PhD Proposal: Learning By Reading
Automatic Knowledge Extraction through Semantic Analysis

Jesse English

8/13/07

1. Statement of Topic and Goals

Semantically annotated text has received plenty of attention recently from various academic and commercial interests. The Semantic Web, an extension of the World Wide Web, whose content is expressed in both natural language, and a marked-up, or meta-language understandable to software agents, is a hot topic, and for good reason. Extracting opinions from blogs, topic gisting of informative pages, and question/answering in search engines: all of these processes, and more, benefit from semantically annotated text.

Although text can be semantically annotated by hand, this is a time consuming, and error prone task, and is the cause of one of the major drawbacks to basing a system around semantically annotated text. Natural Language Processing (NLP), is a field of artificial intelligence concerned with extracting semantic annotation from natural language texts (either automatically, or semi-automatically). A NLP system aims to understand some language (or subset of a language), and facilitates any task whose input is semantically annotated texts.

Machine Learning (ML), another field of artificial intelligence, is concerned primarily with the development of algorithms which allow a computer to learn something, given a set of inputs, that it did not yet know; a primary goal of ML is to facilitate this task in a fully automated way, and to produce meaningful and relevant evaluations (also automatically) of the learned information. A successful ML system will be able to both learn new and useful information, and report back how accurate the information learned is, on the whole.

The proposal presented in this document reflects a hybrid of the two field of AI described above: NLP and ML, for the purpose of facilitating the development of an NLP system, which in turn facilitates automatic and semi-automatic semantic annotation of text. Any NLP (and indeed ML) system needs a series of inputs, or a set of bootstrapped data. In NLP, this is an extremely time consuming process, and the quality and coverage of the NLP system is directly related to the quality and coverage of the "world knowledge" it is given. The proposal presented in the remainder of this document aims to use a NLP system and the web as a corpus as inputs to a ML system. The ML system, as outputs, will produce more semantically annotated "world concepts" to be fed into the NLP system. This process will feed upon itself, creating a spiral learning cycle. As the NLP system grows, it will produce more and better coverage, allowing the ML system to learn more and better knowledge, which will in turn grow the NLP system again.

Such a system will facilitate the production of semantically annotated texts, and qualifies as a "lifelong learning" system: as long as new information exists, and is available through the WWW, the system will continue to learn, annotate, and archive that information. In addition, such a system would also promote the growth of one of the long-term goals in the NLP/AI community: the creation of a cognitive system (or agent), capable of communication with humans (in the form of comprehensive language understanding and generation).

2. Motivation

One of the greatest challenges to the quality and coverage of a NLP system is the acquisition of knowledge. Without a sufficient lexicon of words, and a broad coverage of "world knowledge", a NLP system would fall short on many, if not most, texts it encountered. Although clever NLP systems may be able to get around their knowledge shortcomings with crafty syntactic dependency structures, among other tricks, this is merely a workaround to produce some sort of results. The "knowledge acquisition bottleneck" is often observed as one of the fundamental problems facing the development of a quality, broad coverage NLP system.

For this reason, many NLP systems are developed with specific corpora in mind. Knowing a system will only ever have to semantically annotate a corpus of very specific texts on certain medical conditions can alleviate the need for acquiring knowledge on sports teams, soft drinks, or Mongolian history. However, such an NLP system does not have broad coverage, as its purpose is singular in nature.

For this reason, automating knowledge acquisition is considered a fundamental step towards developing a quality NLP system; acquiring all the world's knowledge by hand would be far too laborious a process. Automating knowledge acquisition has been worked on in a variety of fashions for some time now. [Navigli et al., 2004] used a method known as structural semantic interconnections to produce the meaning of a complex concept from the meanings of its constituent concepts (*business plan* derived from *business* and *plan*). Often times, ML methods are applied over a syntactically parsed sentence to extract semantic roles [Yangarber, 2003], [Reinberger and Spyns, 2004], [Toutanova et al., 2005] to name a few.

There is an inherent drawback in the way these learners work, however: in general, a system that is only taking into account a syntactic parse tree (be it deep or shallow) as input to extract semantic information from a text is missing out on a wide amount of data that can only be extracted with a semantic parse of the same text. However, in order to produce a semantic parse, a semantic analyzer, with a substantially broad coverage of "world knowledge" must be present. This presents a Catch-22: in order to accurately capture all of the knowledge hidden in a text, a semantic text analyzer must be used, but in order to produce such an analyzer, knowledge must be learned. The only way out of this cycle is to spend a long time acquiring knowledge for the analyzer by hand. This, naturally, leads to the hope that given a significantly broad coverage of knowledge, the

remainder (and majority) of the knowledge can be learned automatically, thus saving significant time in the development of a robust and powerful semantic text analyzer.

If such a system could be devised, it could fuel itself until it had substantially broad enough coverage and quality of "world knowledge" to begin annotating the low-quality texts found widely through the WWW automatically. Such a system could semantically annotate the web accurately, and without human intervention, thus facilitating opinion extracting, topic gisting, question answering, and more.

3. Proposal

This document proposes a system that, given a semantic analyzer of varying quality, a set of "world knowledge" with broad enough coverage of common words in a given language (note that common words refers to commonly used words, but in no way the majority of words, or lexical senses, found in the language), and an open corpus, will continually seek out unknown words and senses in the corpus, and using the corpus and semantic analyzer, produce a high quality set of new "world knowledge" to be appended to the existing set, thus broadening coverage and allowing for more accurate learning to continue.

Such a system would work in the following way:

1. Given an input of an unknown word (this may be found automatically be simply scanning the corpus until one is discovered, or entered manually), the corpus will be scanned, and all relevant documents will be retrieved.
2. The documents will be semantically analyzed using the NLP system. Although the unknown word will produce less than completely accurate results (by definition), a process known as *unidirectional selectional restrictions* will allow the NLP system to make quality guesses as to the meaning of the unknown word, and will proceed to produce an analysis that assigns meaning and knowledge to the guessed word.
3. The snippets of knowledge about the unknown word collected from each analyzed text in the corpus will be combined to produce a much more accurate "meaning" of the unknown word.
4. This meaning will be compared to existing meanings in the "world knowledge", in order to find a hierarchical, or hypernym/hyponym, relationship between the new and existing knowledge. Such a relationship will allow the system to "fill in the blanks", fleshing out the knowledge of the new word. This word can then be added to the "world knowledge", and is ready to be used in further semantic analysis.

For a system to undergo lifelong learning, it must be able to endlessly pick up on the existence of new information, and have a suitable amount of supporting facts to learn from. If this is to be the case, a closed corpus, or domain specific corpus is not sufficient. Using a domain specific corpus, a learning system could eventually exhaust all of the knowledge contained, and would eventually be left to spin its wheels. Although this may

be very practical for a specific task, and indeed the proposed method could be adapted to this task (given a suitably large enough corpus), it is not the desired goal of a lifelong learner. The corpus proposed must be an open corpus, with general knowledge on most anything. Due to its accessibility, size, rate of growth, and naturally occurring inaccuracies, the web is a perfect candidate for a lifelong learning system.

Using the web as a corpus has been done before, in a variety of learning tasks. It may be used to provide data (in the most standard way a corpus does), or to provide statistical measures of linguistic phenomena (using Google hit counts, among others). [Kilgarriff and Grefenstette, 2003] covered the issue of the web as a corpus. They concluded that the web fits the definition of a corpus, and can provide a variety of positive information by the nature of its noisy data. Although erroneous data (factual, grammatical, etc.) exists, it is often far outweighed by the volume of correct data; this can be beneficial information to any language learning system: knowing what can be wrong, and how it can be wrong is a powerful tool, especially when it the pattern of "wrongness" can be clearly picked out and recorded.

The web works especially well as a corpus for the system proposed: the system is intended to simply find unknown words, of no particular domain, and learn their meaning. The system is presupposed to have broad enough coverage of common world knowledge mapped to lexical senses. Having this background, the proposed system should be able to query the web for texts matching the unknown word, retrieve an ample amount of simple sentences, of which all but the target word are known to the system, and construct a reasonably accurate semantic representation from the text. In this way, the web provides a more ideal corpus than a domain specific one: a domain specific text would have more domain specific (and hence, unknown) words per sentence than a general, or domain in-specific one. In order for the system to work as well on a domain specific corpus, knowledge acquisition on the common words to that corpus would have to first be done, in order to grant the system a "leg up" on the knowledge.

However, the noise of the web can also be a drawback to such a system. A variety of clever filtering methods will have to be used to weed out bad data. These methods can be at the syntactic or semantic sentence meaning level, or as basic as systems level filtering: removing malformed html, and timed-out page requests.

Filtering will also have to be done at the knowledge learning stage. After semantic analysis, the extracted knowledge will have to be filtered to provide a more accurate view of the unknown word. This filtering can be done by comparing the extracted data with other extracted data, or by comparing it to the bootstrapped (manually acquired, or previously learned) world knowledge. Following the filtering, the pruned knowledge will have to be placed into the existing knowledge appropriately. Comparison algorithms will have to test the similarity of the new knowledge to existing data, in an attempt to find a location in the data hierarchy. This can be done using a *nearest neighbor* technique (using Euclidean distance in an n-dimensional space), or other similarity measures.

Once the data has been extracted, filtered, and placed in the existing knowledge, it needs to be evaluated. Evaluation can take place both manually, and automatically. Learned knowledge needs to have both its accuracy and correctness evaluated, its usefulness in the "world knowledge" as a whole, as well as its benefit to the learning system in the future. Recall, learning knowledge through this system is intended to reduce the pressure of manual knowledge acquisition. If the learned knowledge is inaccurate, or serves little purpose in the existing knowledge, then time has been wasted.

To test the accuracy of the learned knowledge, an expert can manually acquire knowledge. The automatically learned, and manually acquired knowledge can be compared side by side to determine a level of correctness. To judge the usefulness of adding the learned knowledge to the "world knowledge", semantic annotations of text containing the previously unknown word can be generated twice: once with, and once without the learned knowledge. The quality of the annotations can be compared to each other, or compared to a third, manually constructed annotation. Judging the benefit to the learning system can be done by attempting to learn a subsequent, yet related word, once with the new knowledge and once without. The newly learned data can then be evaluated in the same ways described before, giving a third method of evaluation to the previously learned knowledge.

Any of these methods of evaluation can also be performed automatically (with less accuracy) by using various ML evaluation techniques. *Leave one out cross-validation* can be used to compare semantic annotations, and automatically generated knowledge.

4. Setting up the Experiments

In order to begin experimentation, the system presupposes the existence of a semantic text analyzer. To fill this need, the OntoSem system [Nirenburg and Raskin, 2004] will be used. OntoSem, an implementation of the theory of ontological semantics is a system developed at the ILIT research lab at UMBC. OntoSem's primary function is to produce semantic annotations of text (in the form of text meaning representations: TMRs), and uses, as input, a semantic ontology of world knowledge, and a semantic lexicon of word sense mappings (with constraints) to the ontology.

The use of OntoSem provides a perfect starting point for the proposed system. OntoSem itself is capable of semantically annotating texts, and the quality of its annotations are in a large part dependent on the quality and coverage of its ontology and lexicon. The ontology consists of a collection of concepts, which are hierarchically ordered and defined as a set of property/value pairs. The output of the annotations, the TMR, is a collection of constrained property/value pairs on instances of ontological concepts. This setup would allow the proposed learning system to use OntoSem to semantically analyze a text, and extract snippets of world knowledge about an unknown word in the form of automatically generated property/value pairs. These pairs have an intimate relationship with the definition of the ontology, and would allow for accurate comparisons to existing knowledge.

To provide texts to OntoSem for analysis, a search engine could query the web, which was previously discussed as a powerful open corpus; results, in the form of texts and meta-data (number of hits, suggested alternate searches, etc.) could be analyzed and archived for further data extraction.

In order to setup any experiment, and to evaluate the results in any of the discussed methods, a set of tools and application programmer libraries will need to be available. Knowledge acquisition for both experimental setup, as well as evaluation of automatically generated knowledge is a time consuming and difficult process, which can be expedited by the use of user-friendly, custom-tailored tools. The same is true for the creation of manually annotated sentences, or hand-generated TMRs. In order to produce a series of functional tools that works seamlessly with the proposed learner, a solid API will need to be created to handle the data backend, and provide standard functionality to supplement the semantic annotation provided by OntoSem.

5. Prior Work

Work on the proposed system has progressed in two distinct areas. Initially, the system was developed and tested, with varied results. However, in the course of development and research it became clear that in order to facilitate the development of the proposed system, as well as the NLP tools (including OntoSem's static knowledge resources), a solid tools set with supporting API would need to be developed.

5.1 DEKADE

For two years, the DEKADE (Development, Evaluation, Knowledge Acquisition, Demonstration Environment) system has been under development to support various knowledge-based development and learning tasks. DEKADE has been designed as an immediate, high-level API and toolset for OntoSem and its supporting static knowledge resources. The current version of DEKADE is a custom, client-server architecture, built around an API designed to have uniform access to all of OntoSem's modules and static knowledge.

This process was a necessary step in the construction of the proposed system. In order to properly create bootstrapped knowledge for the system, and to extract relevant data from both the static knowledge and the results of semantic analysis, the ML component of the system needs high-level access to the various components that make up the OntoSem environment. The reasoning behind this is to produce the appropriate tools for the ML component to work properly, instead of forcing the ML component into a set of existing, "off-the-shelf" tools. OntoSem is an incredibly complex, and very unique environment: no existing tools would be capable of capturing the full scope of detail OntoSem brings to the table, hence selecting an existing tool would limit the learning potential of the ML component.

The initial task, and indeed the primary motivation to build DEKADE from scratch, was to develop an open, and powerful, API. DekadeAPI, written in Java, is an extensible

library that supports single function calls to access any of OntoSem's modules, as well as simple, yet robust, queries to the static knowledge resources. To improve the efficiency and coverage of these access methods, high-level Java objects have been created as wrapper classes to parse the results and present them to the user in an intuitive, easily accessible manner.

To further enhance the functionality of these methods, each was extended so that it is now possible to call it across an open-socket network connection, allowing the DekadeAPI to be usable by any user with an Internet connection. With this functionality available, it was time to construct an interface layer between the existing toolset and the user. The interface had to support the demands of three types of user: developer, knowledge acquirer, and researcher (who uses OntoSem as a tool, as it will be in the proposed learning system), just as the DekadeAPI does.

Built on Java/Swing technology, the interface's parent UI frame handles securing the connection between itself (the DekadeClient), and the server application (the DekadeServer), and populates itself with a series of tabbed panes found in the application's root drag-and-drop folder, registering the browsing capabilities of each panel with the others. The interface was developed to support custom user panes that simply append to the interface and integrate with the existing tools. Using standard Java/Swing libraries, and custom DekadeAPI GUI extensions, researchers can easily populate a panel with custom-built or existing DEKADE widgets, and use them for two-way communication with OntoSem.

A key component is the interconnection between the various editors and browsers. In the new DEKADE environment, a knowledge acquirer can begin work on a lexicon entry, and in a single click inspect the corresponding ontological entry, and then swiftly return to the lexicon entry. The developer can also easily inspect the details of the various mappings to static knowledge made by any of the OntoSem processing modules to assist in the testing and debugging process of OntoSem.

The current standard version of the DekadeClient environment is supplied with interfaces to support OntoSem Stepped Analysis, Lexicon Browsing/Editing, Ontology Browsing/Editing, and Fact Repository Browsing/Editing (the Fact Repository is a collection of instances of concepts from the ontology, with specific values and constraints).

Setting up the DEKADE system has facilitated the experimental process of developing the learning system proposed, and will continue to do so. In addition, the DEKADE system has been used in other research areas as well: DEKADE has been successfully used in the SemNews system [Java et al., 2006], [Java et al., 2007], a semantic web annotation project, cataloguing TMR-level annotations of RSS news feeds created by OntoSem. The system uses OntoSem as its backbone NLP system, and OntoSem's static knowledge resources as a default knowledge base. Another application, for which OntoSem provided the basis, and DEKADE the environment, is EBIDS [Stone, 2007], an NLP-based social engineering email detection system. In EBIDS, OntoSem is used

through DEKADE to semantically analyze incoming e-mail messages and identify those of them that can be social engineering ("phishing") threats.

5.2 The Learner

The solid foundation of OntoSem and DEKADE has provided a means to begin initial experimentation in learning on ontological concepts through semantic analysis. Initial findings, published in AAAI Spring Symposium, 2007 [English and Nirenburg, 2007], demonstrated that such a system was feasible, and even at its earliest stages showed promising results.

The first experiment concentrated on learning the meaning of unknown words. One simplifying assumption made at the time was that the meaning of the candidate would be expressed as a univocal mapping into an ontological concept (in general, SEM-STRUC zones of OntoSem lexicon entries can add local constraints to ontological concepts in terms of which the meaning of the lexical unit is described, thus making such lexicon entries unnamed ontological concepts). As a result, at the time, the experiment was effectively constrained to learning ontological concepts.

Ontology learning as a field concerns itself at this time with learning terms, (multilingual) synonyms, concepts, taxonomies (by far the most popular topic), relations and rules and axioms [Buitelaar et al., 2005]. The methods involved include different combinations of linguistic (knowledge-based) and statistical methods but mostly the latter. Among the linguistic tools used for this purpose Buitelaar et al., *ibid.*, list, in order of increasing sophistication, tokenization, part of speech and semantic type tagging, morphological analysis, phrase recognition, (syntactic) dependency structure determination and discourse analysis. It is notable that this list does not include a tool that would be centrally important for learning by reading – an analyzer that creates disambiguated semantic dependency structures for sentences and texts, in which relations between elements are ontological and not syntactic and go well beyond the taxonomic subsumption relations. Work on extracting small subsets of such relations using largely statistical means has been reported (e.g., [Charniak and Berland, 1999] for meronymy, [Cimiano and Wenderoth, 2005] for the qualia of the generative lexicon approach [Pustejovsky, 1995], causation [Girju, 2002], among others). OntoSem, however, addresses the task of extracting knowledge about a large set of such relations using encoded knowledge as heuristics (cf. work by, e.g., [Clark and Weir, 2002] that uses essentially statistical methods for estimating selectional restrictions).

Among the sources of knowledge acquisition are machine-readable dictionaries (e.g., [Nichols et al., 2006]), thesauri (e.g., [Navigli and Velardi, 2006]), as well as text (e.g., [Ogata and Collier, 2004], [Buitelaar et al., 2004], [Cimiano et al., 2005]). The experiment used open text but could be extended to treating MRDs and thesauri as special types of texts.

5.3 The Initial Experiment

The initial experiment started with selecting words whose meaning would be learned by the system. Next, the system developed a corpus of sentences (from the web) containing this word and used OntoSem to generate their TMRs. OntoSem degrades gracefully in the face of unexpected input. It is capable of semantically analyzing sentences with a small number of unknown words by assuming that the unknown word's meaning corresponds directly to a non-existent ontological concept and then applying relevant constraints listed in the knowledge about words syntactically connected with the unknown word to hypothesize the constraints on the latter. Note that, unlike the case when all the words are known and, therefore, constraints are mutually **matched** (this is a major mechanism for lexical and semantic dependency disambiguation, the core process of semantic analysis), in the case of an unknown word the constraints (for example, selectional restrictions) must be applied unidirectionally.

From the TMRs containing the new candidate concept, the system collected both the inventory of relations and attributes attested for the candidate concept and the inventory of values of these relations and attributes. After the candidate concept was thus "assembled," the system compared it with the concepts in the existing ontology to find the most appropriate position(s) for it in the multiple-inheritance hierarchy that organizes the OntoSem ontology. To facilitate the evaluation of our experimental results, some of the existing entries (and the corresponding ontological concepts) were taken out of the OntoSem knowledge resources prior to running the experiment; afterward, they were compared to the automatically generated new concepts.

The learner uses Google's SOAP Search API (http://www.google.com/apis/index.html), which returns a list of websites containing the unknown word. The content of a specified number of these sites is retrieved. This number can be adjusted if needed to extract a larger corpus. Next, an HTMLParser (http://htmlparser.sourceforge.net/) is used to strip out tags and unwanted markup, yielding raw text. At the next step, text is divided into sentences (using a module of the OntoSem analyzer), and those sentences that do not contain the search word are discarded.

The remaining sentences are processed using the OntoSem analyzer that carries out morphological, syntactic and semantic analysis. The output from OntoSem is a list of TMRs containing instances of a candidate concept corresponding to the unknown word and instances of a variety of relations in which this concept participates, such as a case role that relates it to an event-type concept instance. Additionally, the TMR may contain some values for attributes (unary properties) of the candidate concept. At this point in the process there is an option to include human assistance to produce gold-standard TMRs from the system's results (DEKADE would be used if desired here).

The list of properties returned from any one TMR is likely to be small (because only a few will be referred to in a single sentence). The system then has to collate the knowledge extracted from processing individual sentences. Given a list of TMRs, the learner searches through each one, finding all properties associated with the instances of

the candidate concept, and collating them into a single frame for the corresponding candidate ontological concept. When collating the values of each property of the candidate concept, the system filters out weaker constraints if stronger constraints have been attested. For example, if among the fillers of the AGENT-OF property of the candidate concept the system finds READ, ACTIVE-COGNITIVE-EVENT and MENTAL-EVENT, it will retain only READ because it is a descendent of the other two concepts in the ontology. The weaker constraints can, in principle be retained in the OntoSem ontology because the latter uses multi-level constraints to support robust disambiguation processes. Technically, the constraint read may appear in the SEM facet of the property AGENT-OF in the candidate concept while MENTAL-EVENT may appear as the filler of the RELAXABLE-TO facet of the same property.

The final step in the original experiment was to find existing ontological concepts that are most similar to the candidate concept, with the idea of suggesting a place for the new concept in the multiple-inheritance hierarchical organization of the OntoSem ontology. In general, there are three distinct outcomes:

1. The candidate concept is subsumed by an existing concept (meaning that the original unknown word is a synonym of an existing lexical entry).
2. The candidate concept is similar to an existing concept C, in which case the system will create a lexicon entry for the original unknown word and in the SEM-STRUC zone of this entry insert a reference to C, with further local constraints added to reflect the differences between the candidate concept and concept C.
3. The candidate concept is sufficiently dissimilar from existing concepts, and should be added to the ontology.

The method of using the results of learning sketched above will be included (with an optional human validation step) in OntoSem's knowledge acquisition environment. Two strategies were used to create new ontology entries: using genuinely new words (those not in the OntoSem lexicon) and using existing words and removing them from the lexicon (and concepts used to describe their meaning from the ontology) before the corresponding system run. In the latter strategy, the system thus had a gold standard concept against which to compare the candidate. In the former strategy, a hand picked concept in the existing ontology that was thought to be the most appropriate one to use in the description of the meaning of the new word (whether directly, through constrained lexical mapping or creation of a new concept) was used.

The initial experiment produced a set of results of four words – *hobbit* and *pundit* were entirely new to the system and *CEO* (and its corresponding concept PRESIDENT-CORPORATION) and *song* (and its corresponding concept SONG) were temporarily deleted from the OntoSem knowledge resources.

For *hobbit*, HUMAN was selected as the closest ontological concept. Table 1 shows a comparison of the selected properties of automatically generated candidate concept for *hobbit*, HOBBIT, and the existing concept HUMAN.

Ontological Property	Values in HUMAN	Values in HOBBIT
AGENT-OF	LIVE CREATE-ARTIFACT ELECT READ	LIVE CREATE-ARTIFACT ELECT
THEME-OF	RESCUE MARRY KILL	RESCUE KILL
HAS-OBJ-AS-PART	HEAD	n/a

Table 1: Comparison of selected properties of HUMAN to automatically generated candidate concept HOBBIT for the word *hobbit*.

Present in many of the automatically generated concepts were a relatively high proportion of properties labeled RELATION, which means that the system was not able to determine a more precise link (that is, a narrower-defined property) connecting the candidate concept with the filler of a RELATION instance. The OntoSem ontology uses approximately 200 specialized relations to characterize objects (the sub-tree of properties also contains a comparable number of attributes, and one-place predicates). The OntoSem analyzer uses the concept RELATION when it determines that two concept instances are related but lacks heuristics to specify what the specific relation it is. This over-generalized output is the price paid for making sure that the analyzer does not break on unexpected input or due to insufficient quality of the existing knowledge resources or decision heuristics. Even though the connection on RELATION is relatively underspecified, for the first experiment, the system kept this information and used it alongside other constraints in determining the distance between the candidate concept and other concepts in the ontology.

Evaluation was based on measuring the ontological distance between the candidate concept and all the other concepts in the ontology and then determining a) the difference in the distances between the candidate concept and the automatically derived closest concept and to the designated hand-picked closest concept; and b) the rank of the hand-picked concept in the sorted list of closest concepts. Table 2 presents the results of the initial experiment.

Word	Best Match	Selected Match	Difference
pundit	0.800	INTELLECTUAL 0.679	0.121 (15.1%)
ceo	0.900	PRESIDENT-CORPORATION 0.638	0.262 (29.1%)
hobbit	0.900	HUMAN 0.806	0.094 (10.4%)
song	0.800	SONG 0.800	0.000 (0%)

Table 2: Improvement of generated concept vs. target concept over growing corpora.

Distances between ontological concepts are measured using the OntoSearch algorithm [Onyshkevych, 1997]. OntoSearch finds the "best" ontological path (chain of relations) between any two concepts and calculates the weight (score) of each path, which reflects the strength of the association between two concepts. The cumulative score for a path is a function of its length and of the cost of traversing a particular relation link. For example,

subsumption links (IS-A and SUBCLASSES) are less costly to traverse than, say, causal links. The individual link traversal costs in OntoSearch were trained using simulated annealing on a representative subset of OntoSem ontological relations. OntoSearch has been used to provide statistics-based heuristics to supplement static knowledge resources during the operation of the OntoSem text analyzer. For example, to help disambiguate the input "*The doctor performed the operation*", Ontosearch examines the ontological connection between the non-title sense of *doctor* and the two senses of *operation*: PERFORM-SURGERY and MILITARY-ACTIVITY and returns the following link:

ONTOSEARCH(DOCTOR, PERFORM-SURGERY) → 0.8
DOCTOR AGENT-OF PERFORM-SURGERY

(In other words, even though the ontology may not overtly contain the information that a doctor performs surgeries, this information is virtually there and can be derived by the analyzer using OntoSearch.) The OntoSearch score for the DOCTOR/MILITARY-ACTIVITY relationship is much lower, so that the PERFORM-SURGERY sense is preferred.

As an evaluative tool, OntoSearch shows both a correctness metric, as well as a measure of improvement in results given different sizes of a dynamically produced relevant text corpus. As the ultimate goal of this research is developing a learning-by-reading capability, this provides data for testing the hypothesis: whether indeed the more the learner reads, the more it useful knowledge it obtains.

5.4 Refining the Comparison Metric

As a means of improving the above results, a new similarity metric was developed. This metric was designed specifically to judge the distance between two concepts without calculating a minimal-weight path between the concepts (which is the method used by OntoSearch and which was an obvious drawback because such a path was assumed to exist in the ontology). This change allowed for the candidate concept to not be expected to exist in the same ontology as the concepts it is being compared to.

This metric was been designed to look at each property in the current OntoSem ontology and compare concepts on the basis of the values of each property. A vector, whose length is the size of the property inventory in the current ontology, is constructed, each element's value weighed between 0 and 1 (see explanation of weight assignment below).

In constructing a metric for comparing two concepts, the number of property names that they have in common as well as a measure of similarity of value sets for the shared properties were taken into account. A simple average of these two values was taken to produce a similarity value. Let C_1 denote an existing concept and C_2, the candidate concept; let P_t be the total number of properties defined by both concepts (*union* of properties) and P_s , the total number of properties shared by both concepts (*intersection* of properties); and let V_i denote the vector of computed value pairs for all values in C_1 and C_2 with property *I;* V_{iv} stand for combined results of value set comparisons for property *I; and* V_{gt} , for the total number of V_{iv} values greater than 0.0. The system can

then compute a value of the intersection of the sets of properties defined for each of the compared concepts as $P = P_s / P_t$ and the quality of the intersection of value sets for the defined properties as

$$V = \frac{\sum_{i=0}^{P_s} V_{iv}}{V_{gt}}$$

The simple averaging of the two values yields: Similarity = (P + V)/2.

This improvement to the comparison metric also needed attention to the issue of individual value comparisons. Here, different metrics needed to be developed for different types of property fillers (numbers, numerical ranges, symbols, ontological concepts and their sets, etc.) Table 3 shows a partial list of property comparisons, with a brief explanation of how a result was determined; Figure 1 specifies how numerical ranges are compared.

Value 1 Type	**Value 2 Type**	**Comparison Metric**
Text	Text	Case-insensitive character-by-character comparison
Text	Number (literal)	No match
Number (literal)	Number (literal)	Is Num1 within (Num2 x *TOLERANCE*) distance of NUM2?
Number (literal)	Number (relative)	Match
Number (literal)	Number Range	Is Num1 inside Range2? (Range2 is expanded by *TOLERANCE*)
Number (literal)	Concept	No match
Number (relative)	Number (relative)	Is Num1 within (Num2 x *TOLERANCE*) distance of NUM2?
Number (relative)	Number Range	Match
Number (relative)	Concept	No match
Numerical Range	Numerical Range	Are ranges equivalent? See Figure 2
Numerical Range	Concept	No match
Set	Concept	No match
Concept	Concept	Calculate distance to nearest common ancestor

Table 3: Value type comparison overview. *TOLERANCE* is defaulted to 10%.

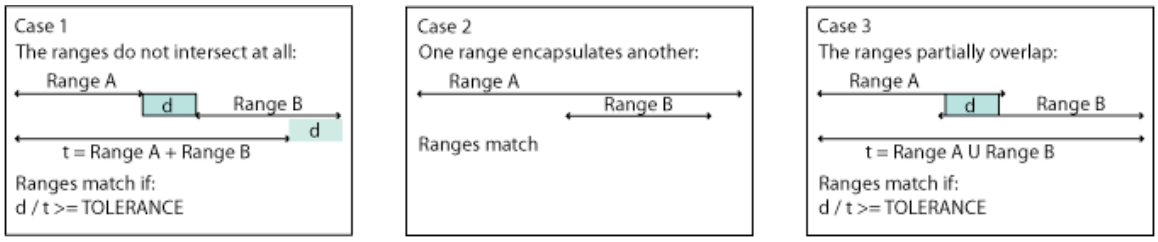

Figure 1: Three cases for numerical range comparison.

Using the new metric, the system obtained the results summarized in Table 4. The last column in the table, **Improvement**, compares the results of using OntoSearch metric and our latest metric by comparing the distances between the system-generated and the human-determined best match using the two metrics.

Word	Sample (no. of sentences)	Property Instances Extracted	Best Match	Desired Match	Differ-ence	Impro-vement
pundit	453	36	0.458	INTELLECTUAL (0.450)	1.75%	13.35%
CEO	552	23	0.448	PRESIDENT-CORPORATION (0.417)	6.92%	22.81%
hobbit	1458	157	0.520	HUMAN (0.493)	5.19%	5.21%
song	339	12	0.446	SONG (0.36)	17.71%	17.71%

Table 4: New comparison metric results. The **Sample** column lists the number of sentences containing the word that were used as the corpus. Those sentences were processed and yielded the number of property-value set instances listed. These property-value set instances were combined into a candidate concept, and a best match was found for it using the new metric. The match selected by a human is listed in the **Desired Match** column. The **Difference** column shows the difference in the similarity scores between the system and the human user. The **Improvement** column compares new results with analysis using the OntoSearch metric.

In three out of four cases there was improvement. In the case of *song*, the new metric yielded worse results. This was attributed to the small size of the set of the automatically generated property/value instances that formed the candidate SONG concept. OntoSearch imparts more weight to subsumption properties (IS-A, SUBCLASSES, which are semantically weaker than other properties) than the revised metric, and these properties were used predominantly to navigate the ontological hierarchy in the absence of more specific properties, resulting in a deceptively high score. The new metric, on the other hand, does not take these into account (very few actual sentences directly invoke subsumption relations), and therefore found very little useful information in SONG, resulting in a "decreased" quality of match (which is actually much more accurate than results using OntoSearch).

5.5 Combining Results

To further prove the usefulness of the proposed system, the results of a series of other parallel learning experiments were incorporated into the results obtained from another run of the system. Although not much was changed in the way the system produced results, a new batch of target words were selected. The first parallel experiment was designed to extract attribute values for existing concepts from an open corpus. This experiment differs from the proposed system in that it seeks to learn more literal values (instead of relations between concepts), and further it seeks to learn these values on existing concepts (to validate, clarify, and expand on existing knowledge). However, the general methodology was easily adapted to bolster the proposed learning system.

The OntoSem ontology was used as the basis for building search queries; each ontological property of the attribute (unary) type is associated with a list of its possible English realizations (obtained from the system's lexicon). A search query was created by combining this list with either a list of words, for example, for the concept ELEPHANT and the attribute WEIGHT, the following query was produced: (elephant) AND (weigh OR mass OR heavy OR heaviness). Note that since Google matches partial strings on queries, the search string *weigh* will match with many strings such as: *weigh*, *weight*, *weighing*, *weighs*, *weighed*, etc.

The result of the search was a list of sentences matching the query. These candidate sentences were then processed further by one of two different methods depending upon whether they contained measurable (e.g., weight), or non-measurable (e.g., color, whose values are represented in the OntoSem ontology by a set of primitive literals, such as *green*) attributes. In the latter case, a count was produced for the occurrences of each of the literals in the data. Table 5 illustrates the results for several runs aimed at empirical validation of existing ontological values. These searches were done on 500 web pages.

Concept/ Attribute	**Web Mining Results**	**Existing ontology has**
ELEPHANT/ COLOR	white: 283; pink: 188; blue: 92; black: 64; red: 61; gray: 45; green: 39; yellow: 39; brown: 23; purple: 13	black brown gray tan white
SPINACH/ COLOR	ed: 509 green: 297 black: 246 white: 227 orange: 129 blue:125	green
GRATER/ SHAPE	conical: 33; circular: 22; curved: 27; cylindrical: 23; rectangular: 23; hexagonal: 12	parallelepiped sheetlike trapezoidal

Table 5: Learning Literal Attribute Values

In the case of scalar attributes, each sentence in the results was searched for value-unit pairs. For example, the sentence "*The elephant weighs five tons and is ten feet tall"* contains two such pairs: *five tons* and *ten feet*. If the property being searched for is WEIGHT, then five tons is accepted a valid measure, since tons is a weight measure, while ten feet is rejected since weight is not measured in feet. All units were then converted into metric units, which is the standard in the OntoSem ontology. In some special cases, stop lists were generated to eliminate any errors that might be introduced by such conversions. Care was also taken to find ranges of values. If a sentence said "*Elephants weigh between 4000 and 9000 kilograms."* then both 4000 kilograms and 9000 kilograms were returned as valid elephant weight values. If the purpose of the data mining run is to determine constraints on a property of a newly learned concept, the range of values mined from the Web were compared with the range of values for the attribute in question within the definition of the concept in question in the OntoSem ontology. Table 6 presents a small sampling of results based on a search of 200 web pages for each concept-property pair.

Concept	**Attribute**	**Range Mined from the Web**	**Range in Ontology**
SQUASH	LENGTH	0.025 – 9.754 (meters)	0.012 – 0.024 (meters)
TUNA	WEIGHT	0.128 – 817 (kilograms)	2 – 820 (kilograms)
ELEPHANT	WEIGHT	0.227 – 10866 (kilograms)	3500 – 13000 kilograms

Table 6: Concept-Property Pairs for Scalar Attributes

The results of this first parallel experiment were combined with the results of preliminary work on the proposed system, with positive results. Adding attribute values to a set of relation values tended to improve the quality of the learned concept, and will be kept in mind as the system progresses. Table 7 shows some results that were found; in some cases the results improved with attributes, in others the results suffered. Discussion can be found in [Nirenburg et al., 2007, unpublished].

Word	A	B	C	D
Brontosaurus	DINOSAUR	0.607	0.607	0.715
Cherimoya	FRUIT-FOODSTUFF	0.607	0.646	0.637
Depose	DEPOSE	0.999 (ALL)	0.999 (ALL)	0.999 (ALL)
Diplodocus	DINOSAUR	0.612	0.612	0.546
Obey	OBEY	0.518	0.646	0.518
Pledge	PROMISE	0.516	0.516	0.760
Spartan	MILITARY-ROLE	0.574	0.574	0.754
Stegosaurus	DINOSAUR	0.720	0.682	0.759
Syrup	PLANT-DERIVED-FOODSTUFF	0.646	0.573	0.760
Triceratops	DINOSAUR	0.643	0.721	0.849
Wigger	SOCIAL-ROLE	0.526	0.635	0.849

Table 7: Results of Combined Experiments A and C on Eleven Unknown Concepts, with clustering

A: The targeted "correct" concept (existing in ontology). Repeated for clarity.

B: The best distance to the target, using attribute and relations (by OntoSearch comparison standards).

C: The best distance to the target, using relations and the best attribute value cluster (by OntoSearch comparison standards).

D: The best distance to the target, using relations only (by OntoSearch comparison standards). Repeated for clarity.

The second experiment, run in parallel, was designed to use statistical data based on web searches to identify the number of possible senses of a word (in particular, verbs). As this experiment already used the case roles AGENT and THEME that are among the properties processed in by the proposed system, the integration of the two experiments was natural.

In one experiment, the system was able to append the following property-value pairs to data already constructed for the unknown term *deport*:

1. <relation type="AGENT" value="NATION">
2. <relation type="AGENT" value="HUMAN">
3. <relation type="AGENT" value="PROCEDURE">
4. <relation type="AGENT" value="ICE">
5. <relation type="AGENT" value="SOCIAL-ROLE">
6. <relation type="THEME" value="HUMAN">
7. <relation type="THEME" value="NATION">
8. <relation type="THEME" value="CITY">
9. <relation type="THEME" value="CITIZEN">
10. <relation type="THEME" value="CRIMINAL">

Again, as with the previous experiment, results were mixed, but on the whole positive. Combining results from two completely independent experiments into the proposed

learning system, although no outstandingly positive results were produced, did have a very positive benefit: the learning system was shown to be lacking in certain areas, and was proven to be positively influenced by expansion into those areas.

6. Proposed Plan of Work

To begin using, and evaluating the proposed system, a collection of TMRs must be generated from the open corpus. Phase 1 of implementing and testing the proposed system includes a the implementation of a sub-system to take a requested unknown word as input, query the web for resulting texts, pass these texts through the semantic analyzer, and archive the resulting TMRs in a relational database. This sub-system will require a supporting database, with schemas supporting the TMR format and allowing easy query over the various components of interest in a given TMR (common search queries would include searching by keyword in the original text, searching for properties of an instance generated from a keyword, and searching for all TMRs containing instances of a concept, to name a few).

This sub-system will need to be setup to be "always running". It should be able to take each input word and append them to a queue, to be queried for and analyzed after the current tasks are completed. In this way, the system can be buffered with a large list of unknown, and set to work over time, while other tasks are being simultaneously worked on.

Phase 1 of the experiment must also involve careful selection of input words. At this point, the goal is to vastly improve the quality of learned material by tweaking various stages of the learning process, and incorporating new filters and stages to eliminate as much noise as possible. Selecting words with broad coverage of usage will assist in pointing out holes in the learning algorithm. Phase 1 should conclude with a much more solidly constructed set of learned concepts, which have been placed in the ontology with a high degree of accuracy.

Phase 1 will be evaluated mostly by hand, throughout its process. Gold-standard concepts will be created through the DEKADE interface, and the automatically generated concepts will be judged against them. Throughout this process, the method of judgment will also be improved over the one previously described. Phase 1 will conclude when the concepts created automatically have a high degree of similarity to the hand crafted ones, and the similarity metric is proven to be accurate as well.

Phase 2 of the research will involve implementing the "spiral learning method". The goal of this phase is to learn new concepts automatically, introduce them to the ontology, and use the new knowledge to learn another set of concepts. Phase 2 will have to be done in two parts, in order to show improvement. To begin, two sets of semantically related words will need to be selected: one as the original set of unknown words to be learned, and the second as subsequent set of unknown words, whose definitions are intrinsically dependant on the first set.

The first part of Phase 2 will involve using the sub-system to and automatic learning to extract the meanings of all words from both sets. The meanings of the second set will be stored for later. A copy of the first set will be made, and then manually corrected. The second part of Phase 2 involves adding the uncorrected set of knowledge from the first batch of words to the ontology, and then re-learning the second set of words. The uncorrected set will then be replaced with the corrected set, and the second set of words will be re-learned again.

Phase 2 will conclude with an examination of the three resulting sets of learned words. The quality and correctness of the sets produced using the uncorrected and corrected set of prerequisite words should be improved over the original set of learned words without the prerequisite knowledge. Further, all four sets of automatically learned words will be evaluated as in Phase 1 (both manually and automatically).

Phase 3 will involve production of TMRs of texts, which include unknown words from the second set of Phase 2. The TMRs will be produced once without the knowledge learned, once with the knowledge learned, but without the prerequisites, once with the knowledge learned (along with the prerequisites), and once by hand. The quality and correctness of the four sets of TMRs will be judged, and should be shown to improve as the knowledge improves.

7. Work By Others

Work in the fields of NLP (as a theory), ML (as a theory), semantic annotation, and automatic semantic learning, are all relevant foundations of the work proposed in this document. At this point, a certain amount of publications in related fields have been read and annotated. These texts are presented below (note that the annotations are written informally, are intended as summations of the text and are intended to be in an easy-to-read format); following the annotated texts is a list of texts that must also be digested as work continues.

7.1 Annotated Papers for the Core Learning Experiment

Automatic Ontology Learning: Supporting a Per-Concept Evaluation by Domain Experts [Navigli et al., 2004]

> The paper primarily focuses on two methods of evaluating automatically learned ontologies: a quantitative and a qualitative method. The first half of the paper describes (in brief) the OntoLearn system (no direct reference is given), which appears to be a hybrid ML and NLP ontology learning system. Starting with a WordNet/FrameNet springboard, a large corpus of domain specific text is scanned; keywords are extracted, and made into concepts using NLP. They use an algorithm, known as structural semantic interconnections, to derive the meaning of a complex term (such as business plan) from the meanings of its constituent terms (business and plan). The ontology is then augmented (business plan becomes a type of plan) and non-domain related material is cut.

Quantitative analysis is performed by a manual verification of the created concepts. Domain experts are used to verify that the concepts created appear to be sound... this is problematic as many domain experts are not ontologists.

This leads to the qualitative analysis; glosses are created for each concept, and these are given to a domain expert who votes on the quality of the information. A gloss is natural language snippet, generated from the concept. This is done by defining a grammar consisting of rules for each different property in the ontology. Each rule defines how to output some readable natural language, given the concept, slot and filler. These glosses produce something reads reasonably well, allowing a domain expert with no knowledge of ontologies to verify the data is accurate.

Building Concept Representations from Reusable Components [Clark and Porter, 1997]

The authors present a method for abstracting interactions between frames into groups (called components) that detail how a cluster of concepts interacts (sort of like a script). Using an ontology-based approach, frames are created, and are populated with property/value pairs. Inheritance (even multiple inheritance) is used to further define frames. To specify how multiple frames can similarly interact with a system, a component is defined. The example given is of the component containment: this component describes a container, a contained-object, and a portal. The interactions between them are defined in the component, and then other concepts are mapped to them; a human can be a container, with food as the contained-object, and mouth as the portal. By simply connecting these three concepts to the generic concepts of the component, human has now inherited the (potentially complex) structure of containment.

Using this methodology, the authors present a way to automatically build concept representations, and set the task to question answering. A rather lengthy example is given, where the question is "what is the cost of the equipment required for microbe-soil-treatment". Using a structure called an access-path (a sequence of variable pairs, where evaluating one allows for evaluation of the next, to the end), the authors create a system that starts with a known concept, and enhances a component graph (using defined components) by traversing an access path to the answer.

The example given finds the component for treatment, and uses the *microbes* and *soil* as fillers for *theme* and *patient*. Looking at the properties of treatment, they find that it has two events, get and apply. Apply is narrowed down (through selectional restriction on the components) to mix (mix has a patient soil, no other child of apply does). Looking at the properties of mix, it has an instrument mixer. Mixer is further narrowed to rototiller (as it mixes soil, and no other mixer does), and the cost of property of rototiller is $200. Thus the answer to the question is $200.

Although their system is not formally evaluated in the paper, the example gives a strong argument towards the usefulness of a system that is constructed using this methodology.

Counter-Training in Discovery of Semantic Patterns [Yangarber, 2003]

This paper describes work in semantic pattern classification; the authors are attempting to extract the relevant semantic patterns from a corpus of documents, and simultaneously select which documents should fall into which categories. This is done by starting with a corpus, and selecting a set of categories: each category is given a seed pattern to help with semantic classification.

At the base level, the corpus is preprocessed several times. Name-entity recognizers are run to replace all names with a few generic tokens (person, location, etc.). A syntactic parser is run to produce a dependency tree for each document. The dependency trees are normalized to further reduce noise and variation, and finally a series of tuple are extracted for each clause (containing the subject, verb and object). The seeds provided for each class are in the form of these tuples: as an example, the experiment was run on Wall Street Journal articles, one class to put documents in was labeled Management Succession, and its seed patterns were [Company appoint Person] and [Person quit]. Using these seed patterns, the learner can attempt to divide the set of documents into Management Succession documents, and not (or in the case of the basic learner, between Management Succession, and the n other classes each with their own seeds). After doing so, tuple matches can be found, which allows the learner to further understand which tuples are relevant to the class, and find new documents (or shuffle existing ones around) to properly categorize them.

The problem with the basic learner described (and the main focus of the paper) is that the learner has no obvious stopping point. It will continue to learn until some hard coded halt is called, or it has degraded its quality of learning so much that the precision and recall of the classified documents will suffer drastically. To counterbalance this, the authors propose using multiple learners for each class simultaneously. At each iteration, learners can tag one (or some x number of) document(s); this claims these documents as part of the learners specific class. They can be used to further grow the seed tuples, but for all intents and purposes they are claimed and can't be claimed by other learners (of course with enough weight behind the tuples they can be shuffled, but in principle they have found a home). Once the learners can't pick any more they stop learning. Once all learners but one have stopped, the algorithm halts. This provides a solution to the degradation of learning quality, and in the experiments showed promising results.

Discovering Knowledge in Texts for the learning of DOGMA-inspired ontologies [Reinberger and Spyns, 2004]

The authors introduce the problem of learning semantically rich ontology knowledge from text, and describe some work done in the field, and give some discussion on this differences (in both execution, and results) of using an automated, versus a domain-expert enhanced method to ontology learning. They make a detailed list of tasks that are potentially involved in ontology learning, which include:

-collecting and preprocessing a corpus
-discovering sets of matching words and expressions
-validating these sets through human intervention
-discovering sets of semantic relations
-validating these sets
-formalizing the learned knowledge

The DOGMA (Developing Ontology-Guided Mediation for Agents) ontology system is introduced. There are several methodologies behind the DOGMA approach that define the system: "a text is representative if it embodies by definition or by facts relevant domain knowledge". Also, it is pointed out that a difference between "must" and "may" when modeling a law is very important.

For the experiment directly, two assumptions are made: selectional restrictions and co-composition are assumed in the texts. They use a shallow syntactic parser (which provides tokenisation, POS tagging, etc.). They also chose to use a medical domain, particularly Medline, and extracted two corpuses: one with texts matching "hepatitis A" and "hepatitis B", and a second matching "blood". Both corpuses contained several million words.

After syntactic tagging, a clustering algorithm produces a set of verb-term relations. They are organized into classes, and are associated with a different term each. The clustered terms are now known to share a semantic relation, but finding that information requires a second step. Prepositional structures from text bearing the clustered terms are also clustered. The type of relation can now be extracted (if the preposition is specific enough). Overall, the experiment yielded between 3%-17% precision, and between 9%-41% recall.

How Similar Is It? Towards Personalized Similarity Measures in Ontologies [Berstein et al., 2005]

The authors survey the field of ontological similarity measurements, and identify two broad categories: edge based similarity measures, and node based similarity measures. Edge based methods follow the edges from one concept to the other, penalizing each move. This is similar to the method OntoSearch uses, however the authors noted (obviously having not heard of OntoSearch) that this method

has a primary flaw: each edge has a similar weight, and can thus cause odd results when traversing paths that are less important than others. However, they did point out that this method is the most obviously logical one, when dealing with ontologies of easily taxonomized classes (such as animal kingdoms). The second method, node based, looks at the defined properties of the node and compares them with other defined properties. The authors mention that often this method is similar to bag of words, and thus has a flaw that it the properties are being compared as literal strings, and similarities such as ontological closeness are not being observed.

After identifying the main methods of similarity calculation, the authors present five published means, and conduct an experiment. They present human subjects from various fields with lists of concepts (in context), asking them to rank their similarity, thus providing a gold standard. Then they ask the same of the five metrics, and compare the results. They found that overall the humans did not completely agree, but neither did the metrics. In fact, they were able to cluster the apparently disparate results, showing that the disagreements made by the humans and by the algorithms were very similar. They concluded that the best method for similarity construction must be personalized to your application.

Investigating Semantic Knowledge for Text Learning [Ankolekar et al., 2003]

The paper describes several experiments attempting to combine features found in an ontology with word feature sets when applied to text-classification problems. The researchers wanted to know if adding in features from an ontology would increase the amount of properly classified documents.

For text classification, they used a series of Bayesian methods: Naive Bayes (or zero feature) classifiers, 2-feature classifiers (where the features were extracted from standard word feature extraction methods... chi-squared, etc.), and 2-feature classifiers (where the features came from relations in a specially constructed ontology).

The ontologies were hand-made to suit the domain texts (although the authors admitted that some of the results could be skewed by improper domain knowledge used to create the ontologies). Relations were kept as generic, meaning that no specific relations were applied between concepts; just the fact that a relation exists (so this created a feature) was all that was noted.

Overall, the addition of an ontology for feature extraction provided better results, however, when applying those features, along with traditional extraction methods, the classification results were not as desirable. The authors verdict is the use of ontologies for feature extraction in text classification exists, but the ontologies must be tailor made to the text documents in order to be of assistance.

Joint Learning Improves Semantic Role Labeling [Toutanova et al., 2005]

The authors present a method for improving semantic role labeling by building a joint model of argument frames, which include features that model the interactions of the frames into log-linear models. Their goal, given a sentence and target verb, is to return all fillers of semantic roles, appropriately labeled. The existence of a syntax parser is assumed (so their experiments rely on the 2004 release of PropBank, which has syntactic trees already annotated).

The authors divide the task into two subtasks: identification, where they classify nodes in the dependency tree as either an argument, or a non-argument, and classification, where all arguments must be labeled with their correct semantic role. The authors create a dynamic programming algorithm for labeling. After creating all local models (models that label nodes in the tree independently of other nodes), they can be combined to form joint models (by concatenating templates of features together).

Learning Semantic Classes for Word Sense Disambiguation [Kohomban and Lee, 2005]

The authors present an alternative method to the WSD problem: that of converting a word instance into a generic semantic class. The authors use three classifiers, and combine their results in a voting, and weighted voting system, to select the resulting semantic class for a word sense.

Using a publicly available corpus (SemCor), and task data (Senseval English), the three classifiers were constructed: the first used local context (n words to the left and right), the second used parts of speech (n tags to the left and right), and the third used syntactic relations with the word. In the case of the third, syntactic patterns are either collocation (features that connect one word to another) and relational (direct grammatical relations between words).

The results of the experiments showed promise, being on par or slightly improved in the area of recall against previous, completely different attempts to classify words in the task set (~65%).

Learning Taxonomic Relations from Heterogeneous Evidence [Cimiano et al., 2004]

The paper presents four methods of automatically learning is-a relations for a specific domain (travel in this case), evaluates each method against an expert created gold standard, and then shows how combining the methods produces significantly better results.

The first method, using Hearst-patterns in a large text corpus, is done by scanning the text, looking for matches to NP0 such as NP1, and other such patterns. The text has to be POS tagged, at which point the frequency of such patterns (which was quite low) can be used to construct reasonably accurate is-a relations.

The second method, using hypernymy in WordNet, is performed by calculating a weighted overall distance between all taxonomic paths between the two target concepts. If a hypernymic path exists, WordNet is claiming an is-a structure. However, this has to be taken with a grain of salt, as the authors point out, when attempting to construct a domain specific ontology. WordNet is too general, and ambiguity in senses led to the authors having generally poor results from this method.

The third method, vertical-relations, is one where two concepts share root words, and one is modified with adjectives, thus indicating an is-a relation. The example given is conference, and international conference. Using this method, although quite simple, had very high precision, but very low recall.

The final method, using the GoogleAPI to query for Hearst structures and count the frequency of them had decent recall but low frequency. In this method, each pair of concepts created a series of Google queries, e.g. T1 such as T2. The frequency of the results was weighed against the frequency of just a search on T1 to produce a reasonable estimation of how closely T1 and T2 were related.

The authors then took all four methods, and combined them in two ways, taking the mean of the results of the four values above, and one in which they took the maximum of the four results. The difference between these two strategies was negligible, however either one did improve over each of the previous methods alone.

Measuring Semantic Similarity by Latent Relational Analysis [Turney, 2005]

The paper describes a method of comparing similarity behind the semantic meaning of two concepts using latent relational analysis. Latent relational analysis is a process that looks at the frequencies of relational words and phrases (such as of, for, at), and constructs a sparse matrix from them. SVD is run on the matrix and the results are combined to produce a 0 to 1 value of similarity.

The idea is to be able to judge how similar two word pairs are to each other; for example mason:stone and carpenter:wood have highly similar relations.

The system works as follows: given three inputs, a search engine with a large corpus, a thesaurus of synonyms, and an efficient implementation of SVD, each word pair is expanded to multiple word pairs using synonyms from the thesaurus (this results in 1 original word pair, and 2n alternate word pairs, n synonyms for each original word).

For each of the alternate pairs associated with an original pair of word, the search engine is queried for the number of phrases that start and end with the alternate pair of words. The top scores (3 in the experiment) are retained.

For all pairs, search the corpus for phrases starting and ending with the pair of words; these phrases denote the similarity among the pairs. Now, replace the intervening words in each phrase with unique wildcards; these wildcards create patterns... the top m patterns (4000 in this case) are retained.

The sparse matrix is constructed by making each pair (and its inverse) a row, each pattern (and its inverse) a column, and each cell the frequency of that pattern in that pair.

Entropy is calculated, and then SVD. Now, any two pairs A:B, C:D can have their similarity calculated by taking the average of the cosine values of each row containing A:B, and each row containing C:D. This value is the LRA of A:B-C:D.

Overall, on SAT level questions, LRA performed as well as the average college bound high school student (57% accuracy).

Measuring the Specificity of Terms for Automatic Hierarchy Construction [Ryu and Choi, 2004]

In this paper the authors describe a method for automatically constructing ontological hierarchies of words by assigning a specificity value to the words. The assumption is that a word that is both similar to another, and more specified, is a child of the other word. The method involves producing a positive real number to represent the specificity of a term X. The system uses three modules, or managers, to produce a specificity-based ontology: the term manager contains a list of all terms in the corpus, and evaluates the specificity values passed to it by various methods. The statistics manager calculates various statistics on the corpus and provides them where needed. The specificity manager provides the calculations of specificity described below.

Using a corpus of medical texts, and focusing on a sub-tree of the MeSH thesaurus, the authors develop several methods of calculating specificity, and apply it to a gold standard human created ontology for the target terms (436 disease names). A domain specific corpus like this one has its own feature sets; adding new features to existing terms creates more specific terms.

To calculate specificity, composition and context can be used. Composition means the parts of the term produce the meaning of the term. Context means the distribution of surrounding words as compared with the context of known terms produces a meaning of the term.

In one method, the authors examine how words like diabetes mellitus must be less specific, and therefore a parent of, words like insulin-dependent diabetes mellitus. This method proves to fail in places where semantically the meaning is in no way similar to the word count of the unit words.

The second method takes this shortcoming into account, and uses a parser to attempt to extract a more meaningful specificity through semi-semantic methods. This method does prove to be significantly better, but also fails to capture certain phenomena that the first method picked up on.

The final method, a hybrid of the first two, assigns a higher specificity when both methods return high specificity. This method had the highest overall accuracy, being over 80% similar to the gold standard.

Ontology Express: Statistical and Non-Monotonic Learning of Domain Ontologies from Text [Ogata and Collier, 2004]

The authors present a system of extracting taxonomies of domain specific ontologies through statistical analysis of related corpora. They present many common sense approaches with in-house terms; i.e. typing information refers to finding the type, or IS-A, of a concept. The name of each concept (or semantic class) is given as the name of the type of entity being learned in the corpora.

The goal is to facilitate construction of ontology trees, but in no way to fill concepts with meaningful property/values. The work is not meant to replace domain experts, nor do the authors claim that IS-A relations among assumed knowledge will be learned (if it is not in the text specifically, it will not be added to the ontology!).

Methods used include finding definitions: token name is a type name. Exemplification: token name and other type name. Exception: type name other than token name. There are multiple heuristics used for each method.

Ontology and Lexicon Evolution by Text Understanding [Hahn and Marko, 2002]

This paper presents a method of simultaneously learning a new semantic and corresponding lexical concept through text understanding. The authors syntactically parse the text, and using a semi-semantically rich lexicon, and a method similar to selectional restrictions, attempt to identify where in the is-a hierarchy a new term belongs, and then map a lexical entry to it.

The example given is: The R600MX of the company Vaio costs approximately 1600 Euros. In the example, the word Vaio can be syntactically parsed as a noun or an adverb. Using the existing semi-semantic rules, only the noun is valid when taking in the restrictions of company (for the noun) and costs (for the adverb). Seeing as noun is the only option, Vaio is parsed as a company type, and an is-a relation is created. Using this same methodology, R600MX is seen as a product. The methodology was tested on domain specific texts using a domain-specific knowledge base.

Semantic Role Labeling Using Different Syntactic Views [Pradhan et al., 2005]

The authors present a way to enhance automatic semantic role labeling with SVMs by extending the feature sets used and by using different syntactic parsers (or views) in a complimentary way.

The authors started by introducing a baseline system, constructed using the Charniak syntactic parser. The corpus used was the 2004 release of PropBank (which is constructed from semantically hand-labeled Penn TreeBank excerpts). The baseline system had a series of features defined (to be used by the support vector machine for class or role labeling). Features included such things as predicate lemma, voice, position of the constituent in relation to the predicate, etc. Three tasks were used to judge performance: argument identification (identifying constituents in a sentence that are semantic arguments to the predicate), argument classification (assigning argument labels to arguments of a predicate) and a combination of the first two tasks.

After creating and judging the baseline, a variety of experiments were conducted, which on the whole showed improvement over the baseline. More features were added to the feature set. Features in the feature set were pruned to fit the problem better. And most interestingly (and the main focus of the paper), two other syntactic parsers were used in place of the Charniak parser: Minipar, and a Chunk-based parser were implemented. Each parser had nuances that were dealt with by the researchers; the three parsers combined with expanded and filtered feature sets were able to more accurately label semantic features than the baseline alone.

7.2 Annotated Papers for the Development of NLP Tools

ConceptNet - A Practical Commonsense Reasoning Tool-Kit [Lie and Singh, 2004]

A full-blown treatise on ConceptNet, the authors cover from start to finish everything there is to know: conception, implementation, comparison, and evaluation. ConceptNet fills a void in the current pile of large semantically annotated corpora but emphasizing common-sense semantic links. Starting with the Open Mind Common Sense corpus (which details such things as: getting fired means not having money, which is used for food and shelter, which are necessary to survive... thus getting fired may suggest feelings of fear, anger and sadness), an online effort compiling fill-in-the-blank sentences about common sense world data from volunteer users around the world, ConceptNet has crunched this data into a semantic ontology (extracting expressions, normalizing them, the relaxing property/value pairs as needed). The resulting data structure consists of 1.6 million edges and over 300,000 nodes (semi structured English fragments, which are related through an ontology of 20 semantic relations).

In the meanwhile, a NLP (based on MontyLingua) system has been developed to assist researchers in using the ever-growing corpus of semantic common sense. A series of API-level functions (FindPathsBetweenNodes, GetContext, GetAnalogousConcepts) allow the researcher to scan the data in a variety of ways, mostly from the point of view of an input sentence or text. Finding relevant concepts, formulating analogies (apple and cherry share the same back-edges: sweet, red, fruit), projection (if A->B, and B->C, then A->C; this allows for speculation in a temporal sense, as well as physical enclosure), topic gisting (suggesting what a text on the whole could be about), disambiguation, classification, unknown concept identification, etc.

Heavy comparisons (in the theoretical sense... the point was made that a numerical comparison is not appropriate) to WordNet and Cyc are made; the details of what functionality each serves as a corpus, as well as the methodology of their construction are compared to ConceptNet. ConceptNet is evaluated very roughly, and the authors have no problem admitting that evaluation is difficult on this type of system. They suggest various schemes using human evaluation. They also discuss the evaluation of Open Mind Common Sense, as it is the underlying data of ConceptNet. Human judges were used to evaluate a sample of the corpus, with results being: 75% of items as largely true, 82% as largely objective, 85% as largely making sense, and 84% as knowledge someone would have by high school. The paper closes with a list of research projects that have used ConceptNet.

Evolving GATE to Meet New Challenges in Language Engineering [Bontcheva et al., 2004]

The authors give a complete rundown of the GATE system (v2), explaining in details every change and addition that was made from the first version of the system.

In addition to keeping with the theme of GATE (extensible do-it-yourself NL tools), the GUI tools are now also extensible, and a variety of knowledge types and processing capabilities have been added.

Support for ontologies and lexicons now exist, along with full integration of ML algorithms (using WEKA). GATE supports a variety of formats including: plain text, HTML, XML, RTF, and SGML. These formats are automatically converted inside GATE to a single model. GATE also supports a distributed resources model, opening an easy method to create a client server system, where the language resources are stored on a central server. GATE has widened its view of what a language resource is, from just documents/corpora to include lexicons, ontologies, and thesauri. These new types of resources are integrated, such that the new processing capabilities of GATE (ML, distributed resources, etc.) are compatible.

Lexicons are most extensively supported through WordNet (as it is commonly used). Protégé provides the backbone for ontology editing support, showing that GATE is willing to open to other generic knowledge tools for good solutions to its new feature sets.

GATE: An Architecture for Development of Robust HLT Applications [Cunningham et al., 2002]

GATE is a framework design to facilitate development of research using NL tools. The GATE system ties in a variety of NL resources, such as syntactic and semantic annotation, POS tagging, and reference resolution, along with a variety of ways to develop knowledge resources, into an all-encompassing API. The API is broken into three resource types: language resources (LRs), lexicons, corpora, etc., processing resources, such as those listed above, and visual resources, reusable GUI components that tie the whole thing together.

The developer is able to construct a custom fit NL processor through an interface that allows connections of existing, or home made resources. Developers can plug in, and then remove processors, facilitating the comparison of results with and without a certain NL module.

The system also allows for evaluation, by providing statistical analysis of results before and after a change to the processor has been implemented. The system also supports multilingual applications.

Although GATE is not directly a NLP system, it is a suite of tools that allows one to quickly construct a NLP system, tailor made to their specifications. This reduces the overhead of integration of NL tools into a full NLP system by providing standardized mechanics in Java and XML.

As a consequence of not being a system designed for a specific task, evaluation in the paper is described as an additional set of tools GATE provides to evaluate systems it creates (rather than a heuristic that was used to evaluate GATEs usefulness itself).

The Annotation Graph Toolkit [Maeda et al., 2006]

This paper presents the work done on a system to allow user to easily annotate time-series data. The system is constructed on an API, allowing users to quickly create new tools specific to their needs. The API provides for a client-server method of data storage (using SQL on the server side), and uses IDL as a widget generation tool.

An annotation graph is an efficient and expressive data model use for linguistic annotations of time-series data. This system allows users (through the API) to create anchors in the time-series data. Once two anchors have been created, an

annotation can be added that spans them. Annotation objects can have any number of features associated with them, and the system can be queried for annotations that have certain feature types or feature values.

The Berkeley FrameNet Project [Baker et al., 1998]

The paper reports on the efforts of the FrameNet project in its second year. FrameNet is described in a theoretical sense: a database containing descriptions of the semantic frames underlying words, along with the valence representation of accompanying words and phrases, as well as annotated examples. FrameNet's primary goal is to produce a machine-readable database of semantic knowledge; the knowledge is to be constructed by humans, thus the FrameNet software is a set of tools for assisting this acquisition, along with a set of tools for using (or querying) the existing data.

FrameNet's lexicon is composed of entries which have four fields: a dictionary definition (for humans), a series of formulas which represent the morphosyntactic ways the entry can be used (syn-struc), links to semantically annotated examples, and links to the frame database (as well as other external semantic sources such as WordNet).

FrameNet's frame database is composed of entries, which contain the conceptual structure of the frame, and descriptions of their contained elements.

FrameNet, in year two, consists of several thousand such entries (although many are stubs), with a defined and functional set of software tools to assist in further developing the database. Much of the software is off the shelf, and is held together using dynamic web based technologies, such as perl. The system allows for people to define, describe, and annotate new frames (allowing for new concepts and lexical mappings).

The Evolution of Protégé: An Environment for Knowledge-Based Systems Development [Gennari et al., 2003]

The paper describes the motivation behind each incarnation of the Protégé system, from its ancestor Opal, to Protégé I, Protégé II, Protégé/Win, and Protégé 2000. Each system builds on the findings and usability issues of the others, but each is a complete redo in terms of the system architecture.

Protégé originally existed to support modeling expert systems from a knowledge base. This was expanded to allow the knowledge base to be defined; later the knowledge base became an ontology, which opened up further end user support. By the time Protégé 2000 hit, it was an all-in-one ontology and expert system design and test system.

The Opal system, developed in 1987, was a knowledge-acquisition tool designed to let domain experts enter some domain specific knowledge into an advice system (Oncocin). The domain experts (physicians in this case) would enter the knowledge in the form of protocols by filling out graphical forms.

Protege-1 (1989) was an abstraction of Opal. The idea was, instead of having to have a new specific tool, like Opal, created for every domain that needed knowledge entered by experts, that rather a tool would be generated from the structural knowledge of the domain. Knowledge engineers would create the basis for the knowledge base, and through Protege-1, a tool would be generated that domain experts could use to augment the specifics of the knowledge base.

Protege-2 introduced a way to generalize the algorithms, or problem solving methods, used on a set of domain knowledge. In other words, after the basis has been created, the tool generated, and the knowledge populated, a series of number-crunching methods could be used and then re-used on the data. In order to support re-usability, the data in the domains would need to be viewed and stored in an ontology.

Protégé/Win was developed to expand the use of Protege-2 by making the system available on the Windows operating system (as opposed to the NeXTStep OS). Protégé/Win also enhanced the modularity and re-usability of the ontology data structure, and increased the functionality of the toolset provided.

Protégé 2000 compares very well with DEKADE. It is built on Java, and provides users with an API for data access and manipulation. The API is extensible, so the users can ditch the default functionality for something more custom made. The interface is a tabbed pane view, allowing users to add and remove tabs at will, and develop custom tabs when needed. The application supports a variety of data view and storage methodologies, as well as various editors and browsers.

The FrameNet Data and Software [Baker and Sato, 2003]

This paper describes the systems level work behind FrameNet in a brief format. The way frames and lexical units are constructed (including what they consist of) is discussed. Also, the API under construction, the GUI for acquisition, and the SQL database supporting the knowledge is explained.

The basic data type is the semantic frame. It defines an event or state, and contains properties, or frame elements. One sense of a word is directly associated with a frame, and thus makes up a lexical unit. The lexicon was developed to be readable by both humans and machines. At print, the FrameNet II database consists of 450 frames, and more than 3,000 frame elements, and over 7,500 lexical units with example sentences and annotations. The data is stored in a server-side MySQL database, and accessible through a client interface.

The FrameNet Database and Software Tools [Fillmore et al., 2002]

> The primary subject of the article is to describe the various tools and supporting data structures available to FrameNet at the time of publication. The authors, however, spend a good amount of the paper describing what is to be annotated by FrameNet, and what is not. Valid annotation targets include: for verbs, nouns, adjectives and prepositions, their post-head complements. For nouns, frame-relevant possessive determiners are considered valid targets.
>
> Concerning the tools, FrameNet uses a mixture of off the shelf, and in house tools to assist in the annotation. An in house frame editor, a GUI written in Java, is supported by a MySQL database. FrameSQL, a project written by Sato in Japan, facilitates queries through the database. A web-based Lexical Entry report is available, which shows the lexical entry in full, along with definition and valence patterns (and relevant links to other data).

7.3 Further Readings

The following works will also be studied as preliminary work to any final written documentation (these are not the only works intended to be read, but merely what is currently "on the plate").

A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge [Landauer and Dumais, 1997]
Automatic Labeling of Semantic Roles [Gildea and Jurafsky, 2001]
Automatically Learning Qualia Structures from the Web [Cimiano and Wenderoth, 2005]
Commonsense Reasoning in and over Natural Language [Liu and Singh, 2004]
Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations [Pantel and Pennacchiotti, 2006]
The State of the art in Ontology Learning: A Framework for Comparison [Shamsfard and Barforoush, 2003]
Using Access Paths to Guide Inference with Conceptual Graphs [Clark and Porter, 1997]

8. Conclusion

This document presented a proposed system for automatic learning of unknown words by, and for, a semantic text analyzer. Motivation for such a system was given, with emphasis on the acquisition bottleneck: acquiring quality, broad coverage world knowledge is time consuming and error prone. The need for such a wealth of knowledge was motivated with examples of the intended use of a quality semantic text analyzer: question-answering systems, topic gisting, opinion extraction, etc. The proposed system aims to reduce the strain of the acquisition bottleneck by using a bootstrapped, existing semantic analyzer to extract information on an unknown concept from an open corpus (the web), and compile this information into a well formed, new data structure for inclusion in the systems own existing static knowledge resources. The eventual quality of such a system would be shown by producing semantically annotated texts concerning

an unknown word, then learning the word, and reproducing those same annotations (and comparing the qualitative differences).

9. References

Ankolekar, Anupriya, Young-Woo Seo and Katia Sycara. *Investigating Semantic Knowledge for Text Learning*. In Proceedings of ACM SIGIR Workshop on Semantic Web. 2003.

Baker, Collin F., Charles J. Fillmore and John B. Lowe. *The Berkeley FrameNet Project*. In Proceedings of COLING-ACL. Montreal, Canada. 1998.

Baker, Collin F., and Hiroaki Sato. *The FrameNet Data and Software*. Poster and Demonstration at Association for Computational Linguistics. Sapporo, Japan. 2003.

Bernstein, A., E. Kaufmann, C. Buerki and M. Klein. *How Similar Is It? Towards Personalizing Similarity Measures in Ontologies.* In Proceedings of the 7th Internationale Tagung Wirtschaftsinformatik, pp. 1347-1366. 2005.

Bontcheva, Kalina, Valentin Tablan, Diana Maynard and Hammish Cunningham. *Evolving GATE to Meet New Challenges in Language Engineering*. In Natural Language Engineering 10 (3/4), pp. 349-373. 2004.

Buitelaar, P., S. Handschuh and B. Magnini (eds.). Proceedings of the ECAI-2004 Workshop on Ontology Learning and Population (OLP). Valencia, Spain. August, 2004.

Buitelaar, P., P. Cimiano, M. Grobelnik, M. Sintek. *Ontology Learning from Text.* Tutorial at ECML/PKDD, Porto, Portugal. October, 2005.

Charniak, E., M. Berland. *Finding parts in very large corpora*. In Proceedings of the 37th Annual Meeting of the ACL, pp. 57-64, 1999.

Cimiano, Phillipp, Aleksander Pivk, Lars Schmidt-Thieme and Steffen Staab. *Learning Taxonomic Relations from Heterogeneous Evidence*. In Proceedings of the ECAI 2004 Ontology Learning and Population Workshop. 2004.

Cimiano, P., G. Ladwig and S. Staab. *Gimme' the context: Context-driven automatic semantic annotation with c-pankow*. In Proceedings of the 14th WWW, ACM. 2005.

Cimiano, P., J. Wenderoth. *Automatically Learning Qualia Structures from the Web*. In Proceedings of the ACL Workshop on Deep Lexical Acquisition, pp. 28-37. 2005.

Clark, Peter and Bruce Porter. *Building Concept Representations from Reusable Components*. In Proceedings of AAAI 97, pp. 369-376. 1997.

Clark, S., D.J. Weir. *Class-Based Probability Estimation Using a Semantic Hierarchy*. Computational Linguistics, 28(2), pp. 187-206, 2002.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva and Valentin Tablan. *GATE: An Architecture for Development of Robust HLT Applications.* In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL 02). Philadelphia, PA. July, 2002.

English, Jesse and Sergei Nirenburg. *Ontology Learning from Text Using Automatic Ontological-Semantic Text Annotation and the Web as the Corpus*. In Proceedings of the AAAI 2007 Spring Symposium Series on Machine Reading. March, 2007.

Fillmore, Charles J., Collin F. Baker and Hiroaki Sato. *The FrameNet Database and Software Tools.* In Proceedings of the Third International Conference on Language Resources and Evaluation (LREC), pp. 1157-1160. Las Palmas, Spain. 2002.

Gennari, John H., Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubezy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. International Journal of Human-Computer Studies, Volume 58, Issue 1, pp. 89-123. January, 2003.

Gildea, Daniel, and Daniel Jurafsky. *Automatic Labeling of Semantic Roles.* Computational Linguistics 28(3) 2002, pp. 245-288 14. 2002.

Girju, R., D. Moldovan. *Text Mining for Causal Relations.* In Proceedings of the FLAIRS Conference, pp. 360-364, 2002.

Hahn, Udo, and Kornel G. Marko. *Ontology and Lexicon Evolution by Text Understanding*. ECAI 02 Workshop on Machine Learning and Natural Language Processing for Ontology Engineering. 2002.

Java, A. et al. *SemNews: A Semantic News Framework*. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06*)*. 2006.

Java, A., et al. *Using a Natural Language Understanding System to Generate Semantic Web Content.* Accepted to the International Journal on Semantic Web and Information Systems (IJSWIS), 2007.

Kilgarriff, Adam, and Gregory Grefenstette. *Introduction to the Special Issue on the Web as a Corpus.* Computational Linguistics, Volume 29, pp. 333-347. 2003.

Kohomban, Upali S., and Wee Sun Lee. *Learning Semantic Classes for Word Sense Disambiguation*. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 34-41. Ann Arbor, MI. 2005.

Landauer, Thomas K., and Susan T. Dumais. *A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge*. Psychological Review, v104 n2, pp. 211-240. April 1997.

Lie, H. and P. Singh. *ConceptNet – A Practical Commonsense Reasoning Tool-Kit*. BT Technologies Journal, Vol. 22, No. 4, pp. 221-226. October, 2004.

Maeda, Kazuaki, Steven Bird, Ziaoyi Ma and Haejoong Lee. *The Annotation Graph Toolkit*. In Proceedings of the 1st International Conference on Human Language Technology Research, pp. 1-6. San Diego, CA. 2006.

Navigli, Roberto, Paoloa Velardi, Alessandro Cucciarelli, and Francesca Neri. *Automatic Ontology Learning: Supporting a Per-Concept Evaluation by Domain Experts*. In Proceedings of the Workshop on Ontology Learning and Population (OLP), in the 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 1-6. Valencia, Spain. August, 2004.

Navigli, R., P. Velardi. *Enriching a Formal Ontology with a Thesaurus: an Application in the Cultural Heritage Domain.* Proceedings of OLP-06. 2006.

Nichols, E., F. Bond, T. Tanaka, F. Sanae and D. Flickinger. *Multilingual Ontology Acquisition from Multiple MRDs.* Proceedings of OLP-06. 2006.

Nirenburg, S., V. Raskin. *Ontological Semantics. SERIES*: Language, Speech, and Communication, MIT Press. 2004.

Nirenburg, Sergei, Donald Dimitroff, Jesse English and Craig Pfeifer. *Three Experiments on Mining the Web for Ontology and Lexicon Learning*. Unpublished Technical Report. University of Maryland, Baltimore County. February, 2007.

Ogata, Norihiro and Nigel Collier. *Ontology Express: Statistical and Non-Monotonic Learning of Domain Ontologies from Text*. In Proceedings of the Workshop on Ontology Learning and Population at the 16th European Conference on Artificial Intelligence (ECAI-2004). Valencia, Spain. 2004.

Onyshkevych, B. *Ontosearch: Using an ontology as a search space for knowledge based text processing*. Unpublished PhD Dissertation. Carnegie Mellon University. 1997.

Pantel, P., M. Pennacchiotti. *Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations*. In Proceedings of Conference on Computational Linguistics / Association for Computation Linguistics (COLING/ACL-06). 113-120. 2006.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin and Daniel Jurafsky. *Semantic Role Labeling Using Different Syntactic Views.* In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 581-588. Ann Arbor, MI. June, 2005.

Pustejovsky, J. *The Generative Lexicon.* Cambridge/London: MIT Press. 1995.

Reinberger, Marie-Laure and Peter Spyns. *Discovering Knowledge in Texts for the learning of DOGMA-inspired ontologies*. In Proceedings of the Workshop on Ontology Learning and Population (OLP), in the 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 19-24. Valencia, Spain. August, 2004.

Ryu, Pum-Mo and Key-Sun Choi. *Measuring the Specificity of Terms for Automatic Hierarchy Construction*. In Proceedings of the 16th European Conference on Artificial Intelligence, Workshop on Ontology Learning and Population (ECAI-2004). Valencia Spain. August, 2004.

Shamsfard, Mehrnoush and Ahmad Abdollahzadeh Barforoush. *The State of the art in Ontology Learning: A Framework for Comparison*. The Knowledge Engineering Review, Volume 18, Issue 4, pp. 293-316. December, 2003.

Stone, A. *EBIDS-SENLP: A System to Detect Social Engineering Email Using Natural Language Processing*. Unpublished Master's Thesis, University of Maryland Baltimore County. 2007.

Toutanova, Kristina, Aria Haghighi and Christopher D. Manning. *Joint Learning Improves Semantic Role Labeling*. In Proceedings of the 43rd Annual Meeting on Association for Computation Linguistics, pp. 589-596. Ann Arbor, MI. June, 2005.

Turney, Peter D. *Measuring Semantic Similarity by Latent Relational Analysis*. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), pp. 1136-1141. Edinburgh, Scotland. July, 2005.

Yangarber, R. *Counter-Training in Discovery of Semantic Patterns*. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003). 2003.