Standard Edition, Application Server, etc. are *manufactured by* Oracle" and "Liberate Technologies and OracleMobile are *subsidiaries of* Oracle." SEE can use these relationships to put entities within context.

When a document mentions "Redwood City," SEE can add "California," "USA," "North America." Thus, when a user looks for stories that occur in the United States or California, a document containing "Redwood City" can be returned, even though the more generalized location is not explicitly mentioned. This is one of the capabilities a keyword-based search cannot provide. The interesting thing about this example is just how much information can be found in a typical document. By placing this information within context, the information implicit in the text is revealed and can then be linked with other sources of content.

9. Related Work

SEE shares objectives with projects such as CARMEN [17]. However, its document processing is significantly more comprehensive as it deals with heterogeneous types of data and broader analysis of text. Its metadata extraction capabilities are significantly more advanced than primarily statistical techniques reported in CARMEN [17].

Riloff and Jones [18] describe a technique of extracting metadata based on a bootstrapping technique in which an initial seed of entities is used to discover linguistic patterns or triggers that are, in turn, used to discover additional entities. The approach used by SEE's eMDExEnhancer is based instead upon the use of a relevant ontology (and more specifically the Knowledgebase of SCORE). Rather than trying to build patterns, internal and external knowledge sources are used to create a comprehensive list of all possible entities of interest in the domain. SEE's use of classification also allows for multiple domains to be fed off of the same Knowledgebase by targeting specific groups of entities based on the results of automatic classification. This means that SEE will be able to find entities that may not have appeared in the collection of training documents.

Nymble [19] uses a tagged corpus to train an HMM variant to automate data extraction. Like many information retrieval oriented systems, it identifies entities by type, but not *specific* entities. While such a system can discover "IBM" as a company in a document, it does not have the ability to make the connections to "Mainframes," "Microchannel Bus," and "IBM Global Services" that SEE provides (along with the ability to identify and annotate with relevant relationships if modeled as part of the ontology, specifically the WorldModel of SCORE). The ability to add information contextually by exploiting semantic relationships simply not available.

Systems that rely on discovering patterns in training text without canonical lists of entities [20] have achieved good results. However, they make the assumption that incoming document will closely match the wording and entity sets of the documents in the training collection. By leveraging the knowledge available from the World Wide Web, internal databases and human experts, this assumption can be avoided. Keeping the Knowledgebase up-to-date, maintaining both the pool of entities and their interrelations, allows SEE not only to identify the occurrence of entities, but to identify entities exactly: not just a TV show called "Cowboy Bebop," but *the* TV show "Cowboy Bebop." The correct identification of individual entities allows the power of semantic association to be brought to bear.

Semantic metadata annotation is an important area of focus in the Semantic Web research. SEE supports significantly more automation than the current semi-automatic annotation tools in academic research (such as OntoMat-Annotizer, [21]) or available as a product (such as Ontoannotate[2]).

Conclusions

SEE is able use its highly configurable architecture to support document processing at a per-document level for format parsing, word parsing, and phrase/sentence parsing. It also puts documents into context through automated classification methods. Once the domain of discourse is known, metadata extraction techniques can be customized on a per-category basis. This targeting effectively reduces the range of possible types of information that need to be analyzed for a given document. Knowing when to look for CEOs and when to look for baseball players improves accuracy. Matching regular expressions for one category only increases throughput. By restricting the scope of semantic metadata, SEE makes it possible to deal with the world one domain at a time. Through the identification of the unique entities in a document, SEE is able to harness the knowledge available from disparate sources and use it to add meaning and relevance to text. SEE is also unique in its ability to find contextually relevant relationships. The scope of relevant knowledge used and the documents across which relationships are identified can be controlled. We believe SEE takes advanced heterogeneous document processing to a semantic level with highly automated metadata extraction and semantic annotation capabilities. Semantic annotation is likely to be one of the first critical scalability challenge that the Semantic Web will face. Doing so automatically with high quality metadata is important, and this work presents a step in addressing that challenge.

[2] http://www.ontoprise.de/com/start_products.htm

Acknowledgements

We would like to thank our colleagues who helped make this work possible. Special thanks go to Clate Sander for assistance with presentation.

References

[1] N. Collier, "Machine Learning for Information from XML-markup on the Semantic Web," 2000.

[2] A. Maedche, S. Staab, "Mining Ontologies from Text," 12th International Workshop on Knowledge Engineering and Knowledge Management (EKAW 2000), October 2000.

[3] P. Clerkin, P. Cunningham, C. Hayes, "Ontology Discovery for the Semantic Web Using Hierarchical Clustering", Semantic Web Mining Workshop at ECML/PKDD-2001, September 3, 2001, Freiburg, Germany.

[4] A. Sheth, D. Avant, and C. Bertram, "System and Method for Creating Semantic Web and Its Applications in Browsing, Searching, Profiling, Personalization and Advertisement," U.S. Patent #6,311,194, 30 Oct. 2001.

[5] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke, "Semantic Content Management for Enterprises and the Web," IEEE Internet Computing, July/August 2002, pp. 80-87.

[6] Mark Wasson, "Classification Technology at LexisNexis," SIGIR 2001Workshop on Operational Text Classification

[7] C. Grover, C. Matheson, A. Mikheev, and M. Moens, "LT TTT - A Flexible Tokenisation Tool", in Proceedings of the Second Language Resources and Evaluation Conference, 31 May-2 June 2000, Athens, Greece.

[8] E. Brill, "Some Advances in Transformation-Based Part of Speech Tagging," National Conference on Artificial Intelligence, 1994.

[9] G. Salton, "Developments in Automatic Text Retrieval Science," Vol. 253, pages 974-979, 1991.

[10] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, 2002, pp. 1-47.

[11] C. Adams, "Word Wranglers: Automatic Classification Tools Transform Enterprise Documents from 'Bags of Words' to Knowledge Resources." Intelligent KM, January 2001.

[12] R. Liere and P. Tadepelli, "Active Learning with Committees for Text Categorization," *Proc. 14th Conf. Am. Assoc. Artificial Intelligence*, AAAI Press, Menlo Park, Calif., 1997, pp. 591-596.

[13] L.S. Larkey and W.B. Croft, "Combining classifiers in text categorization," Proceedings of SIGIR-96, 19-th ACM International Conference on Research and Development in Information Retrieval, 1996, pp.289-297.

[14] Z. Byoung-Tak and J. Je-Gun, "Building Optimal Committees of Genetic Programs," Parallel Problem Solving from Nature - PPSN VI 2000

[15] M. Porter, "An algorithm for suffix stripping," Technical Report 14, Program, 1980. http://www.muscat.co.uk/~martin/stem.html.

[16] A. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," http://www.cs.cmu.edu/mccallum/bow/, 1996.

[17] J. Krause and J. Marx, "Vocabulary Switching and Automatic Metadata Extraction or How to Get Useful Information from a Digital Library," Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries, Zurich, Switzerland, December, 2000.

[18] E. Riloff and R. Jones, “Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping”, Proceedings of the Sixteenth National Conference on Artificial Intelligence, 1999.

[19] D. Bikel, S. Miller, R. Shwartz and R. Weischedel, “Nymble: a high-performance learning name-finder,” Proceedings of ANLP-97.

[20] A. Mikheev, M. Moens and C. Grover. "Named Entity recognition without gazetteers," Proceedings of EACL, Bergen, Norway, 1999.

[21] S. Handschuh and S. Staab, *“*Authoring and Annotation of Web Pages in CREAM,” WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.