

# Technical Report: Semantic Annotation Platforms

Lawrence Reeve, Hyoil Han

College of Information Science and Technology

Drexel University

Philadelphia, PA 19104-2875, USA

Email: [lh24@drexel.edu](mailto:lh24@drexel.edu)  
[hhan@cis.drexel.edu](mailto:hhan@cis.drexel.edu)

December 2004

Abstract:

Semantic annotation is a key component for the realization of the Semantic Web. The volume of existing and new documents on the Web makes manual annotation problematic. Semi-automatic methods have been designed to alleviate the burden, and these methods have begun to be implemented with Semantic Annotation Platforms (SAPs). SAPs provide services that support annotation, including ontologies, knowledge bases, information extraction methods, APIs, and user interfaces. This chapter examines the considerations annotation systems must take into account, such document structure and initial ontology, and provides an overview of current SAP implementations. The Semantic Web also results in new as well as extended applications, such as concept searching, custom web page generation, question-answering systems as well as visualization. These applications are all made possible by the semi-automatic annotation services provided by SAPs.

Technical Report: Semantic Annotation Platforms .....	1
1. Introduction.....	4
2. Development of Semantic Annotation Platforms .....	6
2.1 Platform Considerations.....	10
2.1.1 Document Type.....	12
2.1.2 External Input.....	14
2.1.3 Extensibility .....	16
2.1.4 Information Extraction Tool Support.....	16
2.1.5 Initial Ontologies .....	18
2.2 Classification of Platforms.....	19
2.2.1 Pattern Discovery.....	21
2.2.2 Rules .....	22
2.2.3 Machine Learning .....	23
2.2.3.1 Probabilistic .....	23
2.2.3.2 Wrapper Induction .....	24
2.2.3 Multi-strategy.....	25
2.3 Platform Overviews .....	26
2.3.1 AeroDAML.....	28
2.3.2 Armadillo.....	29
2.3.3 KIM.....	30
2.3.4 MnM .....	32
2.3.5 MUSE .....	33

2.3.6 Ont-O-MAT using Amilcare.....	35
2.3.7 Ont-O-MAT using PANKOW.....	36
2.3.8 SemTag .....	37
4. Future Trends.....	38
4.1 Information Extraction Method Integration.....	39
4.1.1 Hidden Markov Models and Semantic Annotation .....	41
4.1.2 Integration of HMMs into Semantic Annotation Platforms .....	45
5. Conclusion .....	47
6. References.....	49

# 1. Introduction

The Semantic Web, as proposed by Tim Berners-Lee (1998), is the next generation of the Web providing machine-understandable information that is based on meaning. One way to provide meaning to Web information is by creating ontologies, and then linking information on a Web page to specifications contained in the ontology using a markup language (Berners-Lee, Hendler, & Lassila, 2001). Ontologies are conceptualizations of a domain that typically are represented using domain vocabulary (Chandrasekaran, Josephson, & Benjamins, 1999). Semantic annotation is the process of mapping instance data to an ontology. Benefits of adding meaning to the Web include: query processing using concept-searching rather than keyword-searching (Berners-Lee et al., 2001); custom Web page generation for the visually-impaired (Yesilada, Harper, Goble, & Stevens, 2004); using information in different contexts, depending on the needs and viewpoint of the user (Dill et al., 2003); and question-answering (Kogut & Holmes, 2001). These are all applications that use the document annotations produced during the semantic annotation process.

A key problem for the realization of the Semantic Web is providing document markup, or annotations, for both existing and new documents on the Web. Annotations are what provide the link between information stored within a document and the ontology (Berners-Lee et al., 2001). The purpose of semantic annotation is to tag ontology class instance data and map it into ontology classes. Semantic annotation can be seen as a typical information extraction for named-entity recognition process, but is different in that type information from a rich ontology is more specific and also the entities must be clearly identified and not just recognized as an entity of some type, as is the case with basic information

extraction efforts (Popov et al., 2003). Semantic annotation has been developed using research done in the areas of information extraction, information integration, wrapper induction, and machine learning (Dingli, Ciravegna, & Wilks, 2003).

Manual approaches were the initial approaches to annotation because automatic or semi-automatic systems were not yet available. Manual annotation can be done using tools such as Semantic Word (Tallis, 2003), which provides an environment for authoring as well as marking up documents from within a single interface. Manual approaches suffer from several drawbacks. Human annotators can provide unreliable annotation for many reasons: complex ontology schemas, unfamiliarity with subject material, and motivation, to name a few (Bayerl, Lungen, Gut, & Paul, 2003). It is expensive to have human annotators markup documents (Cimiano, Handschuh, & Staab, 2004). A human annotator may not consider using multiple ontologies (Dingli et al., 2003). Documents and ontologies can change, requiring new or modified markup, which leads to document markup maintenance issues (Dingli et al., 2003). Finally, the volume of existing of existing documents on the Web can lead to an overwhelming task for humans to manually complete (Kosala & Blockeel, 2000). For all these reasons, manual efforts have been identified as a “knowledge acquisition bottleneck” (Maedche & Staab, 2001).

To overcome the limitations of manual annotation, semi-automatic systems were developed. Semi-automatic systems, rather than completely automatic systems, are used because it is not yet possible to automatically identify and classify all entities within source documents with complete accuracy (Popov et al., 2003). Building a completely automatic annotation system is an open research problem. There are many advantages of semi-automatic annotation, such as providing document volume scalability by

reducing or potentially eliminating the human workload (Dill et al., 2003), and providing annotation services where the source document is not available for write access (Dill et al., 2003).

Semantic Annotation Platforms (SAPs) are systems for performing semi-automatic semantic annotation. SAPs vary in their architecture, information extraction tools and methods, initial ontology, amount of manual work required to perform annotation, performance and other supporting features, such as storage management of ontologies, knowledge bases, and annotations. Some SAPs were designed for a specific domain, but usually can be adapted to fit new domains. The SAPs presented in section 2 were chosen from a literature review of recent semantic annotation journal articles. The idea is to survey the supporting features, annotation methods used, and effectiveness of a set of SAPs that represent each classification from the SAP pattern-based and machine learning-based classifications. The work on SAPs is not yet complete. The section on futures discusses how additional work from the information extraction field might be used to improve annotation effectiveness.

## **2. Development of Semantic Annotation Platforms**

Semi-automatic annotation systems can be viewed as platforms, or infrastructure, for providing semantic annotation services (Popov et al., 2003). For example, in the Seeker platform, such facilities are described as Semantic Label Bureau, where the document annotations are stored separately from the source document, and applications make requests to the Semantic Label Bureau to find the annotations for a particular document and ontology (Dill et al., 2003). These semantic annotation platforms (SAP)

are composed of components that can often be interchangeably replaced. Figure 1 shows the general architecture of a SAP. The figure shows that semantic annotation platforms are generally architected as composable systems. Most SAPs are extensible, meaning various components can be replaced with alternate implementations. The advantage of a composable annotation platform is that it can be adapted to serve many needs, such as easily changing domains, languages, or providing scalability. For example, the work done for the Seeker platform (Dill et al., 2003) aims to provide large scale document annotation and annotation retrieval services on the Web.

Figure 1 shows the general abstraction layer of a SAP. The Application layer is responsible for providing an end-user interface to the annotation services provided by a SAP. Examples include facilities for annotating a document or document set and then potentially confirming the annotations before committing them, providing a query interface for searching annotations, and providing a user interface for configuring the information extraction component. The Upper Interfaces layer is primarily the application programming interface (API) layer. A set of programmatic interfaces are described in this layer. Applications call the defined APIs in order to perform actions on behalf of an application. The APIs can be quite numerous, covering annotation, information extraction, search, storage management, and many other provided services. The Upper Interface APIs are designed to shield the applications from changes in the Lower Interface. The Lower Interface contains the actual components that perform work for an application. The Upper Interface will remain consistent to an application, but the Lower Interface is expected to change based on the various components used. For example, the Information Extraction component may switch from a pattern-based tool to a statistical tool, and it is unlikely the programmatic interface is the same for both. The Upper Interface implementation will need to change to

accommodate the various Lower Interface components. Finally, the Storage Layer is designed to provide storage and storage management facilities for storing long-term data such as annotations and knowledge bases.

Examples of existing annotation platforms include AeroDAML (Kogut & Holmes, 2001), Armadillo (Dingli et al., 2003), MnM (Vargas-Vera et al., 2002), MUSE (Maynard, 2003), Ont-O-Mat (Handschuh, Staab, & Ciravogna, 2002), and SemTag/Seeker (Dill et al., 2003). The platforms are primarily distinguished by 1) the features offered, 2) the information extraction method used to find entities within documents, and 3) whether or not they are extensible. Features offered by SAPs include ontology and knowledgebase management (storage, editors), access APIs, annotation storage to allow multiple ontologies/annotations per document, and information extraction methods.

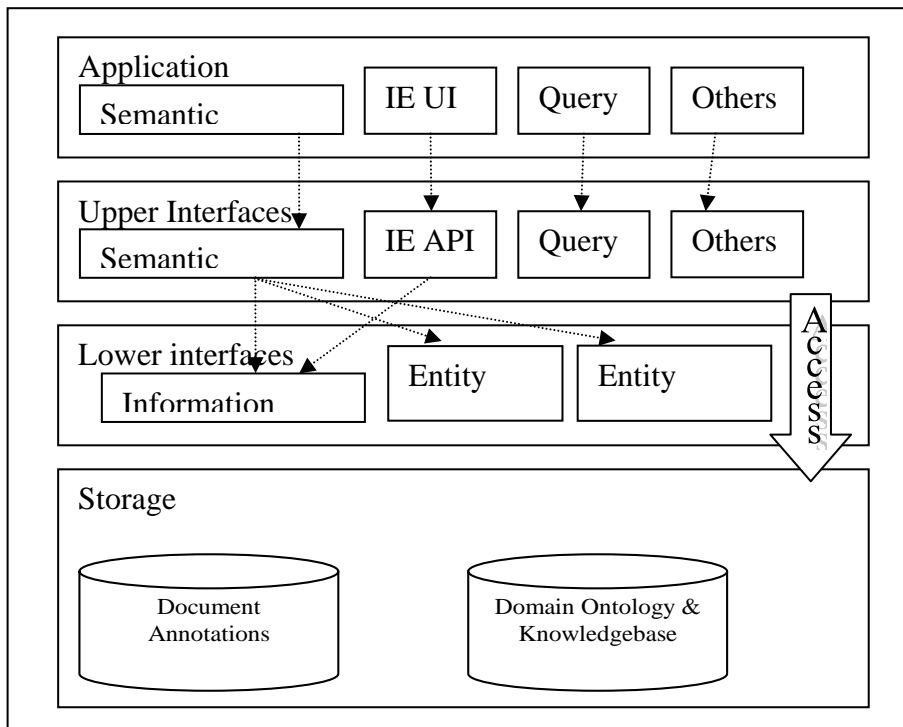


Figure 1: General architecture of a semantic annotation platform.

## 2.1 Platform Considerations

Figure 1 shows the various levels of abstraction in the design of a Semantic Annotation Platform (SAP). At each level, the implementer must make decisions which impact the performance and effectiveness of the level based on a set of design goals. These goals are considerations platforms must take into account. As shown in Table 1, several SAPs have been already been developed, but they are not implementation clones of one another. Instead, each SAP is designed to address a slightly different annotation need. Table 1 shows some key considerations for developing a SAP, listed by platform. The document type shows the type of input typically presented to the platform. Some platforms support many document types, while others only a single or several. The ML column indicates whether or not the platform uses machine learning. This has implications for manual effort required, for example, for developing training examples. The Method column indicates the type of annotation method used. Manual rules specifies whether or not manually generated rules are required. The External Input column gives some idea what type of external input is required, such as pre-annotated pages, gazetteer lists, and so forth. Since information extraction component is central to all platforms surveyed, the IE Tools lists the information extraction tool(s) used by each platform. These are usually pre-built IE toolkits from researches in the information extraction field. Finally, the Bootstrap Ontology column lists where the initial ontology is from. Semantic annotation, by definition, requires an ontology to map instances to. The initial ontology may be minimal, and may be added to and refined as part of the on-going annotation process, driven by user needs and new aspects of domains.

Table 1: Representative sample of semantic annotation platforms.

Platform	Document Type	Model	Method	Manual Rules	External Input	Extensible	IE Tools	Bootstrap Ontology
AeroDA ML	HTML	N	Rule	Y	Rule	N	AeroText	<b>WordNet;</b> <b>AeroText KB</b>
Armadillo	HTML	N	PD	Y	Seed	N	Amilcare ANNIE	<b>Address Book;</b> <b>Paper Citation</b>
KIM	HTML	N	PM	Y	Gazetteer KB population	N	GATE	<b>KIMO</b>
MnM	HTML, Plain Text	Y	WI using ML:LP <sup>2</sup>	N	Annotated corpus	Y	Amilcare	<b>KMi</b>
MUSE	Plain Text	N	Rule	Y	Gazetteer Rules	Y	GATE JAPE	<b>User constructed</b>
Ont-O- Mat:	HTML	Y	WI using	N	Annotated corpus	Y	Amilcare	<b>User constructed</b>

Amilcare			ML:LP 2					
Ont-O-Mat: PANKO W	HTML	N	PD	N	Web pages	Y	PANKO W	User constructed
SemTag	HTML	N	TLM	N	Taxonomy with labels	Y	Seeker platform	TAP with 72K labels

(HMM: Hidden Markov Model; JAPE: Java Annotations Pattern Engine; KB: Knowledge Base; ML: Machine Learning;

MUS: Multistrategy; PD: Pattern Discovery; PM: Pattern Matching; TLM: Taxonomy Label Matching; WI: Wrapper Induction)

### 2.1.1 Document Type

Most efforts at Semantic Web annotation are focused primarily on HTML because of its ubiquity, although other document formats such as XHTML, SGML, XML, RTF and others are also supported. HTML presents a challenging environment for entity extraction because HTML markup is designed for presentation and not data manipulation. While machines can effectively render the format, they cannot easily provide semantic interpretation of it. Also, most browser implementations do not strictly enforce

the markup requirements, such as requiring closing tags, and therefore complicate information extraction tasks such as identifying document structure from markup. General information extraction (IE) techniques can be applied to many different types of data. Applying IE to the goal of identifying entities and relationships in Web documents requires that IE systems be adapted to this environment. The format of document must be considered because each document type can potentially feed different information into an IE system. For example, XML documents are rich in schema description as compared to HTML, which consists mostly of presentation markup. Related to the document type is the structure of the document. Documents can be the result of a template generation from a database, and are therefore highly structured, they can contain some structure, or they can be completely unstructured by using primarily natural language text. The overall document structure can be important in determining what type of IE method will be most effective. Other extraction considerations are the amount of training required by an IE method to perform effectively.

Document structuring is classified as unstructured, semi-structured or structured. Unstructured documents consist of natural language text without any intended structure. Examples are magazine and journal articles. An IE using natural language processing techniques is needed to parse such documents. Structured and semi-structured documents are usually generated documents from back-end content management systems using templates (Mukherjee, Yang, & Ramakrishnan, 2003). Wrapper induction techniques can be used to deduce the location of entities within a document effectively if the structure is well-defined (Kushmerick, Weld, & Doorenbos, 1997).

The document structure classification is based on the primary type of structure. Documents can also be constructed using a combination of document structure types. For example, the Amazon product catalog ([www.amazon.com](http://www.amazon.com)) displays book product information that is composed of both structure data common to all books, such as title and author, as well as free-form text areas that are descriptions by the publishers and evaluators. A wrapper technique can be used to extract the common entities, and natural-language processing (NLP) techniques can be used to find entities in the free-form text. The approach used in the development of a SAP determines what domains the SAP can be used for. For example, the Armadillo (Dingli et al., 2003) platform is designed specifically for extracting web site data using linguistic patterns, while the KIM (Popov et al., 2003) platform has been designed to use NLP techniques to find entities.

### 2.1.2 External Input

All systems require some type of manual effort in order to begin the process of semantic annotation. Rules, gazetteers and knowledge bases are common methods of implementing semantic components. Rules can be specified using a grammar such as JAPE (Cunningham, Maynard, & Tablan, 2000), or can be a small set of initial set of seed patterns to facilitate pattern discovery (Brin, 1998). Gazetteers are lookup lists that map specific literal strings into a semantic concept. The string mappings are predefined by a user for a specific domain. Knowledge bases contain additional entity information than can be stored in the ontology alone. For example, the city of New York can have several literal expressions, such as {NYC}, {N.Y.C}, and {New York} (Kiryakov et al., 2003). Alternatively, knowledge bases can store a large number of additional entity details. For example, in the KIM platform which targets news

article annotation, the knowledgebase has been pre-populated with about 80,000 entities (Popov et al., 2003).

There exist several problems common to manual external input, such as manual rules, gazetteers, and knowledge bases. First, manual effort is still required to develop the external input to semantic components, although this effort is can be spread over many thousands of documents, which is still substantially less than annotating each document manually. Second, the rules and gazetteers must be changed to accommodate different domains and languages (Maynard, 2003). To overcome some of these problems, semantic annotation platforms often incorporate approaches to alleviate this manual bottleneck by automating the construction of the external input. For example, the KIM system incorporates a mechanism where the knowledgebase is continually expanded with new instance data, which is verified against a manually annotated corpus and a manually constructed smaller knowledgebase (Popov et al., 2003). To alleviate manual construction of rules, several systems use rule induction in order to automatically learn rules to process document text. For example, both the Ont-O-Mat (Handsuh et al., 2002) and MnM (Vargas-Vera et al., 2002) platforms use the LP<sup>2</sup> algorithm from the natural language processing community to learn rules. The algorithm is a wrapper induction algorithm that uses linguistic data rather than structural data, and is based on facilities within the Amilcare toolkit (Vargas-Vera et al., 2002). There has also been work done in the information extraction community using wrapper induction for structural data in order to extract entities from structured or semi-structured sources (Kushmerick et al., 1997). A drawback with the wrapper induction approaches, as with any machine learning approach, is that training samples must be provided. Interestingly, the

developers of the MUSE system report that they can move their rule-based system to different languages more quickly than machine-learning based systems because of the training problem (Maynard, 2003).

### 2.1.3 Extensibility

Extensible platforms allow for various components to be exchanged. For example, the lower interface of a SAP contains the critical components for semantic annotation performance. In an extensible SAP, a rule induction information extraction (IE) component can be substituted with a statistical one. The benefits of the rest of the platform components continued to be used while newer IE components are evaluated and integrated. Another advantage is allowing the SAP to adapt to different domains where the information extraction component may perform differently based on the domain document input, as demonstrated by the MUSE system (Maynard, 2003).

### 2.1.4 Information Extraction Tool Support

There are number of information extraction (IE) technique implementations available as a result of work done in the information extraction and natural language processing communities. Semantic annotation platforms usually take advantage of this work that in some cases has a long history. For example, the GATE language processing system has over eight years of development (Cunningham, Maynard, Bontcheva, & Tablan, 2002). In some cases, alternative approaches ignoring this work are used. The Seeker platform uses its own taxonomy populated with 72,000 entity labels that it searches for in the document and then tags when the label is found in the document (Dill et al., 2003). Examples of annotation platforms using existing IE toolkits are Amilcare used in S-CREAM (Handschuh et al., 2002)

and MnM (Vargas-Vera et al., 2002), and GATE used in KIM (Popov et al., 2003), MUSE (Maynard, 2003), and Armadillo (Dingli et al., 2003).

The most common toolkits are GATE and Amilcare. GATE is produced by the University of Sheffield's Natural Language Processing Group (University of Sheffield, 2004) and is a language process environment for building human language processing systems. GATE is divided into three parts: 1) an architecture which defines a goal component for componentizing a language processing system; 2) a concrete framework implementation; and 3) a graphical development environment. GATE includes an information extraction system called ANNIE (A Nearly-New IE system) that is composed of a tokenizer, gazetteer, sentence splitter, part-of-speech tagger, semantic tagger, coreferencing (OrthoMatcher) and other related components (University of Sheffield, 2004). Since GATE contains a vast number of components, it is typical that only certain parts of GATE may be used in semantic annotation platform implementations. For example, Armadillo uses ANNIE's named entity recognizer (Dingli et al., 2003), MUSE (Maynard, 2003) and KIM (Popov et al., 2003) use the tokenizer, sentence splitter, part-of-speech tagger, gazetteer, semantic tagger, and coreferencing.

Amilcare is produced by the University of Sheffield's Computer Science Department and is specifically designed to perform semantic annotation (University of Sheffield, 2002). Amilcare is a rule-based system. The rules are induced with a learning algorithm that uses a training corpus that has been annotated with XML tags. Amilcare has three modes of operation: training, test, and production. The training mode is used to induce rules from the training corpus. The test mode uses the induced rules from the training mode on a separate tagged corpus in order to evaluate the performance of the learned

rules. In test mode, the tags are first removed and the corpus is annotated using the learned rules. A comparison is then automatically run against the original tagged corpus and the corpus just tagged. The production mode annotates a corpus, and incorporates a user-feedback cycle to learn annotation corrections. Amilcare uses the LP<sup>2</sup> algorithm, which induces two types of rules: 1) rules for annotating text, and 2) rules that correct mistakes generated by the first type of rules. Interestingly, Gate is used within Amilcare for tokenization, sentence identification, part of speech tagging, gazetteer lookup and named entity recognition (University of Sheffield, 2002).

### 2.1.5 Initial Ontologies

Semantic annotation requires an ontology in order to perform concept instance mapping. Ontologies are usually architected using levels, such as upper and lower. The upper ontology consists of general concepts, while the lower ontology has a deeper specialization of the upper ontology concepts (Missikoff, Navigli, & Velardi, 2002). Some semantic annotation platforms place the responsibility on the user for constructing an initial ontology. Examples include MUSE (Maynard, 2003) and Ont-O-Mat (Handschuh et al., 2002). Other platforms provide an initial ontology as part of their development. The KIM platform provides an ontology called KIMO that is designed to provide a minimal open-domain ontology, and is based on OpenCyc, WordNet, DOLCHE and other upper-level resources (Popov et al., 2003). KIMO is composed of approximately 250 classes and 100 attributes and relations, and the specialization of classes is derived from an analysis of a corpus of general news (Popov et al., 2003). The Seeker platform uses TAP, which is a shallow knowledgebase that contains information about a broad range of popular culture subjects, such as movies, sports, and so forth (Dill et al., 2003). The TAP

knowledgebase has about 72,000 labels that are used to tag instances found in documents. The MnM platform uses a hand-crafted ontology called KMi (Knowledge Management Institute) (Vargas-Vera et al., 2002). The AeroDAML platform uses the commercial product Aderotext, and utilizes an upper-level ontology based on Wordnet, while the lower-level ontology uses the common knowledgebase of AeroText (Kogut & Holmes, 2001). Armadillo provides an example of a platform where the initial ontology is very light weight, consisting of an address-book type of ontology where members of a computer science department are discovered and populate address information, such as name, phone number, address, and so forth (Dingli et al., 2003).

## 2.2 Classification of Platforms

Current semantic annotation platforms use several methods for information extraction (IE) from Web documents. Figure 1 shows a hierarchical classification of annotation platforms, and this classification can be used to organize the platforms performing semantic annotation. While semantic annotation platforms have many aspects, the information extraction approach currently used to find entities within text has the most impact on the effectiveness of the platform. For this reason, the IE approach of each platform is used to organize the platforms. Table 1 in the previous section lists several semantic annotation platforms and the information extraction method used by each. As semantic platforms develop, it is anticipated that the classification structure will adapt to newer approaches as well.

The top-level approach is multi-strategy, which uses a combination of the lower level approaches. A platform using a multi-strategy approach is able to adapt its IE methods based on the text it is processing

in order to obtain the best results. The multi-strategy approach uses a high-level identification of text genre, and then executes the appropriate IE methods. This is in contrast to lower-level approaches using newer IE algorithms such as LP<sup>2</sup> (Ciravegna, 2001) in platforms such as KIM (Popov et al., 2003), which are able to use machine-learning to perform rule induction using both structural and linguistic information. No semantic annotation platform to date is using a complete multi-strategy approach incorporating both pattern and machine-learning approaches. The MUSE system comes the closest by using text features and then conditionally executing rules based on the text features (Maynard, 2003).

The two primary lower levels are pattern-based methods and machine-learning methods. Pattern-based methods are systems composed of manual rules. The rules are typically hand-crafted rules that define how entities can be found in text (Maynard, 2003). Examples of such systems are AeroDAML (Kogut & Holmes, 2001), MUSE (Maynard, 2003) and SemTag/Seeker (Dill et al., 2003). A limiting factor on the scalability of such systems is that the manual rule generation process can be maintenance intensive. Each time a data source changes, the pre-defined rules may also need to be changed. Machine-learning approaches use pre-annotated examples to learn how to identify entities. Rules are learned automatically, and this type of rule learning is currently used in platforms using the Amilcare toolkit (University of Sheffield, 2002), which implements the LP<sup>2</sup> algorithm (Ciravegna, 2001). Examples of systems using rule learning are Ont-O-Mat (Handschuh et al., 2002) and MnM (Vargas-Vera et al., 2002). Hidden Markov Model (HMMs) is an example of another machine-learning approach that can also be used. HMMs are not currently being used by any of the semantic annotation platforms, and in the last part of this chapter, the use of HMMs for semantic annotation is discussed.

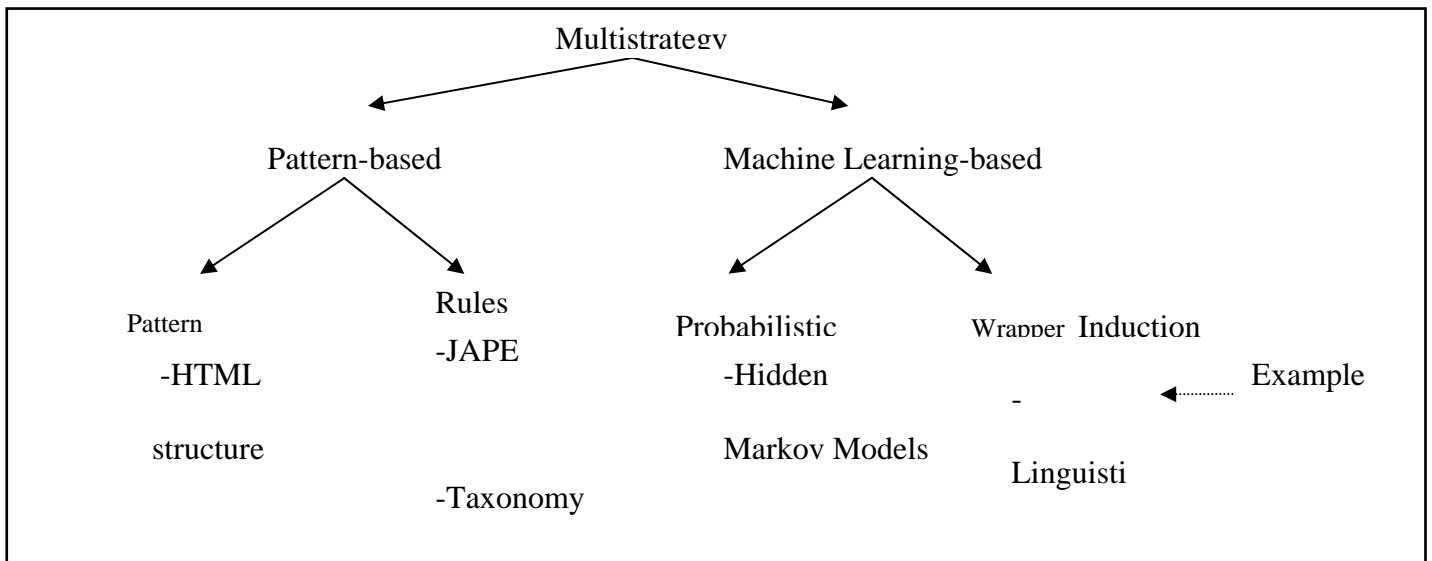


Figure 2: Classification of Semantic Annotation Platforms

### 2.2.1 Pattern Discovery

Patterns are also a widely-used technique in semantic annotation platforms. Pattern discovery works by taking a few seed samples, finding entities based on the patterns, expanding the seed samples with

patterns from the new entities found, and repeating the process until no more instances are found, or the user stops the iterative process (Brin, 1998). Patterns can exploit known linguistic patterns, such as Hearst patterns (Hearst, 1992), to find entities, as is done in the Ont-O-Mat using PANKOW platform (Cimiano et al., 2004). The Ont-O-Mat platform has been updated to replace the Amilcare component with a pattern-based component, called PANKOW. The Ont-O-Mat platform (Handsuh et al., 2002) demonstrates the usefulness of an extensible semantic annotation platform, where components can be replaced without losing or duplicating the features already available (Cimiano et al., 2004). The Armadillo platform is also example of a platform that uses a small set of initial seeds to begin a pattern discovery process.

### 2.2.2 Rules

Rules can be manually-generated, as MUSE (Maynard, 2003) does with the JAPE grammar (Cunningham et al., 2000), or they can be generated with machine-learning techniques. In this case, rules are considered the rules that are a subset of the pattern classification in Figure 2. The difference is the rules are initially manually specified by the user. Rules can take many forms. In the Seeker platform, the rules are labels (Dill et al., 2003). SemTag, the semantic tagging component of Seeker, uses the labels stored in the knowledgebase to find instances of ontology concepts, and then uses statistical likelihood to determine where in the ontology tree an instance is most likely to be contained (Dill et al., 2003). The process is not completely automatic, however, as an initial set of training data must be provided. The authors report 700 entries as the initial size of the training set (Dill et al., 2003). In the MUSE platform, rules are written using the JAPE grammar (Maynard, 2003). The MUSE platform is an

interesting rule-based system because it can adapt the rules used in entity identification depending on text attributes, such as language, document type, document source, and so forth (Maynard, 2003). The result is a rule-based platform that performs competitively with machine-learning based platforms (Maynard, 2003).

### 2.2.3 Machine Learning

Platforms based on machine learning are divided using the machine-learning method, which are currently two approaches: probabilistic and wrapper induction. The more common of the two approaches is wrapper induction. It is anticipated that future research in semantic annotation will take advantage of machine-learning based approaches, because they help to relieve the manual effort required in building rules, which is the primary drawback of pattern-based approaches. For example, the Rainbow system is a domain-specific annotation system developed for aggregating product information from multiple web sites, and then providing search over the aggregated information (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004).

#### 2.2.3.1 Probabilistic

Probabilistic methods use algorithms such as Hidden Markov Models (HMMs) to perform information extraction. For example, the DATAMOLD tool (Borkar, Deshmukhy, & Sarawagiz, 2001) uses HMMs for information extraction, but it has not yet been integrated into a semantic annotation platform. The Seeker platform uses a probabilistic approach to help eliminate mis-classification caused by its simple

tagging approach within the SemTag component (Dill et al., 2003). SemTag has a pool of approximately 72,000 labels that it uses to find entity instances within text. It is likely that some of the labels will be duplicated but contained in different parts of the ontology. SemTag incorporates an algorithm as part of its tagging process to determine the probability of a particular label belonging to a particular class in the ontology. In the Rainbow project, where product catalog information from a specific domain is extracted from multiple sites, the information extraction portion project uses Hidden Markov Models as its primary method. While the performance is competitive with other approaches, the resulting system developed from Rainbow has not yet been incorporated into a general purpose semantic annotation platform. The most likely reason is that the generated model is domain-specific (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004).

#### 2.2.3.2 Wrapper Induction

A wrapper is a function from a page to the set of tuples it contains (Kushmerick et al., 1997). Wrappers are used when there exists a repeatable structure to extract information from. Many web sites have pages generated using from a back-end database, and the pages generated follow a common template. The purpose of wrappers is to reverse the page generation process in order to retrieve the original database tuples. Wrappers can be hand-crafted, or they can be learned. Manual wrappers require the user to mark areas of interest within a document. The machine can then extract entities from documents with a similar structural format as the manually marked-up document (Vargas-Vera et al., 2002). Kushmerick (1997) defined a method for performing wrapper induction, where the wrappers are automatically learned from

example query responses from a data source. Wrappers are most effective when the data is presented in a structured format, such as product catalogs (Dingli et al., 2003).

Wrappers can also be linguistic-based, where the wrapper induction process discovers linguistic rules for identifying entities (Vargas-Vera et al., 2002). Amilcare implements the LP<sup>2</sup> algorithm (Ciravegna, 2001), which performs rule induction using both linguistic and structural information (Ciravegna, 2001). It is intended to provide a combination of IE approaches such as wrapper induction (Kushmerick et al., 1997) and linguistic approaches from the natural language processing community. The Amilcare toolkit is used by several platforms, such as MnM (Vargas-Vera et al., 2002), and Ont-O-Mat (Handschuh et al., 2002).

### 2.2.3 Multi-strategy

A multi-strategy approach uses a combination of both machine-learning and pattern-based approaches. No surveyed SAP uses the multi-strategy approach. However, this approach could be used by a platform which adapts its processing based on the document genre or specific text features. Such adaptive processing would apply the most effective processing for each type of document. The closest SAP that performs adaptive processing is the MUSE system (Maynard, 2003). MUSE uses a pipeline approach to identify text features that can be used to perform conditional processing. MUSE uses the JAPE grammar (Cunningham et al., 2000) to define rules to perform semantic annotation, and the JAPE rules are fire conditionally, based on text features. The conditional processing approach used by MUSE makes it competitive with machine-learning based approaches (Maynard, 2003).

### 2.3 Platform Overviews

There have been many efforts to build platforms for semantic annotation. Platforms provide support not only for IE implementations, but also ontology and knowledgebase management, and likely other facilities, such as access APIs, storage (such as RDF repositories), and user-interfaces for front-ends such as ontology and knowledgebase editors (Popov et al., 2003). Not all platforms support all features. Some of the platforms presented here are focused on a single document type or domain and are therefore not as feature-rich as more multi-purpose annotation frameworks. Also, some platforms are considered extensible if various components can be replaced. For example, the gazetteer can be replaced with a custom implementation. Platforms are distinguished from pure semantic annotation algorithms, which usually are implemented within a platform, and do not provide a wider range of annotation services.

**Table 2 shows the author-reported performance of various platforms, with the exception of AeroDAML, Ont-O-Mat using Amilcare and SemTag, whose authors did not provide complete performance information. The standard measures of Precision, Recall, and F-measure, taken from the information retrieval field, were used by the remaining SAP authors in determining annotation effectiveness. In the general definition of recall and precision shown below, “accurate” and “inaccurate” refer to annotations generated semi-automatically by a SAP, while “all” refers to all annotations generated by a human annotator.**

$$\textit{Annotation Recall} = \frac{\textit{accurate}}{\textit{all}}$$

$$\textit{Annotation Precision} = \frac{\textit{accurate}}{\textit{accurate} + \textit{inaccurate}}$$

**F-measure is the harmonic mean of precision and recall. The highest performing machine learning-based platform is MnM. For pattern-based platforms, MUSE is best. The worst performing is Ont-O-Mat using PANKOW. PANKOW is a recent effort to use unsupervised learning in a pattern-based system, and performance improvements are expected as the system develops further (Cimiano et al., 2004).**

Table 2: Semantic annotation platform information extraction methods and performance measurements.

Platform	IE Method	Precision	Recall	F-Measure
Armadillo	Pattern Discovery	91	74	87
KIM	Manual Rules	86	82	84
MnM	Wrapper Induction	95	90	n/a
MUSE	Manual Rules	93	92	93
Ont-O-Mat using PANKOW	Pattern Discovery	65	28	25
SemTag	Semi-automatic Rules	82	n/a	n/a

### 2.3.1 AeroDAML

AeroDAML (Kogut & Holmes, 2001) is designed to map proper nouns and common relationships to corresponding classes and properties in the DARPA Agent Markup Language (DAML) ontologies. AeroDAML uses the AeroText for its information extraction component. The AeroText Java API is used to access extracted parts and map them into RDF triples using the user-specified ontology as a guide. The default ontology consists of two parts. The upper level ontology uses the WordNet noun synset hierarchy, while the lower level ontology uses the knowledgebase provided by AeroText (Kogut & Holmes, 2001).

AeroDAML operates in two modes depending on the version used. The web-based system allows users to enter a URI and have the DAML-based annotation of the page returned using the default ontology. The client-server version allows users to enter a filename and returns the DAML-based annotation of the text returned using a custom ontology.

The AeroText system on which AeroDAML is based provides an integrated environment to perform information extraction. It consists of four main components: 1) Knowledge Base (KB) compiler for transforming linguistic data into a run-time knowledge base; 2) Knowledge Base Engine for applying the KB to source documents; 3) an IDE for building and testing KBs, and 4) Common Knowledge Base containing domain independent rules for extracting proper nouns and relations (Kogut & Holmes, 2001).

### 2.3.2 Armadillo

Armadillo (Dingli et al., 2003) is used to mine web sites that have a highly-regular structure. Armadillo uses the Amilcare information extraction system to perform wrapper induction on web pages. Armadillo uses a pattern-based approach to find entities, but is unique in that it finds its own initial set of seed-patterns, rather than requiring an initial set of seeds, as described in (Brin, 1998). Manual patterns are used for the named entity recognizer. No manual annotation of corpus documents is required. Once the seeds are found, pattern expansion is then used to discover additional entities. The idea of information redundancy is exploited to verify entities that have been extracted. This is accomplished by querying various web services, such as Google and CiteSeer, and examining the query results to confirm or deny

the existence of an entity. A similar approach is used in the PANKOW algorithm (Cimiano et al., 2004) to disambiguate concepts.

The use case examined is extracting worker details from a university computer science department web site in order to find personal data, such as name, position, home page, email address, and other contact information. The seed-discovery and expansion finds worker names in the web pages. Since many names may be discovered, the Web services are queried to confirm a person actually works in the department. The names are then used to discover home pages, where detailed information about a person can often be found and extracted. Armadillo is also interesting in that it attempts to discover bibliographic citations for each person discovered. Bibliographic extraction is a more difficult task for several reasons: patterns are difficult to write for paper titles because they are difficult to characterize; authors may be editors and not actually a title's author; and papers may be co-authored. The information redundancy approach was also applied to bibliographic entries, but with a lower success rate than discovering and extracting information about people from home pages (Dingli et al., 2003).

### 2.3.3 KIM

The Knowledge and Information Management (KIM) platform (Popov et al., 2003) contains an ontology, knowledge base, a semantic annotation, indexing and retrieval server, as well as front-ends for interfacing with the server. For ontology and knowledge base storage it uses the SESAME RDF repository, and for search it uses a modified version of the Lucene (Cutting, 2004) search engine. The semantic annotation process relies on a pre-built lightweight ontology called KIMO as well as an inter-

domain knowledgebase. KIMO defines a base set of entity classes, relationships, and attribute restrictions. The knowledge base is populated with 80,000 entities consisting of locations and organizations, gathered from a general news corpus. Named-entities found during the annotation process are matched to their type in the ontology and also to a reference in the knowledgebase. The dual mapping allows the information extraction process to be improved by providing disambiguation clues based on attributes and relations.

The information extraction component of semantic annotation is performed using components of the GATE (Cunningham et al., 2002) toolkit. GATE provides IE implementations of tokenizers, part-of-speech taggers, gazetteers, pattern-matching grammars (JAPE), and coreference resolution (Popov et al., 2003). Some components of GATE have been modified to support the KIM server. For example, pattern-matching grammar rules are based on ontology classes rather than simple types (Popov et al., 2003). Other components of semantic annotation have been custom developed. The gazetteer, for example, performs entity alias lookups using the knowledgebase. The flow of text through the semantic annotation process is shown in Figure 3. The MUSE system, described later, uses a similar flow, but modifies the flow based on attributes within the text. MUSE also uses the GATE toolkit.

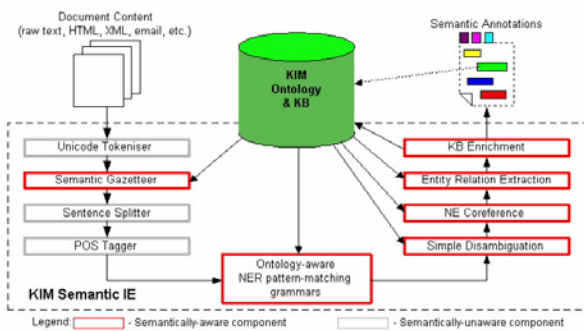


Figure 3: Semantic Annotation flow in KIM (Popov et al., 2003).

#### 2.3.4 MnM

MnM (Vargas-Vera et al., 2002) provides an environment to manually annotate a training corpus, and then feed the corpus into a wrapper induction system based on the Lazy-NLP algorithm. The resulting output is a library of induced rules that can be used to extract information from corpus texts. Lazy-NLP systems are based on linguistic information, and rules are generated using sets of conjunctive conditions on adjacent words (Vargas-Vera et al., 2002). Once the system has been trained and rules have been induced from a training corpus, the system can begin to extract information from source documents and return annotated documents based on user-defined semantic tags. Once a document has been annotated, it is presented to the user for approval. Upon user acceptance, the annotated document is sent to the ontology server in order to populate the ontology with instance data.

The rule induction process generates two types of rules: tagging and correction. A rule is composed of a pattern of conditions on a connected sequence of words, followed by an action performed when the pattern is found to match a part of text (Vargas-Vera et al., 2002). For tagging rules, the action performed is the insertion of a semantic tag into the text. For correction rules, the action performed is the insertion of information that causes semantic tags to shift location, based on training information. The corrective tags are inserted during the training period, when the training corpus is re-annotated using the induced rules. If an induced tagging rule is found to be incorrect in its location, it is not replaced, but corrected using a correction rule.

Populating the ontology with instance data is done by taking the data from the semantic annotations and populating each ontology class with its attribute values. The population phase will attempt to provide values for as many class attributes as possible. It is sometimes the case not enough information from the semantic annotation process will be available. This will occur if the attribute value is not mentioned in the text, or if the annotation rule set is incomplete and needs more training (Vargas-Vera et al., 2002). In such cases, the user will need to manually provide attribute values.

### 2.3.5 MUSE

MUSE (Maynard, 2003) was designed to perform named entity recognition and coreferencing. It is implemented using the GATE (Cunningham et al., 2002) framework. MUSE calls the information extraction components *processing resources* (PRs). The PRs form a processing pipeline used to discover named entities. While some annotation systems execute processing resources sequentially, one after the other, MUSE executes PRs conditionally, based on text attributes. For example, source documents from different domains or different languages can use different subsets of processing resources. The conditional processing is handled using a Switching Controller, which calls the appropriate processing resource in the specified order. The use of conditional processing allows MUSE to obtain accuracies similar to machine learning systems (Maynard, 2003). Semantic tagging is accomplished using the Java Annotations Pattern Engine, also known as JAPE (Cunningham et al., 2000) . Rules using the JAPE grammar are constructed to generate annotations. The Semantic Tagger can use tags generated by processing resources run earlier in the pipeline. For example, if the gazetteer recognizes a first name and

the part-of-speech tagger recognizes a proper noun, a JAPE rule can use both tags to annotate an entity of type Person (Maynard, 2003). The MUSE system is more sophisticated than a simple gazetteer. A gazetteer cannot provide complete named-entity recognition: 1) it is difficult to provide an exhaustive list of all potential named-entities, and 2) discovered names can be ambiguous (e.g., Washington can be a city or a person). The Semantic Tagger component can take advantage of both the gazetteer and the part-of-speech tagger, for example, to resolve ambiguity.

MUSE is also able to perform coreferencing between named entities. For example, Bill Smith, William Smith, and Mr. Smith can all be recognized as the same entity. The Orthomatcher module is responsible for performing the coreferencing. The Orthomatcher is completely rule driven using manually-generated rules. The rules can be conditionally applied based on the entity type.

MUSE has been used to perform annotation in languages other than English to examine its adaptability and compare its performance with machine learning systems, which typically require large training data. Since MUSE is modular, only the language-dependent parts needed to be converted between languages. For example, the part-of-speech tagger, gazetteer lists, and potentially language-dependent parts of the semantic tagger may need to be modified. The multiple language adaptation projects demonstrated that MUSE provides an advantage over machine learning based approaches because it requires a smaller amount of training data (Maynard, 2003).

### 2.3.6 Ont-O-MAT using Amilcare

Ont-O-Mat is an implementation of the S-CREAM (Semi-automatic CREAtion of Metadata) semantic annotation framework (Handschuh et al., 2002). The information extraction component is based on Amilcare. Amilcare is machine-learning based and requires a training corpus of manually annotated documents. Amilcare uses the ANNIE ("A Nearly-New IE system") part of the GATE toolkit to perform information extraction tasks such as tokenization, part-of-speech tagging, gazetteer lookup, and named-entity recognition (Handschuh et al., 2002). The result of ANNIE processing is passed to Amilcare, which then induces rules for information extraction using a variant of the LP<sup>2</sup> algorithm. The wrapper induction process uses linguistic information, and is the same Amilcare wrapper induction process as MnM (Vargas-Vera et al., 2002), generating tagging and correction rules.

Ont-O-Mat with Amilcare also tackles the issue of discourse representation. The idea is to provide unity of concept between sentences. An entity class may not receive all of its values and relationship information from a single sentence. Discourse representation allows class values to be completed as the text is analyzed. New sentences can provide additional information, such as 1) new entities, 2) new relationships, and 3) denotation equivalence between different entities (Handschuh et al., 2002). Discourse representation is accomplished using a template-based approach for simple discourse representation. A single tag is used to introduce a new entity, and every other tag in the template refers to an attribute value. For more complex discourse representation, rules can be provided (Handschuh et al., 2002).

### 2.3.7 Ont-O-MAT using PANKOW

Ont-O-Mat provides an extensible architecture that allows replacement of selected components. The original Ont-O-Mat implementation was done using Amilcare. In this case, Ont-O-Mat replaces the annotation portion with a implementation of the PANKOW (Pattern-based Annotation through Knowledge On the Web) algorithm (Cimiano et al., 2004). The PANKOW process takes proper nouns from the information extraction phase and generates hypothesis phrases based on linguistic patterns and the specified ontology. For example, a sports ontology may generate hypothesis phrases from the proper noun “Pete Rose” using patterns such as “Pete Rose *is a* Player” and “Pete Rose *is a* Team,” where “Player” and “Team” are ontology concepts. The hypothesis phrases are then presented to the Google web service. The phrase with the highest query result count is then used to annotate the text with the appropriate concept. The core principle is called “disambiguation by maximal evidence” (Cimiano et al., 2004). This principle is similar to the approach used by Armadillo (Dingli et al., 2003), which used multiple web services to find evidence.

The linguistic patterns used to generate hypothesis phrases have been previously identified in the literature. For example, Hearst Patterns provide an initial set of 4 patterns. Definites (“the <INSTANCE><CONCEPT>” as well as “the <CONCEPT><INSTANCE>”), apposition (<INSTANCE>, a <CONCEPT>) and copula (<INSTANCE> is a <CONCEPT>) are also patterns used to generate hypothesis phrases.

To recognize hypothesis phrases, the World Wide Web is used as the corpus because of its size and the likelihood of finding significant numbers of hypothesis phrase matches. The number of web pages

containing the hypothesis phrase are counted, and the count is used to indicate the strength of a particular hypothesis phrase. The content of the individual web pages is ignored and plays no role in determining the validity of a hypothesis phrase.

### 2.3.8 SemTag

SemTag is the semantic annotation component of Seeker, a comprehensive platform for performing large-scale annotation of web pages (Dill et al., 2003). SemTag performs annotation in passes: Spotting, Learning, and Tagging. The Spotting pass examines tokenized words from source documents and finds label matches from the taxonomy. If a label match is found, a window of ten words to either side of the source document match is kept. In the Learning pass, a sample of the corpus is examined to find the corpus-wide distribution of terms at each node of the taxonomy. The Tagging pass is then executed, scanning all of the windows from the Spotting pass and disambiguating the matches. Once a match is confirmed, the URL, text reference, and other metadata are stored. SemTag/Seeker is an extensible system, so new annotation implementations can replace the existing Taxonomy-based Disambiguation algorithm (TBD). The training required by the TBD is both automatic and manual. The manual portion requires users to provide approximately 700 yes/no judgments regarding label matching based on context (Dill et al., 2003).

SemTag has been applied as a non-domain specific semantic annotation tool. It has annotated 264 million web pages, generating 434 million automatically disambiguated semantic annotations (Dill et al., 2003). The taxonomy used by SemTag is TAP. TAP is shallow and covers a range of lexical and

taxonomic information about popular items such as music, movies, authors, sports, health and so forth (Dill et al., 2003). The annotations generated by SemTag are stored separate from the source document. It is assumed that the source document is read-only, since the annotator in this case is not the original author. The intent of the SemTag/Seeker design is to provide a public repository with an API that will allow agents to retrieve the web page from its source and then request the annotations separately from a Semantic Label Bureau (Dill et al., 2003).

#### 4. Future Trends

Semantic annotation platforms for the Web have only recently been developed, and they are not complete in their accuracy and elimination of manual effort. The precision and recall still vary widely depending on the platform used, IE methods, and data source type (unstructured, semi-structured, or structured), as shown in Table 1. There still exists opportunity to improve the performance of SAPs and reduce their required manual effort. A promising approach is the use of Hidden Markov Models. A future trend in semantic annotation platforms, then, is the continued integration of technologies originally developed in the field of information extraction. In the following sections, the integration of Hidden Markov Models into SAPs, as an example of applying other information extraction methods to semantic annotation, is discussed.

#### 4.1 Information Extraction Method Integration

Hidden Markov Models (HMMs) are models of output produced by real-world processes (Rabiner, 1989). Examples include acoustic signals used to identify speech, and bibliographic entries used to identify published works. HMMs are distinguished by the fact that they can only observe outputs, and not the events that produced the output. The models are then generated using only the outputs, in an attempt to understand the events which produced them (Rabiner, 1989). Rabiner (1989) describes a simple example of this. Assume there is a set of coin-toss using two coins. The person tossing coins is located behind a wall so that you cannot see the coin toss occur. As each coin toss occurs, the person calls out heads or tails. The modeling task is to explain the sequence of observed heads or tails.

A more practical example involves the use of HMMs in the AutoBib system (Geng & Yang, 2004). AutoBib is designed to extract information from bibliographic sources. The idea is to take the regularity of a bibliographic citation, and extract various fields, such as Title, Author, Publisher, and Year. In the AutoBib application, the observations are the instances of each defined field (for example, 'Yang' is the 'Author'), and the hidden states are the fields (no values) Title, Author, Year, and so forth. Bibliographic information is an example of a structure modeled by an HMM.

HMMs are constructed using finite state machines. The state transitions are associated with a probability on each link, indicating the probability of transition from one state to the next. Each state also has an associated set of output probabilities. The output probabilities define what the probability is for each of a set of defined output symbols (Duda, Hart, & Stork, 2000). A core challenge is defining what the states are, since that is the basis of constructing a model (Rabiner, 1989).

A problem with HMMs is that they require a set of data to train the model with. Once the states of the model have been defined, the transition probabilities and the output probabilities associated with each state need to be trained. Training normally occurs by setting the transition and output probability matrix values to some arbitrary starting value, and then iteratively refining the probability values in each matrix using the training data (Duda et al., 2000). The amount of manual effort required to produce the training set can be reduced by providing a method to bootstrap the system with data, and then automatically expanding the number of training samples (Geng & Yang, 2004). As an example of the effort required to train HMMs, the MUSE system reports that their rule-based system, based on manual rule generation, can be ported to different languages faster than machine-learning based systems because of the effort required to generate training samples (Maynard, 2003).

Hidden Markov Models have been used successfully in the Information Extraction field for parsing structured information, such as addresses (Borkar et al., 2001), and bibliographic information (Geng & Yang, 2004). A key attribute of HMMs is that they are adaptable in the case where information may be missing. For example, in structured record extraction, HMMs can handle variations in the record structure, such as missing or optional fields (Geng & Yang, 2004). A visualization of a typical HMM is shown in Figure 4.

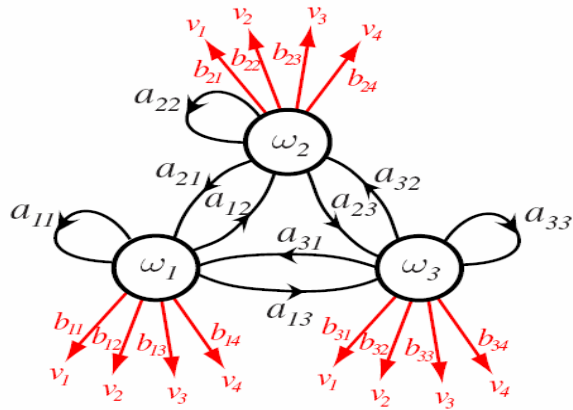


Figure 4: Visualization of typical HMM.  $w$ 's are hidden states,  $a$ 's are state transition probabilities, and  $b$ 's are observation probabilities (Rabiner, 1989)

#### 4.1.1 Hidden Markov Models and Semantic Annotation

Applying Hidden Markov Models (HMMs) to semantic annotation has just recently begun to be explored. The DATAMOLD tool (Borkar et al., 2001) was designed to extract address information from various datasets as part of the cleansing process in building a data warehouse. The problem DATAMOLD was designed to solve is to extract information from different text styles, including free-form text. This is a similar process to semantic annotation, where documents may be in structured, unstructured, or semi-structured form, and the goal is to identify entities within the text and assign them to slots in the ontology. The AutoBib system (Geng & Yang, 2004) uses HMMs in a similar way, except the application domain is bibliographic information. Also, AutoBib provides a facility for automatically generating training samples from a few seed samples. Both DATAMOLD and AutoBib represent the

text within the model as a sequence of words. The words form the set of output symbols that a particular state may observe. This approach has led to good results for DATAMOLD, depending on the domain of addresses. For U.S. addresses, overall precision and recall is 99.605%, while for corporate addresses, overall precision and recall is 83.656% (Borkar et al., 2001). AutoBib saw similar results, depending on the data source type. For the DBLP bibliographic data source, the measured accuracy is 98.9%, while for the CSWD data source 93.4% accuracy was obtained (Geng & Yang, 2004).

In an effort to improve effectiveness with complex language texts, such as biomedical texts, work was done to construct HMMs by incorporating grammatical structure information. The idea is to perform a shallow parse of a sentence in order to generate phrase segments of a particular type, such as noun-phrase and verb-phrase. These phrases are then labeled if they contain words from the target terminology. The labeled phrases are then used as the hidden states within the HMM (Ray & Craven, 2001). The result is that this phrase model works better for some types of datasets as compared to the sequence-of-words model, while it can actually be lower performing for other types of datasets (Ray & Craven, 2001). The architecture of this model is shown in Figure 3. The same group later did expanded work to build Hierarchical Hidden Markov models, in which there are two layers of an HMM rather than one. The upper level represents phrases, as before, but the lower level represents individual words within a phrase (Skounakis, Craven, & Ray, 2003). This model results in improved performance for all tested data sets, as compared to the sequences of words model (Skounakis et al., 2003). The architecture of the phrase model is shown in Figure 5.

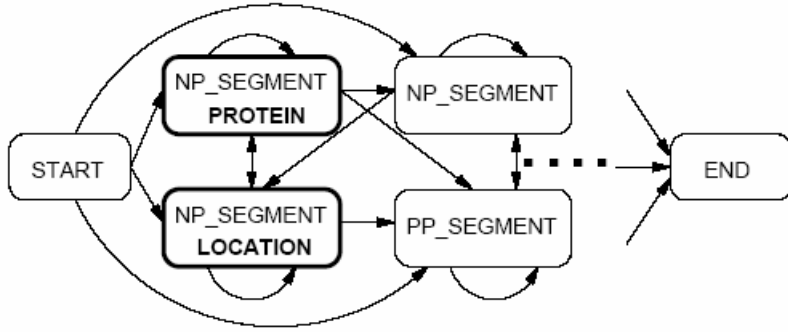


Figure 5: Architecture of a grammatical-information HMM based on part-of-speech phrases. (Ray & Craven, 2001)

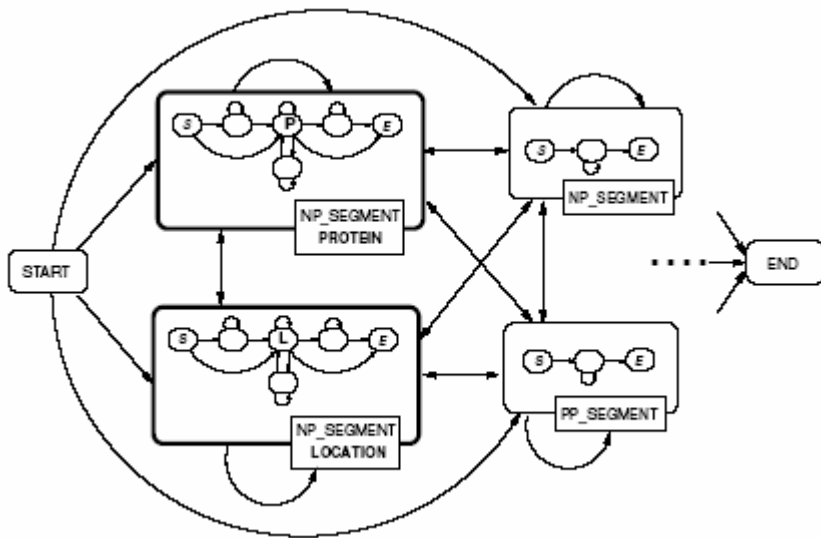


Figure 6: Architecture of a hierarchical phrase-based HMM. The dark rectangles indicate the upper level HMM, while the contents of the rectangle are the lower level HMM. (Skounakis et al., 2003)

All of these efforts are stepping stones to integrate HMMs in order to perform semantic annotation. Most of the focus on HMMs has been on producing information extraction methods, rather than integrating the methods into a larger semantic annotation platform. There has been some recent effort to use HMMs for Semantic Web annotation, although none to date has integrated the HMM methods from the IE field into a semantic annotation platform. The Rainbow project (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004), which aims to build a domain-specific semantic search engine, has implemented HMMs in order to perform semantic annotation. The implemented HMM uses the sequence-of-words model. The target documents are HTML pages, and each page has transformation applied before the HMM is used. The transformations include: 1) HTML block elements (such as table cells) are identified; 2) inline HTML element strings are converted into abstract tags (such as <important>); and 3) manual rules are applied to replace well-known sections of forms with tags (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004). This means that substantial pre-processing must occur before the HMM can be applied, and some of this pre-processing is manually generated. An interesting aspect of the Rainbow project HMM is that a single model incorporating all semantic slots to be filled is used, rather than a model (HMM) for each semantic slot (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004). The ontology used defines product catalog information for bicycle products, so the domain is very narrow and the ontology slots to be filled number only a few. The results of the HMM application are different depending on the semantic slot. The best slot, *price*, has a precision of 98.9% and a recall of 89.5%, while the worst performing slot, *picture*, has a precision of 89.6% with a recall of 69% (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004). Further work to extract sets of slots, in order to perform template extraction of an entire ontology class rather than single slot extraction of each ontology class property, results in a less than 50% match as compared to manual annotation (Svab Ondrej, Labsky Martin,

Svatek Vojtech, 2004). The result is that single slot (unary) extraction is currently more effective with their current work than multi-slot (n-ary) extraction. Project Rainbow expects to provide some level of annotation services similar to what is available in a semantic annotation platform in future work. However, there are no plans to integrate its HMM work into an existing SAP. The CROSSMARC project also looks to integrate HMMs into the semantic annotation process. CROSSMARC focuses on annotating named entities, and uses the HMM to find new entities not found by its current semantic tagger, which uses string matching, similar to a gazetteer (Valarakos, Sigletos, Karkaletsis, & Paliouras, 2003). The idea is to supplement the string matching to find known entities, and use the HMM to find additional entities that are not currently known. Once new entities are discovered, they are added to the ontology and become known entities. In this way, the pool of known entities continues to grow for subsequent annotation processing.

#### 4.1.2 Integration of HMMs into Semantic Annotation Platforms

With the number of available semantic annotation platforms currently available, as shown in Table 1, it is possible to extend existing SAPs with newer annotation implementations that may lead to improved annotation accuracy beyond what current platforms are producing. While work has been done to use HMMs in information extraction, as discussed previously, and related work has also been done to perform annotation in a domain search project, HMMs have not been integrated with any of the SAPs to date. It seems reasonable that this prior work could be used as a starting point to integrate HMMs into one of the SAPs.

As an example, the integration of other methods into an existing SAP has been done. The Ont-O-Mat system (Handschuh et al., 2002) was originally designed using Amilcare to learn rules about linguistic patterns in order to identify entities. Further work done by separate researchers developed a method called PANKOW (Pattern-based Annotation through Knowledge on the Web) and integrated it into the Ont-O-Mat platform (Cimiano et al., 2004). The advantage of the integration is that the work could focus on the method rather than on constructing supporting services provided by a SAP. In particular, PANKOW utilizes the ontology and document management facilities provided by Ont-O-Mat (Cimiano et al., 2004).

A primary disadvantage HMMs have over the rule-based systems that are largely implemented in current SAPs (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004) is the training problem. The data required to get good accuracy out of an HMM may be fairly large (Duda et al., 2000). If this training data must be manually produced, a bottleneck and maintenance issue results. However, since SAPs normally have a knowledgebase as a component of the system, it may be possible to develop a way to bootstrap the training process using the knowledgebase (Valarakos et al., 2003). For example, in the CROSSMARC project, entities are tagged using information from the ontology and knowledgebase (Valarakos et al., 2003). These annotations are then fed into a HMM in order to train it. The HMM is then used to find additional entities not tagged by the ontology tagger.

As discussed, the integration of HMMs into semantic annotation platforms is an uncompleted effort. HMMs offer advantages over existing semantic annotation approaches, such as rules. For example, HMMs are more flexible than static rules due to their probabilistic nature. HMMs also have

disadvantages, primarily the amount of training data required. However, this disadvantage may be reduced or even eliminated by using information contained in the SAP ontology and knowledgebase to bootstrap annotated examples. However, current HMM work developed outside of SAPs are not able to take advantage of such platform features. It is also anticipated that HMMs focusing directly on semantic annotation may also be developed, such as the HMM work the Rainbow project started (Skounakis et al., 2003). Semantic HMMs would focus on filling semantic slots directly from text rather than going through a HMM at the information extraction stage. Typical information extraction HMMs can locate entities, which are then semantically tagged using gazetteers or other semantic tagging methods (Skounakis et al., 2003).

Semantic annotation platforms, then, would benefit from HMMs by taking advantage of potentially improved annotation accuracy through their inherent structure flexibility over alternate methods, such as rules. At the same time, SAPs provide supporting features that can potentially mitigate the machine learning training disadvantage of HMMs.

## 5. Conclusion

The Semantic Web requires the widespread availability of document annotations in order to be realized. Benefits of adding meaning to the Web include: query processing using concept-searching rather than keyword-searching (Berners-Lee et al., 2001); custom Web page generation for the visually-impaired (Yesilada et al., 2004); using information in different contexts, depending on the needs and viewpoint of

the user (Dill et al., 2003); and question-answering (Kogut & Holmes, 2001). Annotations are currently problematic for several reasons. Manual annotation does not scale to the volume of documents on the Web, and suffers from problems such as annotator motivation and domain knowledge (Bayerl et al., 2003). There are additional problems with manual annotation, such as changing ontologies, and having a document annotated using multiple ontologies, providing multiple perspectives (Dill et al., 2003).

Semantic Annotation Platforms were developed to provide a level of automation to the semantic process, and overcome the limitations of manual annotation. Several semantic annotation platforms currently exist, distinguished primarily by their annotation method, as that component has the largest impact on the effectiveness of semantic annotation. The two primary approaches are pattern-based and machine learning-based. Machine learning algorithms often perform more effectively than pattern-based methods, but the MUSE system shows that a rule-based system using conditional processing can perform as well as a machine learning system (Maynard, 2003).

SAPs designed with extensible architectures can adapt to evolving technology. Information extraction components can be replaced as different approaches are developed. The most common toolkits used for entity identification are GATE (University of Sheffield, 2004) and Amilcare (University of Sheffield, 2002).

Applications making use of annotations generated by semantic annotation platforms are beginning to appear. These applications are information retrieval by semantic entity rather than by keyword, custom web-page generation based on different user needs and perspectives, question-answering systems, and visualization of a domain.

Future work will likely focus on integrating other information extraction approaches from the information extraction and natural language processing communities. In addition, ways to bootstrap the semantic annotation process and reduce the amount of manual effort will also evolve. This includes ontology learning and knowledgebase population. Extensible platforms will enable the rapid development of new advances by allowing individual components of a semantic annotation platform to be replaced or extended. The continuing evolution of SAPs to provide better annotation and new features while extending existing ones is vital to the realization of the Semantic Web.

## 6. References

Bayerl, P. S., Lungen, H., Gut, U., & Paul, K. I. (2003). Methodology for Reliable Schema Development and Evaluation of Manual Annotations. *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003)*, Florida, USA.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34-43.

Borkar, V., Deshmukhy, K., & Sarawagiz, S. (2001). Automatic Segmentation of Text into Structured Records. *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, Santa Barbara, California, USA, 30(2) 175-186.

Brin, S. (1998). Extracting Patterns and Relations from the World Wide Web. *Proceedings of the WebDB Workshop at 6th International Conference on Extending Database Technology*.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies and why do we need them? *IEEE Intelligent Systems*, 14(1), 20-26.

Cimiano, P., Handschuh, S., & Staab, S. (2004). Towards the Self-Annotating Web. *Thirteenth International Conference on World Wide Web*, New York, NY, USA, 462-471.

Ciravegna, F. (2001). Adaptive Information Extraction from Text by Rule Induction and Generalisation. *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, Seattle, USA.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

Cunningham, H., Maynard, D., & Tablan, V. (2000). *JAPE: A java annotation patterns engine*.

Cutting, D. (2004). Apache Jakarta Lucene.

Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., & Jhingran, A. et al. (2003). SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. *Twelfth International World Wide Web Conference*, Budapest, Hungary, 178-186.

Dingli, A., Ciravegna, F., & Wilks, Y. (2003). Automatic Semantic Annotation using Unsupervised Information Extraction and Integration. *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003)*, Florida, USA.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification* (2nd ed.). Hoboken, NJ, USA: Wiley Interscience.

Geng, J., & Yang, J. (2004). Automatic Extraction of Bibliographic Information on the Web. *Proceedings of the 8th International Database Engineering and Applications Symposium (IDEAS'04)*, Coimbra, Portugal, 00 193-204.

Handschuh, S., Staab, S., & Ciravegna, F. (2002). S-CREAM -- Semi-automatic CREATION of Metadata. *SAAKM 2002 -Semantic Authoring, Annotation & Knowledge Markup - Preliminary Workshop Programme*.

Hearst, M. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France,

Kiryakov, A., Popov, B., Ognyanoff, D., Manov, D., Korilov, A., & Goranov, M. (2003). Semantic Annotation, Indexing, and Retrieval. *ISWC 2003, Second International Semantic Web Conference*.

- Kogut, P., & Holmes, W. (2001). AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP 2001), Victoria, BC.
- Kosala, R., & Blockeel, H. (2000). Web Mining Research: A Survey. *SIGKDD Explorations*, 2(1), 1-15.
- Kushmerick, N., Weld, D. S., & Doorenbos, R. (1997). Wrapper Induction for Information Extraction. *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI '97)*, Nagoya, Japan, 729-737.
- Maedche, A., & Staab, S. (2001). Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), 72-79.
- Maynard, D. (2003). Multi-Source and Multilingual Information Extraction. *Expert Update*,
- Missikoff, M., Navigli, R., & Velardi, P. (2002). The Usable Ontology: An Environment for Building and Assessing a Domain Ontology. *1st International Semantic Web Conference (ISWC2002)*, 39-53.
- Mukherjee, S., Yang, G., & Ramakrishnan, I. V. (2003). Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis. *Second International Semantic Web Conference*.
- Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., & Goranov, M. (2003). KIM - Semantic Annotation Platform. *2nd International Semantic Web Conference (ISWC2003)*, 2870 834-849.
- Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), 257-285.
- Ray, S., & Craven, M. (2001). Representing Sentence Structure in Hidden Markov Models for Information Extraction. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA, 1273-1279.
- Skounakis, M., Craven, M., & Ray, S. (2003). Hierarchical Hidden Markov Models for Information Extraction. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 427-433.
- Svab Ondrej, Labsky Martin, Svatek Vojtech. (2004). RDF-Based Retrieval of Information Extracted from Web Product Catalogues. Proceedings of the 27th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, Semantic Web Workshop, Sheffield, UK.
- Tallis, M. (2003). Semantic Word Processing for Content Authors. *Second International Conference on Knowledge Capture*, Sanibel, Florida, USA.

University of Sheffield. (2004). *GATE - A General Architecture for Text Engineering*. Retrieved December 28, 2004 from <http://gate.ac.uk/>.

University of Sheffield. (2002). *Amilcare*. Retrieved December 28, 2004 from <http://nlp.shef.ac.uk/amilcare/amilcare.html>.

Valarakos, A., Sigletos, G., Karkaletsis, V., & Paliouras, G. (2003). A Methodology for Semantically Annotating a Corpus Using a Domain Ontology and Machine Learning. *Proceedings of the RANLP'2003 Conference*, Borovets, Bulgaria, 495-499.

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., & Ciravegna, F. (2002). MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. *The 13th International Conference on Knowledge Engineering and Management (EKAW 2002)*, 379-391.

Yesilada, Y., Harper, S., Goble, C., & Stevens, R. (2004). Ontology Based Semantic Annotation for Visually Impaired Web Travellers. *Proceedings of the 4th International Conference on Web Engineering (ICWE 2004)*, Munich, Germany, 3140 445-458.