Using Ontologies and the Web to Learn Lexical Semantics

Aarti Gupta
Amazon.com Inc.
1200 12^{th} Avenue South
Seattle, WA 98144
aartig@amazon.com

Tim Oates
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, MD 21784
oates@cs.umbc.edu

Abstract

A variety of text processing tasks require or benefit from semantic resources such as ontologies and lexicons. Creating these resources manually is tedious, time consuming, and prone to error. We present a new algorithm for using the web to determine the correct concept in an existing ontology to lexicalize previously unknown words, such as might be discovered while processing texts. A detailed empirical comparison of our algorithm with two existing algorithms (Cilibrasi & Vitanyi 2004, Maedche et al. 2002) is described, leading to insights into the sources of the algorithms' strengths and weaknesses.

1 Introduction

Semantic resources, such as ontologies and lexicons, are widely used in Natural Language Processing (NLP). Automated text processing and understanding, word sense disambiguation, information extraction, speech recognition, and machine translation all require lexical resources. Ontologies are used in information retrieval for automatic query expansion, and the word sense disambiguation task can be made easier by domain-specific ontologies that reflect the distribution of word senses in a particular domain. Outside the realm of NLP, ontologies are widely used in the Semantic Web to allow computer programs to "understand" web content.

Acquiring knowledge by hand is extremely costly and time consuming. Construction of ontologies such as Word Net (Fellbaum 1998), Cyc (Lenat 1995), and EDR (Yokoi 1995) required enormous investments in both time and money. In addition, hand-crafted lexical resources are prone to human errors, biases, and inconsistencies in judgment.

We propose a novel method for learning lexical semantics. The ontological concept that best captures the lexical semantics of a new word is determined using statistical analysis of web pages containing the word. All experiments reported in this paper use the $OntoSem$ ontology (Nirenburg & Raskin 2004), a large ontology and knowledge representation system for language understanding tasks developed by the Institute for Language and Information Technologies (ILIT). We perform a detailed comparison of the proposed method with two existing methods, and in the process identify the individual components that most significantly impact performance on the task of statistics-based extension of symbolic knowledge for language processing tasks.

Lexical acquisition methods use different sources of data for learning. Historically, a static corpus has been used for training purposes (Widdows & Dorow 2002). Recently, the web has become a popular alternative. We compare our algorithm to that of Cilibrasi and Vitanyi (Cilibrasi & Vitanyi 2004), which uses the web to extract the meaning of words and phrases.

Our work is related to work on learning ontologies. Lin (Lin 1998) constructs an ontology from scratch by identifying similar words in a parsed corpus. Agirre et al. (Agirre et al. 2000) use the web to enrich existing concepts in ontologies. Each concept in WordNet (Fellbaum 1998) is associated with a topic signature (lists of topically related words). To identify the sense of a new word, they build a binary hierarchical clustering of all possible senses, compare the new word's context with the topic signatures for each branch and, similar to a decision tree, choose the highest ranking branch at each step. Maedche et al. (Maedche et al. 2002) determine the class of a new word by employing tree-descending and tree-ascending classification algorithms. Our algorithm determines which concept in an existing ontology is the best candidate for the meaning of a new word via tree descending, and is compared to the algorithm of Maedche et al.

2 Methods

This section describes the proposed approach for learning lexical semantics and summarizes two existing methods that are used for comparison.

2.1 Proposed Approach

The OntoSem ontology consists of concepts and taxonomic relations between concepts. To enrich existing concepts in the ontology with new words, we need to find in the hierarchy of concepts the concept that best approximates the meaning of a new word. This is accomplished using the notion of word similarity, for which we use Latent Semantic Analysis (LSA) (Deerwester et al. 1990).

To decide the most appropriate concept that a new word should belong to, we need to determine the similarity of the new word with words already lexicalized with the ontology.

To do this, we obtain a training set T of words from the ontology. For each word W, where W is in T or is the new word, document W.txt is created by tokenising the list of definitions retrieved after querying Google with "define:W". The resulting document collection is then passed as input to the LSA algorithm. Using the World Wide Web to dynamically build a document collection relevant to a given set of words allows ad hoc word learning.

Words are stopped using *Smart's* stop list, stemmed using *Porter's* stemming algorithm, and filtered based on the logarithm of their term frequencies. A word i with term frequency $freq_i$ is included in the term-document matrix only if $1 + \log(freq_i) \geq t$. Rather than represent raw term frequencies of the remaining content words in the term-document matrix, we use TFIDF.

We treat the OntoSem ontology similar to a decision tree. Starting at the root, we attempt to descend the ontology tree until we reach a leaf concept. At every tree node, the decision of which path to follow is made by choosing the child concept that is most similar to the new word. To compute similarity, for each child of the ontology node we currently are at, we randomly pick a representative set of words that belong to the subtree rooted at the child.

We then obtain definitions for the words in the representative sets and the word we are trying to learn. For every word, we create a new document that contains a list of definitions for the word. So, at the end, we have a document collection that contains a document for every word that we extracted from the various ontological sub-trees and a document for the new word. We then invoke the LSA algorithm which gives us word and document vectors in the reduced space. We compute cosine similarity scores for the word vectors. Finally, the 1-nearest neighbor classifier assigns the new word to the child concept that subsumes the word that was most similar to the new word. The process is repeated until the search reaches a leaf node.

2.2 Automatic Meaning Discovery using Google

This subsection describes a method proposed by Cilibrasi and Vitanyi (Cilibrasi & Vitanyi 2004) to automatically learn the meaning of words and phrases. Intuitively, the approach is as follows. Words that are semantically similar will co-occur more than words that are semantically unrelated. Conditional probabilities can be used as a measure of the co-occurrence of terms. $p(x|y)$ gives the probability of term x accompanying term y while $p(y|x)$ gives the probability of term y accompanying term x. These two probabilities are asymmetric. The semantic distance between terms x and y can be obtained by taking the negative logarithm of conditional probabilities $p(x|y)$ and $p(y|x)$, selecting the one that is maximum and normalizing it by the maximum of $\log 1/p(x)$, $\log 1/p(y)$.

$$D(x,y) = \frac{max\{\log \frac{1}{p(x|y)}, \log \frac{1}{p(y|x)}\}}{max\{\log \frac{1}{p(x)}, \log \frac{1}{p(y)}\}} \quad (1)$$

On the Web, probabilities of term co-occurrence can be expressed by Google page counts. The probability of term x is given by the number of web pages returned when Google is presented with search term x divided by the overall number of web pages M possibly returned by Google. The joint probability $p(x,y)$ is the number of web pages returned by Google, containing both the search term x and the search term y divided by the overall number of web pages returned by Google. The conditional probability $p(x|y)$ is defined as $p(x,y)/p(y)$.

We attempt to use the Google method to extend the OntoSem ontology. Once again, we use the tree descending algorithm to assign a new word to a concept. For each child of the ontology node we currently are at, we randomly pick a representative set of words. We then compute the distance between the new word and every word extracted from the ontological subtrees by submitting queries to Google via the Java URL interface and scraping the page counts from the web pages returned. The decision of which path to follow down the tree is made by choosing the child concept that subsumes a word which is closest to the new word by the distance measure.

2.3 Category Based Tree Descending Algorithm

Maedche et al. (Maedche et al. 2002) use a category-based method to determine the child concept that is most similar to a new word. For the ontology node that they are currently at, they gather all hyponyms at most 3 levels below. Then then build a generalized class vector for each child concept by adding up all the vector representations of hyponyms gathered that belong to the subtree rooted at the child and normalize the resulting vector to unit length. Similarity between the unit vector of a new word and the class vectors is measured by means of three different similarity metrics: Jaccard's coefficient, L1 distance, and skew divergence. The child concept whose class vector is most similar to the new word's vector is chosen as the next node in the path down the ontology tree.

The utility of using the category-based method is that besides reducing computational cost, it summarizes the characteristics of each class by combining multiple prevalent features together, even if these features are not simultaneously present in a single vector.

We use the Maedche method to extend the OntoSem ontology. At each ontology node, we gather all hyponyms at most three levels below, build a term-document matrix for the relevant document collection obtained from Google Define, obtain the normalized class vectors and use the L1 similarity metric to choose the child concept that is closest to the new word.

3 Experiments

This section describes the experiments performed in the process of identifying the components that work best for the statistics-based refinement of taxonomic relations. It also describes the experiments conducted to evaluate the performance of the proposed LSA-based tree descending method against the Google method and the category-based tree descending method.

Since document W.txt is created by obtaining context for word W, similarity between two words W_1 and W_2 can be measured by computing similarity between documents W_1.txt and W_2.txt. In all experiments, we compute similarity between document vectors and not word vectors be-

cause we empirically found that using document vectors gave marginally better results than word vectors.

For all experiments, we performed leave-one-out cross-validation, that is, one word was held back as the test word and the remaining words constituted the training set. Performance was measured in terms of the number of words correctly classified. A word is correctly classified if it is assigned to the correct child concept of an ontology node.

The value of t, that is, the threshold for filtering out low frequency words was fixed at t=1.6, so words that occurred fewer than two times in the corpus were filtered out.

3.1 Impact of corpus on the learning task

We wanted to gauge the impact of the corpus on the LSA-based tree descending algorithm. To do this, we randomly picked at most fifty words from each of the subtrees rooted at the 18 child concepts of the SOCIAL-ROLE node in the OntoSem ontology, giving a total of 269 words (after removing polysemous words).

We used the Wall Street Journal corpus to obtain contexts. For each word W, document W.txt was created by extracting a window of width seven words centered on each of the first 40 occurences of W in the corpus. We then replaced the WSJ document collection with the Google Define document collection, while keeping everything else the same. For each word W, document W.txt was created by submitting to Google the query "define:W".

The results of using the Google Define corpus vs. the WSJ corpus are shown in Figure 1. Accuracies for both corpora are calculated as the number of words correctly classified over the total number of words N that could possibly be classified correctly. For the WSJ corpus, N=202 and for the Google define corpus, N=254. The discrepancy between N and the number of words originally extracted from the ontology is due to the fact that not all words occur in the WSJ corpus or have definitions on the Web, or if they do, they occur fewer times than the threshold frequency.

It can be seen from Figure 5 that over all values of k, the number of singular values retained when running LSA, the system performs significantly better when fed with the Google Define corpus than when it is fed with the WSJ corpus. The Google Define corpus has a best accuracy of 64.9% for k=65 and k=85 while the WSJ corpus has a best accuracy of 22.2% at k=65.

Thus, the Google Define corpus dramatically improves the performance of the system because it contains fewer extraneous and irrelevant words that could potentially distract the LSA algorithm.

3.2 Impact of Similarity metric

Maedche et al. make use of three different similarity metrics: Jaccard's coefficient, L1 distance, and skew divergence in their category-based tree descending algorithm. We show that the performance of the category-based tree descending algorithm can be significantly improved by using the cosine similarity metric.

Initially, we implemented the category-based method using the L1 similarity metric because Maedche et al. empirically

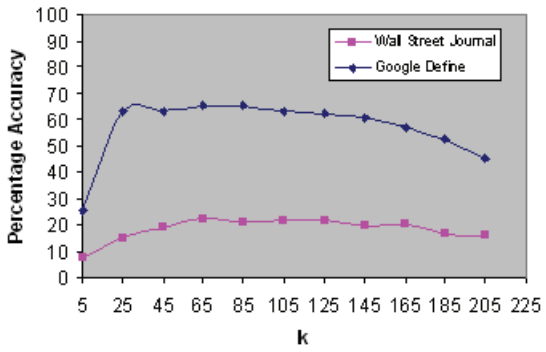

Figure 1: Performance of the LSA-based method using the WSJ corpus and Google Define corpus

Node	Words Classified Correctly
SOCIAL-ROLE	2/254
MAMMAL	1/347
ANIMATE	103/208
OBJECT	17/282

Table 1: Results obtained for the category-based method using the L1 metric

show that it gives better results than Jaccard's coefficient and skew divergence.

We carried out experiments at four nodes in the Ontology tree. For each node, we created a relevant document collection by picking all words at most three levels below the node and Googling for the words' definitions. We measured similarity between the normalized category-based vector representations obtained by summing individual document vectors. Table 1 shows the results obtained. Each cell in the table shows the number of words classified correctly out of the total number of words that could possibly be classified correctly.

Analysis of the results shows that in almost all cases, a new word was found to be similar to the child concept that in comparison to the other child concepts subsumed the fewest number of words. For example, at the SOCIAL-ROLE node, a vast majority of words were found to be similar to the CELEBRITY concept which had just one word under it, while at the MAMMAL node, most words were found similar to the concept FRESHWATER-MAMMAL which had just one word under it. This observation corroborates the findings of Maedche et al. in which they observed that most words were assigned to concepts that subsumed few words.

Given two words and their length n vectors, x and y, the L1 distance between them is given by:

$$Dist(x, y) = \sum_{i=0}^{n} |x_i - y_i|$$

From the definition, we can see that the distance between x and y will be small if the difference between x_i and y_i is small for all i. This means that, if x was a document vector for a

Node	Words Classified Correctly
SOCIAL-ROLE	177/254
MAMMAL	322/347
ANIMATE	186/208
OBJECT	222/282

Table 2: Results obtained for the category-based method using the cosine similarity metric

new word and y was a generalized class document vector, the distance between them would be small if two conditions were satisfied. First, the documents for x and y contained the same words and second, the documents for y as a whole contained few $other$ words. Child concepts that subsume few words in the ontology tree are represented by few documents which means fewer words in y as a whole and thus for such concepts, by default, the second condition is satisfied.

We replaced the L1 metric with the cosine similarity metric, and keeping everything else the same, carried out the experiments at the four ontology nodes again. Table 2 shows the results of using the cosine similarity metric. It can be seen that the cosine similarity metric does significantly better than the L1 metric at all four nodes.

3.3 Evaluation of the Three Methods

This section describes the experiments carried out to evaluate the performance of the LSA-based tree descending algorithm, the Google method and the category-based tree descending algorithm.

We picked a path in the ontology from the root to a leaf node and carried out experiments at every node on this path. Thus, we carried out experiments at the ALL, OBJECT, PHYSICAL-OBJECT, ANIMATE, ANIMAL, VERTEBRATE, MAMMAL, PRIMATE, HUMAN, and SOCIAL-ROLE nodes. The goal was to evaluate the performance of the three methods based on the number of words that were classified correctly at each node on the path. A word is classified correctly if it is assigned to the correct child concept.

LSA-based tree descending method

For every ontology node on the path, we randomly picked at most 50 words from each of the subtrees rooted at the child concepts of the node. So, at the ALL node, we would have words picked from the OBJECT, EVENT and PROPERTY subtrees, at the OBJECT node we would have words picked from the PHYSICAL-OBJECT, INTANGIBLE-OBJECT, MENTAL-OBJECT, SOCIAL-OBJECT and TEMPORAL-OBJECT subtrees. Thus, each node on the path had a different set of words on which the experiments were carried out. For each set of words, we used cross-validation testing to determine the number of words that were classified correctly.

To generalize the results over the random element involved in picking words, at each node, we carried out the experiments 10 times, each time picking a different set of words and performing cross-validation testing. We used values of k when running LSA that were empirically good.

Node	Baseline	LSA	Google	Category
ALL	33.33	53.72	49.72	87.07
OBJECT	20.00	59.74	48.48	78.72
PHYS-OBJ	50.00	74.64	68.54	93.64
ANIMATE	33.33	85.19	80.42	88.30
ANIMAL	50.00	83.30	74.60	93.30
VERTEBRATE	16.66	81.71	61.22	92.38
MAMMAL	12.50	78.65	58.00	94.96
PRIMATE	50.00	93.68	94.76	100.00
HUMAN	100.00	100.00	100.00	100.00
SOCIAL-ROLE	5.55	64.16	37.50	69.60

Table 3: Average accuracy (%) of the LSA-based tree descending method, the Google method and the category-based tree descending method

Google method

As with the LSA-based tree descending method, for every node on the path, we randomly picked at most 50 words from each of the subtrees rooted at the child concepts of the node. Thus, at each node we had a different set of words for which we measured the performance of the Google method. The experiments were carried out 10 times at each node, each time with a different set of words.

Category-based tree descending method

We carried out the experiments for the category-based tree descending method proposed by Maedche et al., except that we replaced the metrics used by them with the cosine similarity metric. For every node on the path, we gathered all words at most three levels below from each of the subtrees rooted at the child concepts of the node. Since there was no random element involved, we carried out the experiments just once at each node.

Results

To have a benchmark for the performance of the three methods, a baseline was calculated, which was the number of words classified correctly when a new word was randomly assigned to a child concept. Since the probability of getting a word correct by chance depends on the number of child concepts of a given node, the baseline varies with the number of child concepts of a node.

Table 3 shows the results for the three methods. Each cell shows the percentage accuracy for a given method, at a given node. For the LSA-based and Google methods, this value is the average of the accuracies obtained for each of the 10 runs of the experiment at a given node. Since Google's estimates of page counts can vary quite a bit for a given search term, the results for the Google method are as per the page counts returned at the time of performing the experiments.

From the table we can see that at all nodes, the category-based method outperforms the LSA-based and Google methods. The LSA-based method does better than the Google method at all nodes except at the PRIMATE node.

At every node, we performed a t-test on the accuracy distributions of the LSA-based and Google methods using the pub-

	y	$\overline{y}$
x	a	b
$\overline{x}$	c	d

Table 4: Contingency table for the co-occurrence of x and y

licly available Simple Interactive Statistical Analysis (SISA) package. The t-test results show that the difference in performance of the LSA-based and Google methods is statistically significant at all nodes other than at the ALL and PRIMATE nodes.

Similarly, the z-scores for the category-based method show that the difference in performance of the LSA-based and category-based methods is statistically significant at all nodes other than at the ANIMATE and ANIMAL nodes.

4 Analysis

The experimental results show that the category-based tree descending method outperforms the other two methods while the LSA-based tree descending method outperforms the Google method. In this section, we attempt to identify the reasons for the superior performance of the category-based method as well as the flaws in the Google method.

4.1 Analysis of the Google method

In this subsection, we argue that the Google method performs poorly because the NGD measure does not capture the true semantic distance between polysemous terms.

Given two words x and y, the Normalized Google Distance (NGD) between them is given by:

$$NGD(x,y) = \frac{max\{\log \frac{1}{p(x|y)}, \log \frac{1}{p(y|x}\}}{max\{\log \frac{1}{p(x)}, \log \frac{1}{p(y)}\}} \quad (1)$$

Words x and y will be perceived to be semantically similar if the distance between them is small. That is, the numerator of (1) should be small while the denominator should be large. Minimizing the numerator means maximizing both $p(x|y)$ and $p(y|x)$ while maximizing the denominator means minimizing either $p(x)$ or $p(y)$.

Table 4 shows a contingency table for the co-occurrence of terms x and y. From the table:

$$p(x|y) = \frac{p(x \cap y)}{p(y)} = \frac{a}{a+c}$$

$$p(y|x) = \frac{p(x \cap y)}{p(x)} = \frac{a}{a+b}$$

In order to maximize $p(x|y)$ and $p(y|x)$, c and b respectively should be minimized. Also, from the table,

$$p(x) = \frac{a+b}{a+b+c+d}$$

$$p(y) = \frac{a+c}{a+b+c+d}$$

To maximize the denominator, we minimize either $p(x)$ or $p(y)$. To minimize $p(x)$, we need to maximize c and d. However, maximizing c is not possible because c needs to be minimized to minimize the numerator. To minimize $p(y)$, we need to maximize b and d. Maximizing b is not possible because b should be small to minimize the numerator. Thus, in either case, d needs to be maximized.

It is clear that the Google method will say that x and y are similar if three conditions are satisfied. First, x and y co-occur in a certain number of web pages (cell 1), second, the number of web pages in which either of x or y occurred without the other is minimal (cells 2 and 3) and, finally, there are infinitely many web pages in which neither x nor y occurred (cell 4).

The Web, by virtue of its unrestricted size satisfies the last condition. However, at the same time, the size of the Web can be a problem too. Due to the sheer mass of web pages on almost every conceivable topic, it is likely that a word on the Web will be used in more than one sense. This means that, even if two words x and y are semantically related, there are likely to be a significant number of web pages in which x is used in a sense not related to y and vice-versa. Thus, for almost all pairs of words, the second condition would not be satisfied. For example, the words $monk$ and $pope$ are both RELIGIOUS-ROLES that a HUMAN plays. Since the two words are semantically related, they co-occur in a significant number of pages. A Google query with the words "monk and pope" retrieves 569,000 web pages. However, the query "monk $and\ not$ pope" retrieves 5,740,000 web pages while the query "pope $and\ not$ monk" retrieves 17,600,000 pages.

4.2 Analysis of the Category-based method

Rather surprisingly, the category-based method, which computes word similarities in the original vector space performs better than the more sophisticated LSA-based method. To identify the reasons for the category-based method performing better than the LSA-based method, we carried out a set of controlled experiments.

We carried out experiments with the LSA-based method and the category-based method at the PHYSICAL-OBJECT, ANIMATE, ANIMAL, MAMMAL and SOCIAL-ROLE nodes. The category-based method was modified slightly in that, at a given node, at most 50 words were randomly picked from each of the subtrees rooted at the child concepts. This modification was made so that, other than the use of LSA in the first method and category-based vector representations in the second method, the two methods were identical in all respects.

For the experiments at each node, two control groups were used. The first "control group" is a method that computes word similarity using document vector representations in the original vector space, that is, it neither uses LSA nor category-based vector representations. The second "control group" is a method that computes word similarity using category-based vector representations in the reduced vector space, that is, it uses both LSA and category-based vector representations.

At each node, we evaluated the performance of the four methods 10 times each, using a different set of words for each evaluation.

Tables 6 through 10 show the results obtained at each node. For each table, the first cell shows the average percentage accuracy for the method that uses both LSA and category-

	$Category$	$\neg Category$
LSA	60.94	74.64
$\neg LSA$	79.86	80.49

Table 5: Results obtained at the PHYSICAL-OBJECT node, k= 15

	$Category$	$\neg Category$
LSA	84.44	85.19
$\neg LSA$	89.95	85.88

Table 6: Results obtained at the ANIMATE node, k= 20

based vector representations, the second cell shows the average accuracy for the LSA-based method, the third cell shows the average accuracy for the category-based method and the fourth cell shows the average accuracy for the non-LSA, non category-based method. The values of k at which the experiments were carried out were the "good" values which were empirically chosen.

By summing the accuracies in the tables row-wise, we can see that, at each node, the non-LSA method does better than the LSA method. Summing the accuracies column-wise. we see that, at the PHYSICAL-OBJECT, MAMMAL and SOCIAL-ROLE nodes, the non-category method seems to be better, while at the ANIMATE and ANIMAL nodes, the use of category-based vectors seems to give better performance. Looking at individual cells, we see that the non-LSA, category-based method does best at all nodes except at the PHYSICAL-OBJECT node. In fact, even the simplistic non-LSA, non category-based method does better than the LSA-based method at all nodes except at the SOCIAL-ROLE node.

5 Conclusions

We presented a novel method that used both symbolic knowledge and statistical techniques to learn lexical semantics. We found that using good context dramatically improved the performance of the system. We then used the Google method and the category-based tree descending method to perform the task of ontology learning. We discovered that the performance of the category-based tree descending method could be significantly improved using the cosine similarity metric. The modified category-based tree descending method was found to be the best approach for extending an existing ontology with new words. We showed that the Google method performed poorly because the NGD measure did not capture the true semantic distance between polysemous terms. We also showed that our proposed method performed poorly because of the LSA algorithm.

	$Category$	$\neg Category$
LSA	86.70	83.30
$\neg LSA$	87.59	86.55

Table 7: Results obtained at the ANIMAL node, k= 20

	$Category$	$\neg Category$
LSA	67.08	78.65
$\neg LSA$	87.06	80.10

Table 8: Results obtained at the MAMMAL node, k= 30

	$Category$	$\neg Category$
LSA	53.93	66.14
$\neg LSA$	69.60	64.17

Table 9: Results obtained at the SOCIAL-ROLE node, k= 75

6 Acknowledgments

This research was supported in part by NSF CAREER award 0447435. We would like to thank the anonymous reviewers for their many suggestions on ways to improve the paper.

References

[1] E. H. Eneko Agirre, Olaz Ansa and D. Martinez. Enriching very large ontologies using the www. In *EJCAI 2000 WOrkshop on Ontology Learning*.

[2] R. Cilibrasi and P. Vitanyi. Automatic meaning discovery using google. *preprint, arxiv.org/abs/cs.CL/0412098*, 2004.

[3] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[4] C. D. Fellbaum. *WordNet: an electronic lexical database.* The MIT press, Cambridge, MA, USA, 1998.

[5] D. B. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, 1995.

[6] D. Lin. Automatic retrieval and clustering of similar words. In *17th International Conference on Computational Linguistics and of the 36th Annual Meeting of the Association for Computational Linguistics*.

[7] A. Maedche, V. Pekar, and S. Staab. *Ontology Learning Part One - On Discovering Taxonomic Relations from the Web*, pages 301–322. Springer Verlag, 2002.

[8] S. Nirneburg and V. Raskin. *Ontological Semantics.* The MIT press, Cambridge, MA, USA, 1998.

[9] D. Widdows and B. Dorow. A graph model for unsupervised lexical acquisition. In *COLING*, 2002.

[10] T. Yokoi. The edr electronic dictionary. *Commun. ACM*, 38(11):42–44, 1995.