Autonomous Classification of Knowledge into an Ontology

Matthew E. Taylor, Cynthia Matuszek, Bryan Klimt, and Michael Witbrock

mtaylor@cs.utexas.edu
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188

{cynthia, bklimt, witbrock}@cyc.com
Cycorp, Inc.
3721 Executive Center Drive
Austin, TX 78731

Abstract

Ontologies are an increasingly important tool in knowledge representation, as they allow large amounts of data to be related in a logical fashion. Current research is concentrated on automatically constructing ontologies, merging ontologies with different structures, and optimal mechanisms for ontology building; in this work we consider the related, but distinct, problem of how to automatically determine *where* to place new knowledge into an existing ontology. Rather than relying on human knowledge engineers to carefully classify knowledge, it is becoming increasingly important for machine learning techniques to automate such a task. Automation is particularly important as the rate of ontology building via automatic knowledge acquisition techniques increases. This paper compares three well-established machine learning techniques and shows that they can be applied successfully to this knowledge placement task. Our methods are fully implemented and tested in the Cyc knowledge base system.

Introduction

Ontologies are an increasingly important tool in knowledge representation, as they allow large amounts of data to be related in a logical fashion. Current research has concentrated on automatically constructing ontologies or large bodies of formally represented knowledge (Fortuna et al., 2005), merging ontologies with different structures (Masters and Güngördü 2003), and optimal mechanisms for ontology building (Witbrock et al., 2003). In this work we consider the related, but distinct, problem of how to automatically determine *where* to place new knowledge into an existing ontology.

As formally represented knowledge bases grow larger and cover a greater range of domains, problems of internal consistency and redundancy become significant (Lenat, 1998). As an example, the commonly heard statements "Count Dracula was a vampire" and "vampires do not exist" are contradictory, unless one is aware of the implicit context shift. One statement is being made in a factual, real-world sense, while the other occurs in a specific, fictional context. These different contexts, which we refer to as *reasoning contexts*, have differing background axioms that apply to assertions made in those contexts. An ontology that has no mechanism for specifying the context of the statements being made must either include contextual information with each assertion, which rapidly becomes unwieldy, or be forgiving of inconsistency.

We demonstrate automatic placement of knowledge into the Cyc system, an ontology-based system designed to capture a significant fraction of human *common sense*—the kind of background knowledge that human learning agents can be assumed to have when presented with a task.[2] The Cyc knowledge base is a suitable target for two primary reasons: it is large, both in number of assertions and in the range of domains it contains; and it already contains a concept of reasoning contexts, called *microtheories*, suitable for classifying knowledge into.

The Cyc *knowledge base* (KB) is made up of *assertions*, formally represented facts of varying levels of complexity (Matuszek et al., 2006), which are asserted into a hierarchy of distinct reasoning contexts called microtheories (or *Mts*). Microtheories serve several purposes. The primary purpose is to allow each assertion to be correctly contextualized along several dimensions, such as temporal qualification or domain of discussion. From an engineering perspective, making assertions in microtheories allows the background assumptions that apply to a particular domain to be stated only once, and makes it far easier to construct a logically consistent knowledge base.

Microtheories in Cyc are arranged hierarchically. Very high-level microtheories capture information about broad distinctions such as physical, temporal, and fictional reasoning, and are then subdivided into progressively more specific contexts. An assertion placed into a microtheory must be true in the upward closure of that microtheory (i.e., that microtheory and more general microtheories that subsume it). Cyc currently contains about 4.6 million assertions in 23,627 microtheories, which have an average of approximately three microtheories directly below them in the hierarchy.

[2] Once common-sense knowledge is available in a format that can be programmatically understood and reasoned over, learning tasks that rely on that background knowledge, such as reading from a textbook, become more feasible (Lenat, 1995).

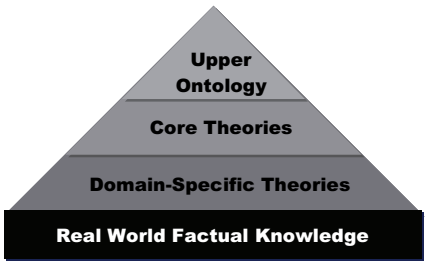

Figure 1: The Cyc Knowledge Base Mt (Microtheory) Hierarchy. A small number of very broad reasoning contexts contain high-level, abstract knowlege, and fan out into progressively more specific contexts.

Description of Task

Historically, all assertions have been added to Cyc by *ontologists*, human knowledge engineers who are familiar with the Mt structure and able to determine the appropriate Mt placement (domain and level of generality) of that an assertion. This is not ideal for several reasons. It is time-consuming; choosing the right microtheory can easily take several minutes for a trained ontologist who is familiar with the Cyc ontology and experienced in how best to organize knowledge for maximum utility. A technique able to autonomously place knowledge into the KB therefore has the potential to save an amount of time proportional to the amount of knowledge being added.

More importantly, with more automated ontology-building techniques such as automated fact gathering from the World Wide Web (Etzioni et al., 2004; Matuszek et al., 2005; Shah et al., 2006), an important goal is to add knowledge into contextualized knowledge bases quickly and with little or no human interaction. It will become infeasible to rely on trained ontologists to label all asserted knowledge once knowledge systems begin generating their own knowledge. The goal of this research, therefore, is to create a classifier that is capable of placing novel knowledge into a hierarchical ontology with both high precision and recall.

Having this automatic classification tool will improve efficient ontology building in multiple ways. When the tool is performing well, it helps make knowledge engineering feasible for less sophisticated users. Automatic classification would also be beneficial for technologies that gather facts automatically or generate novel knowledge by analysis of existing knowledge, such as finding implication rules via ILP (Cabral et al., 2005). In this paper we describe three possible techniques for developing such a tool. These solutions use common statistical machine learning techniques: Naïve Bayes (with and without shrinkage) and Support Vector Machines (SVMs). Experiments empirically demonstrate that automatically determining the placement of facts within a microtheory hierarchy is feasible and able to achieve precision and recall rates of 98%.

Related Work

While classification in knowledge representation is a long-standing area of research, most work has concentrated on the placement of concepts into a hierarchy or ontology (Schmolze and Lipkis 1983), rather than the placement of entire axioms into a knowledge base. In this way, classification of knowledge into a hierarchical ontology is more similar to the classification of text documents into a hierarchy of classes. Using statistical rather than semantic methods in classification of documents is an approach with much significant previous work (Joachims, 1998; Koller and Sahami, 1997; McCallum et al., 1998). Text classification tasks that involve categories arranged into a *semantic hierarchy* in particular are relevant to this task.

In one previous approach (McCallum et al., 1998), a generative Bayesian model was used to predict the classification of text documents into a hierarchy. In this model, *shrinkage* is used to improve the performance of the model on smaller classes by utilizing the structure of the classes. The probability of placing a document in a given class is based not only on the statistics for that particular node in the class tree, but also on other nodes above it in the tree, thus "shrinking" the maximum likelihood estimate of a node towards that of its ancestors.

Another approach to hierarchical text categorization is the so-called "Pachinko machine" (Koller and Sahami, 1997). In this type of algorithm, decisions are made starting at the root of the hierarchy, working down to the leaves. For example, when classifying a cat into a biological taxonomy, a "Pachinko Machine" classifier could first consider "Is 'Cat' a plant, an animal, or neither?" Deciding it is an animal, it could then consider "Is 'Cat' a vertebrate, an invertebrate, or neither?" It continues in this fashion until it classifies 'Cat' into a class with no sub-nodes in the hierarchy.

It has been shown that, when using a single feature set for every node in the hierarchy, a Bayesian pachinko classifier would never perform better (or worse) than a *flat* classifier that considers each node as an independent category (Mitchell 1998). However, this proof does not address the situation in which different features are utilized at each node; thus a Pachinko machine could potentially outperform a flat classifier if features were effectively selected at different nodes. Such feature selection is, however, potentially difficult to do well. As an example, is the presence of the term Mammal useful for classifying terms into the microtheory BiologyMt? Given knowledge of the domain, it clearly is. However, most of the facts asserted in BiologyMt do not mention Mammal, so some statistical feature selection techniques may exclude it. Finding an efficient feature selection algorithm for this context is the topic of current research.

The problem of placing assertions within an ontology is significantly different from text classification in two main ways. First, Cyc's Mt structure is as deep as 50 levels in some domains. Typically, corpora used in text classification experiments are substantially shallower. McCallum et al.'s implementation (1998) is only able to

handle class trees of depth two; even the largest hierarchies used in text classification, such as the Yahoo! taxonomy (Liu et al., 2005), are less than 10 levels deep.

Second, and more important, the formula within any individual assertion contains relatively little data. For instance, an assertion using a binary predicate, such as (isa Cat Mammal), contains only three terms, instead of having an entire document's worth of information. Relatively little previous research has been done on classifying such sparse data in hierarchical contexts.

Approach

In this section we detail the two machine learning methods we use to learn appropriate placement for assertions in an ontology, and describe the data set we use for both training and testing.

Bayesian Model

The first formulation of a generative model to place assertions into Cyc's microtheory hierarchy is based on an earlier Naïve Bayes implementation (McCallum et al. 1998), with two main differences. First, instead of allowing placement only in leaves, we also allow assertions to be placed in intermediate microtheories. Second, the Cyc microtheory ontology is a directed graph, rather than a tree; this requires that we break cycles in Cyc. To accomplish this, our algorithm keeps track of which Mts have already been processed and does not process the same microtheory more than once.

The sketch of our algorithm is as follows: Let there be N assertions in the knowledge base. Each assertion is composed of some number of atomic terms, and we denote the k^{th} term in assertion i as $t(a_i,k)$. Every Mt in the ontology, where Mt j is denoted m_j, is an eligible location for every assertion. Our goal is to find the probability that the i^{th} assertion belongs in a particular Mt, j.

To calculate such a probability, we assume that there is some unknown model, θ, which accurately describes the structure of the KB. Our goal is to calculate $P(m_j \mid a_i ; \theta)$ for all assertions and all Mts. We initially assume that all Mts have a prior probability equal to the number of assertions in that Mt. We then use the number of times a particular term occurs in each Mt to determine the probability that $t(a_i,k)$ occurs in each Mt. After using

$$P(m_j \mid a_i;\theta) = \frac{P(m_j \mid \theta)\prod_{k=1}^{|a_i|} P(t(a_i,k) \mid m_j;\theta)}{\sum_{r=1}^{|Mts|} P(m_r \mid \theta)\prod_{k=1}^{|a_i|} P(t(a_i,k) \mid m_r;\theta)}$$

algebraic simplifications, we can show that the probability of the i^{th} assertion residing in the j^{th} Mt is:

As an additional improvement, we also implement the shrinkage procedure utilized by McCallum et al. To accomplish this, the algorithm utilizes an iterated expectation-maximization (EM) procedure. Full details are presented in the original paper (McCallum et al. 1998).

Support Vector Machines

Support Vector Machines (SVMs) (Vapnik, 1995) are a well-understood method of performing supervised, discriminative machine learning, used for binary classification tasks. An SVM implementation finds a hyperplane in the feature space that will separate a pair of classes by the widest *margin*. SVMs are known to perform well in cases where the proportion of positive and negative examples is highly unbalanced, such as with the leaves of Cyc's ontology. For a given Mt, the positive examples are those assertions that are asserted in that microtheory. The negative examples are *all* other assertions in the knowledge base. SVMs also have a proven track record of dealing with highly sparse data (Joachims 1999), a particularly important feature when each assertion contains only a handful of terms.

Using SVMs for multi-class classification has also been studied (Crammer and Singer, 2001; Tsochantaridis et al., 2004), although in less detail. One common approach is building a "one-vs-rest" binary classifier for each node in the hierarchy (Crammer and Singer, 2001). One implementation of this is SVM-multiclass[3], a freely available multi-class SVM implementation based on (Tsochantaridis et al., 2004) that handles sparse classification. This approach uses a flat hierarchical classification, discarding any information contained in the hierarchical structure. Utilizing hierarchical information in SVM classification is a current topic of research and recently developed methods may improve performance by utilizing the ontology inherent structure.

Data Set

In order to make our experiments tractable, we have chosen one particular sub-area of the Cyc microtheory hierarchy—specifically, the sub-tree rooted under the CyclistsMt microtheory. The CyclistsMt microtheory contains information about Cyc itself and the people who work there, colloquially known as 'Cyclists'. It is reasonably well-populated with a good cross-section of available terms and predicates, making it a good target for our experiments. This microtheory contains 252 sub-microtheories, containing a total of 145,706 assertions. The average fan-out of the tree is 1.71 (in other words, the average microtheory in this tree has 1.71 microtheories directly under it in the tree). Microtheories which contain only a very small number of assertions were excluded from initial experiments, leaving approximately 30 microtheories into which over 32,000 assertions could be classified.

This work also excludes microtheories that contain a time dimension, i.e. temporal microtheories. Temporal microtheories are those that contain assertions that hold true only within a certain time constraint. For example, the assertion (isa RonaldReagan UnitedStatesPresident) is true only in the intersection of PeopleDataMt (the microtheory

[3] *http://svmlight.joachims.org/*

containing data about famous people) and the temporal microtheory:

```
(TimeIntervalInclusiveFn
  (HourFn 12 (DayFn 20 (MonthFn January
    (YearFn 1981))))
  (HourFn 11 (DayFn 20 (MonthFn January
    (YearFn 1989)))))
```

The time period from January 20, 1981 to January 20, 1989

Since temporal microtheories are very specific to individual assertions, it would be difficult to assign them using statistical learning algorithms; PeopleDataMt and the temporal microtheory shown contain very similar knowledge, and will likely require specialized knowledge to place information appropriately. We believe this is an appropriate initial simplification, as the majority of knowledge we wish to place into an ontology can probably be pre-filtered so that it contains only general, atemporal knowledge.

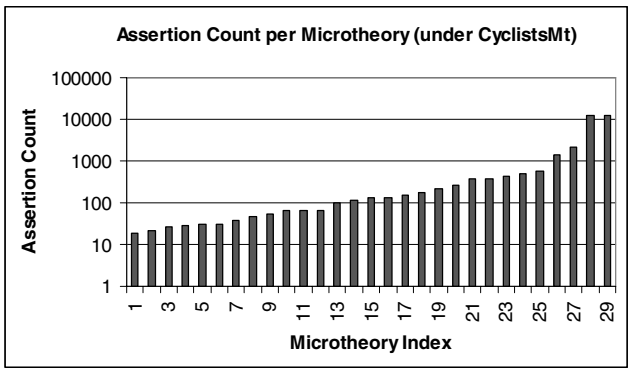

Figure 2: A logarithmic histogram of the distribution of assertions in atemporal microtheories under the CyclistsMt sub-tree used for this experiment.

The sub-tree rooted under CyclistsMt contains roughly 1% of the total number of assertions in the knowledge base and 1.07% of the microtheories. This subset is small enough to make our experiments feasible while still containing more data than a human could readily process. Furthermore, the distribution of assertions across these microtheories is roughly exponential, as is the distribution of knowledge in the KB as a whole.

As input to our classifiers we index all terms so that each assertion's formula is a list of term indices. Each unique atomic term, including strings and numbers, is transformed into a fixed numerical index. This ignores any structural information within an assertion (particularly noticeable in the case of the "not" term) and takes what amounts to a "bag of words" approach. Thus the formulae

```
(isa MattTaylor GraduateStudent)
(owns Dog MattTaylor 1)
```

will be represented as the vectors:

```
(25 570 400)
(40 823 570 429)
```

where all unmentioned terms are assumed to be absent from the assertions.

To evaluate our performance over the dataset, we divided the assertions randomly into 10 groups. For each group, the assertions in that group were used as a test set, and the remaining 9 groups of assertions were used as a training set. This standard statistical evaluation procedure, known as 10-fold cross-validation, works well for these experiments because the number of training data instances heavily outweighs the number of testing instances, as is the case in actual use of the microtheory placement suggestion tool. In each fold, the training set's assertions are labeled with the Mt in which they appear in the KB (i.e., the "correct" placement), and the test set's assertions are not labeled. The placement of the assertion by the SVM is then compared to the actual Mt in which it appears in the KB.

Results

Figure 3 shows the results of our experiments on the CyclistsMt sub-tree dataset. Precision (p), recall (r), and F_1 are standard evaluation metrics for classification tasks:

$$p = \frac{|\, correct \cap predicted \,|}{|\, predicted \,|}$$

$$r = \frac{|\, correct \cap predicted \,|}{|\, correct \,|}$$

$$F_1 = \frac{2 \times p \times r}{p + r}$$

Precision, recall, and F_1, defined in terms of the classification selected as compared to the actual Mt placement of an assertion. In F_1, precision and recall are weighted equally, as both are relevant in this domain.

Precision and recall are based on the number of assertions for which an Mt was predicted and the number of assertions that were predicted correctly. F_1 is the harmonic mean of recall and precision; in order to achieve a high F_1 score the classifier must achieve both high precision and high recall.

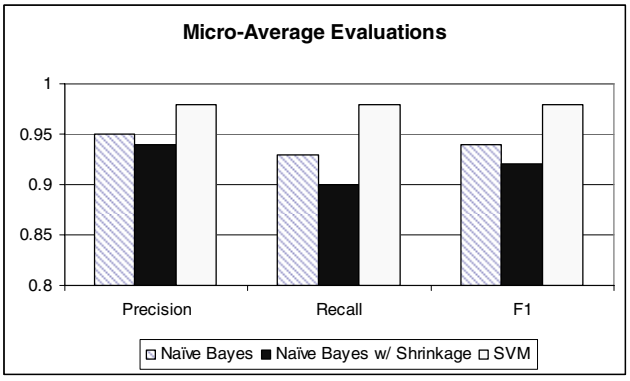

Figure 3: Precision, recall, and F_1 using Naïve Bayes, Naïve Bayes with shrinkage, and SVM classification.

The micro-average is not improved by using shrinkage, possibly because shrinkage allows less likely Mts (those with very few assertions) to have a relatively high weight.

Using shrinkage to take advantage of the microtheory hierarchy provides a significant benefit over using standard Naïve Bayes for the macro-average (see Figure 4); this is to be expected, given the extreme sparsity of the data. Each assertion contains only a few features and being able to use data from parent microtheories greatly improves the classifier's ability to generalize. This is of particular benefit in very specific microtheories that contain few assertions (i.e. positive training examples).

SVMs performed even better in our experiments. This is probably an indication of the same strengths that SVMs usually show over Naïve Bayes in text categorization experiments: SVMs can better deal with imbalance in positive and negative training examples. Because of the nature of the SVM algorithm, only the *support vectors* are used in determining class membership. Any positive or negative training examples that fall outside the *margin* created by the vectors are ignored. Naïve Bayes, on the other hand, includes *prior probability* in its calculations, which means the more negative training examples there are, the lower the score for the microtheory. Another strength is that Support Vector Machines do not make assumptions about feature independence within feature vectors. For example, the terms (YearFn 2006) and dateOfEvent are not statistically independent—when (YearFn 2006) occurs in an assertion, dateOfEvent is more likely to occur than it would otherwise be. Naïve Bayes makes the incorrect assumption that they are independent (which is precisely why it is called "naïve").

While SVMs greatly outperformed Naïve Bayes with shrinkage, the results for the Bayesian techniques confirm the hypothesis that data sparsity is a major difficulty in this problem, which can to some degree be overcome by using data found higher in the hierarchy. Therefore, it should be possible to improve the SVM performance in future experiments by finding ways to reduce data sparsity.

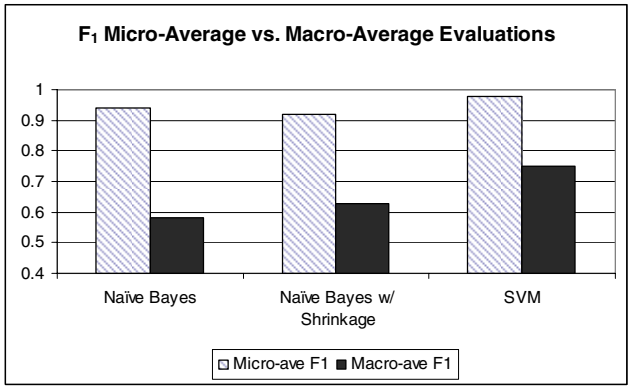

Figure 4: F_1 micro-average and macro-average scores for each of the three approaches taken. Comparative quality of micro- and macro-averages are consistent relative to one another across approaches.

Figure 4 shows the micro-average vs. macro-average F_1 score for each approach. Micro-average F_1 is computed by averaging over each assertion in the dataset. Macro-average F_1 is computed by averaging over each microtheory (category) in the dataset. Thus, micro-average scores are biased toward the classifier's performance on larger microtheories, while macro-average scores are relatively biased toward smaller microtheories. The relative performance of the different approaches is the same, regardless of the type of average being taken; while not surprising, this means that performance cannot be improved by simply using one algorithm for assigning assertions into large microtheories and another algorithm for small microtheories.

Every classifier tested had better performance on large microtheories than on small microtheories. This is to be expected, since there are more positive training examples for larger microtheories. It is also desirable, since new assertions are more likely to go in the larger microtheories.

Future Work

The next step in this research is to evaluate these methods on different datasets. By trying them on the whole Cyc microtheory hierarchy (rather than just a sub-tree), it will be possible to discover how well these approaches generalize. If they generalize well, they can be applied to any task that involves microtheory placement. It would also be advantageous to test them on datasets similar to those applications for which they will be most helpful. An important application of this method will be to classify facts gathered autonomously from the Internet or via ILP. Another important direction is to test more sophisticated classification techniques. The SVM formulation in these experiments was a flat classifier, but there may be some way to improve performance by using hierarchical methods, such as using a Pachinko machine approach.

We would also like to work to make the evaluation function more informative. The results above evaluate an assertion placement as either strictly correct or incorrect, depending on whether the microtheory chosen was where the test assertion was located in the KB. A more appropriate metric might give partial credit for placing an assertion just one level too high or too low in the hierarchy; such a "tree metric" would give a better indication of how far incorrectly labeled assertions were from the correct placement. It is also possible that some placements were acceptable, but were counted as incorrect because they disagreed with the microtheories originally assigned to them by the ontologists. Investigating this would involve having an ontologist examine the output of the classifiers by hand and deciding if the answers were "good enough." Another study would involve having multiple ontologists label data and comparing their inter-annotator agreement. Such a measure of the disagreement between trained ontologists would help define an upper bound on the performance of autonomous machine learning techniques.

Finally, in this work we assume that all assertions currently in the KB are in the correct Mt, and use that placement as training data. However, it is likely that some of the 4.6 million assertions are misplaced in the KB, as they were entered by many different people over a number of years. The classification techniques introduced in the paper, once sufficiently accurate, could be used to check the placement of existing assertions by sequentially excluding an assertion or group of assertions (i.e. "leave-one-out testing") and help produce a more consistent KB.

Conclusion

With precision and recall of 98%, we have shown that it is possible to use existing machine learning techniques to build a tool for automatically placing facts into reasoning contexts in an ontology. Specifically, SVMs overcame both the imbalance of positive and negative training examples and the extreme sparsity of the data to provide immediately usable results. We have argued why such a technique is necessary for automated ontology building via search and introspection. Finally, our experiments have indicated that careful use of the microtheory hierarchy has great potential to improve performance over that of a flat classifier, providing a clear path toward potential future improvements.

Acknowledgements

This research was supported by Cycorp, Inc.

References

Cabral, J., Kahlert, R.C., Matuszek, C., Witbrock, M., Summers, B. Converting Semantic Meta-Knowledge into Inductive Bias. *In Proceedings of the 15th International Conference on Inductive Logic Programming,* Bonn, Germany. 2005.

Crammer, K. and Singer, Y. On the Algorithmic Implementation of Multi-class SVMs, JMLR, 2001.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A-M., Shaked, T., Soderland, S., Weld, D.S., Yates, A. Web-Scale Information Extraction in KnowItAll (Preliminary Results). WWW 2004, pp. 100-110, 2004.

Fortuna, B., Mladenic, D., Grobelnik, M. Semi-automatic construction of topic ontology. In *Proceedings of the 16th European Conference on Machine Learning,* 2005.

Joachims, T. Text categorization with Support Vector Machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning,* pp. 137-142, 1998.

Joachims, T. Making Large-Scale SVM Learning Practical. Advances in Kernel Methods – Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola (ed.), MIT Press, 1999.

Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning,* 1997.

Lenat, D.B. The Dimensions of Context-Space. From http://www.cyc.com/doc/context-space.pdf, 1998.

Lenat, D.B. Cyc: A Large-Scale Investment in Knowledge Infrastructure, *CACM* 38, no. 11. 1995.

Liu, T., Yang, Y., Wan, H., Zhou, Q., Gao, B., Zeng, H., Chen, Z., Ma, W. 2005. An Experimental Study on Large-Scale Web Categorization, WWW 2005.

Masters, J. and Güngördü, Z. Structured Knowledge Source Integration: A Progress Report. In *Integration of Knowledge Intensive Multiagent Systems,* Cambridge, MA, 2003.

Matuszek, C., Witbrock, M., Kahlert, R.C., Cabral, J., Schneider, D., Shah, P., Lenat, D.B. Searching for Common Sense: Populating Cyc from the Web. In *Proceedings of the 20th National Conference on Artificial Intelligence,* Pittsburgh, PA. 2005.

Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J. An Introduction to the Syntax and Content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering,* Stanford, CA, 2006.

McCallum, A., Rosenfeld, R., Mitchell, T., Ng, A.Y. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proceedings of the 15th International Conference on Machine Learning,* pp. 359-367, 1998.

Mitchell, T. *Conditions for the equivalence of hierarchical and non-hierarchical Bayesian classifiers.* Technical report, CALD, CMU. 1998.

Schmolze, J., and Lipkis, T. Classification in the KL-ONE Knowledge Representation. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* pp. 330-332, 1983.

Shah, P., Schneider, D., Matuszek, C., Kahlert, R.C., Aldag, B., Baxter, D., Cabral, J., Witbrock, M., Curtis, J. Automated Population of Cyc: Extracting Information about Named-entities from the Web. In *Proceedings of the 19th International FLAIRS Conference,* pp. 153-158, Melbourne Beach, FL. 2006.

Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y. Support Vector Learning for Interdependent and Structured Output Spaces, in *Proceedings of the International Conference on Machine Learning,* 2004.

Vapnik, V.N. The Nature of Statistical Learning Theory. Springer, 1995.

Witbrock, M., Baxter, D., Curtis, J., et al. An Interactive Dialogue System for Knowledge Acquisition in Cyc. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence,* Acapulco, Mexico. 2003.