UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

REDUCING POLYSEMY IN WORDNET

Kanjana Jiamjitvanich and Mikalai Yatskevich

December 2008

Technical Report # DISI-08-085

Reducing polysemy in WordNet

Kanjana Jiamjitvanich, Mikalai Yatskevich

Department of Information and Communication Technology
University of Trento, Italy

Abstract. A known problem of WordNet is that it is too fine-grained in its sense definitions. For instance, it does not distinguish between homographs and polysemes. This distinction is crucial in many natural language processing tasks. In this paper we propose to distinguish only between homographs within WordNet data while merging all polysemous senses. The ultimate goal of this exercise is to compute a more coarse-grained version of linguistic database. In order to achieve this task we propose to merge all polysemous senses according to similarity scores computed by a hybrid algorithm. The key idea of the algorithm is to combine the similarity scores produced by diverse semantic similarity algorithms. We implemented the algorithm and evaluated it on the dataset extracted from the WordNet. The evaluation results are promising in comparison to the other state of the art approaches.

1 Introduction

WordNet [14] is an electronic lexical database for English which stores lemmas and exceptional forms of words, word senses, sense glosses, semantic and syntactic relations between words, and other information related to the structure and use of the language. WordNet data is used by Controlled Vocabulary (CV)[**?**] component of Sweb [**?**] knowledge management system. CV stores background knowledge to be used by the other components of the system. In particular, CV is used in a number of knowledge management tasks such as semantic search[5], classification [9] and semantic matching [8]. It serves a key role in natural language (NL) to formal language (FL) conversion process (see [6] for more details). At the moment CV contains mostly WordNet data. A known problem of WordNet is that it is too fine-grained in its sense definitions. For instance, it does not distinguish between homographs and polysemes. This distinction is crucial in many natural language processing tasks. In particular, it heavily influences the performance of state of the art Word Sense Disambiguation (WSD) algorithms. They tend to produce significantly higher accuracy scores while exploiting more coarse-grained sense definitions [15].

In this paper we propose to distinguish only between polysemes within WordNet data while merging all homograph synsets. The ultimate goal of this exercise is to compute a more coarse-grained version of linguistic database. In order to achieve this task we propose to merge all polysemous senses according to similarity scores computed by a hybrid algorithm called meta matcher. The key

idea of meta matcher is to combine the similarity scores produced by diverse element level matchers that implement various semantic similarity algorithms. The combination parameters may be learned from training data if available. We implemented meta matcher and three element level matchers: WordNet relation, part of speech context and the novel inverted sense index inexact matcher and evaluated them on the dataset extracted from WordNet. The evaluation results are promising in comparison to the other state of the art approaches. In particular, in terms of f-measure our method outperforms the other approaches.

This paper is structured as follows. In Section 2 we review the structure of WordNet and explain why polysemy reduction is an important step required by end-user applications. Section 3 is devoted to meta matcher we use in our algorithm. Section 4 describes in detail element level matchers we use while Section 5 presents evaluation results. We discuss related work in Section 6 while Section 7 concludes the paper.

2 Polysemy in WordNet

WordNet [14] is the lexical database for English language. A synset is a WordNet structure for storing senses of the words. A synset contains a set of synonym words and their brief description called gloss. For example, *well*, *wellspring* and *fountainhead* have the same meaning according to WordNet, so these three words are grouped in to one synset which is explained by a gloss "*an abundant source*". Words are connected by lexical relations, while synsets are connected by semantic relations. Different relations are defined for various parts of speech. In this paper we exploit *hypernymy* relation defined for noun and verb synsets; *antonymy* and *synonymy* relations for adjectives synsets, and *derivationally related* lexical relation. Hypernym is the generic term used to explain a whole class of specific instances. For example, *animal* is a hypernym of *cat*. Antonymy relation connects the synsets with the opposite meaning. For example, *active* is an antonym of *uninvolved*. Synonymy relation connects synsets with similar meaning. For example, *active* is synonym of *involved*. Derivationally related connects words in different syntactic categories that have the same root form and are semantically related. For example, *activity* is derivationally related to *active*.

During WordNet development synsets are organized into forty-five lexicographer files based on syntactic category and logical groupings. For example, lexicographical file names for nouns include act, animal, event, etc. while lexicographical file names for verbs include change, creation, emotion, etc.

Starting from WordNet 3.0 word forms from the definitions (glosses) in WordNet's synsets are manually linked to the context-appropriate sense in WordNet. Thus, the glosses are a sense-disambiguated corpus and WordNet 3.0 is the dictionary against which the corpus was annotated.

A known problem of WordNet is that it is too fine-grained in its sense definitions. For instance, it does not distinguish between homographs (words that have the same spelling and different meanings) and polysemes (words that have related meanings). This distinction is crucial in many natural language process-

ing tasks. In particular it heavily influences the performance of state of the art Word Sense Disambiguation (WSD) algorithms [15]. In particular, state of the art WSD algorithms tend to produce significantly higher accuracy scores while exploiting more coarse-grained sense definitions. On the other hand WSD is a crucial step in semantic matching [8] and semantic search[5] which are the key knowledge management operations of Sweb system [**?**].

3 A Meta matcher

We propose to distinguish only between polysemes within WordNet data while merging all homograph synsets. The ultimate goal of this exercise is to compute a more coarse-grained version of linguistic database. In order to achieve this task we propose to merge all polysemous senses according to similarity scores computed by a hybrid algorithm.

The key idea of our method is to combine the results obtained by application of various element level matchers (see [7] for extensive discussion) in hybrid way by a meta matcher. Meta matcher takes as an input two WordNet senses and produces *true* if these senses have to be merged. It first calculates the similarity scores for input senses exploiting a set of element level matchers. If the score of any element level matcher exceeds threshold pre-learned on the training dataset *true* is returned.

Learning algorithm obtains an optimal threshold combination by exhaustive search through all threshold combinations of all element level matchers. The pseudo code of learning algorithm is presented in Algorithm 1.

Algorithm 1 Optimal threshold learning algorithm

```
1: void optimizeThresholds (Set mappings, Set goldenStandard, Vector thresholds,
   int matcherIndex)
2: if (matcherIndex<getNumberOfMatchers()) then
3:    Set admissibleThresholds=getAdmissibleThresholds();
4:    for each (threshold in admissibleThresholds) do
5:       thresholds=updateThresholds(thresholds, threshold);
6:       Set matcherMappings=getMappings(matcherIndex, threshold);
7:       mappings=Union(mappings, matchersMappings);
8:       optmizeThresholds(mappings, goldenStandard, thresholds,
         matcherIndex+1);
9:    end for
10: else
11:    double F-Measure=computeFMeasure(mappings, goldenStandard);
12:    if (isMaximumSoFar(F-Measure)) then
13:       storeBestThresholdCombination(thresholds);
14:    end if
15: end if
```

optimizeThresholds takes in input an initially empty set of mappings, a golden standard produced by human annotators for learning dataset, an initially empty vector of optimal thresholds and an index of element level matcher to be executed. If the index corresponds to an element level matcher (line 2) for any of admissible thresholds (line 4), a set of mappings is returned by a matcher for a given threshold (line 6). Those mappings are intersected with mappings produced by previously executed matchers (line 7) and the next element level matcher is executed (line 8). If results from all matchers have been collected f-measure is computed (line 11) and compared with maximum out of f-measures computed so far (line 12). If the new f-measure value exceeds those computed before (for the other threshold combinations) the current threshold combination is memorized as the best so far (line 13). The latest threshold combination saved by **storeBestThresholdCombination** corresponds to the maximum f-measure obtained from the results of element level matchers on the training dataset. Notice that **optimizeThresholds** utilizes the simplest aggregation strategy (introduced in [8]) by taking a union of element level matcher results as a final result.

4 Element level matchers

Element level matchers exploited by meta matcher take two WordNet senses in input and produce a numerical score [0,1] which presents quantitative measure of similarity of two input senses.

4.1 WordNet relation matcher

WordNet relation matcher exploit WordNet structure to compute the similarity of two input WordNet senses. In particular, it starts by obtaining two sets A and B of senses connected to input senses by a given relation. Then the two sets are compared as follows:

$$SimilarityScore = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1}$$

SimilarityScore which is a number between 0 and 1 is returned as a result. We have implemented WordNet relation matcher for the following WordNet relations: *hypernym*, *derivationally related form*, *antonym* and *synonym*. The latter two relations where considered for adjectives only.

4.2 Part of speech context matcher

Part of speech context (POSC) matcher exploits part of speech (POS) and sense tagged corpora for similarity computation. In particular, for each WordNet sense occurrence within corpora a set POS tags in the immediate vicinity (or POS context) of sense is memorized. Given multiple occurrence a sense within corpora each sense is associated with a set of POS contexts. Then, the similarity between

two senses is computed as set similarity between sets of POS contexts associated with them. For set similarity computation we exploit Eq. 1.

For example, let us compare:

- ***head#4***, *chief#1*, *top_dog#1 - person who is in charge* and
- ***head#13***, *principal#2*, *school_principal#1*, *head_teacher#1 - the educator who has executive authority for a school*

Taking sense and POS tagged WordNet 3.0 glosses as corpora we can find the following occurrences of *head#4*, *head#13* and their co-lemmas:

- the job/NN of/IN a head/NN/head#4 of/IN a government department/NN;
- the position/NN of/IN head/NN/head#4;
- the post/NN of/IN principal/NN/principal#2;

So given a window of two content words before and two content words after sense occurrence we have "NN IN NN IN NN" and "NN IN NN" POS contexts associated with *head#4* and "NN IN NN" POS context associated with *head#13*. Eq. 1 gives 0.5 similarity for these two sets. So according to POSC matcher *head#4* is similar to *head#13* with similarity coefficient 0.5.

4.3 Inverted sense index inexact matcher

Inverted sense index inexact (ISII) matcher exploits sense tagged WordNet 3.0 glosses for similarity computation. In particular, for each WordNet sense occurrence within sense tagged glosses, the synset of a tagged gloss is memorized. The process resembles inverted index construction with key difference of building index on WordNet senses rather than words and memorizing synsets rather than documents. As soon as an index is constructed two given senses can be compared by comparing sets of synsets associated with them. We use for set comparison a slight modification of Eq. 1:

$$SimilarityScore = \frac{|InexactIntersect(A,B)|}{|A| + |B| - |InexactIntersect(A,B)|} \tag{2}$$

The key difference is *InexactIntersect* operator that returns in not only shared elements of two sets but also the elements of two sets that are similar (according to predefined similarity measure) but not equivalent. In particular, since elements of A and B sets in our case are synsets we used Resnik similarity measure [19].

The measure is based on the concept of *information content* [19]. Information content of a concept is calculated as follows. Firstly, the number of concept occurrences (F_C) within text corpus is calculated. Then, the number of all subsuming concept occurrences within a corpus is calculated and added to F_C. Thus, the root concept will count for the occurrences of all concepts in the taxonomy. In case of WordNet synsets counts are precomputed for wide range of corpora. In our experiments we used British National Corpus (BNC) [3] as the biggest corpora available. Information content of a concept is defined as follows:

$$IC(c) = -\ln(\frac{F_C}{F_{Root}}) \tag{3}$$

where F_C and F_Root are, respectively, counts of concept C and root concept of the taxonomy. Note that the fraction represents the probability of occurrence of a concept in the large corpus.

Resnik defines semantic similarity of two concepts as the amount of information they share in common what is equal of information content of their lowest common subsumer, that is the lowest node in the taxonomy that subsumes both concepts. For example, the lowest common subsumer of *dog* and *cat* is *carnivore*. Therefore, the Resnik semantic similarity measure is defined as:

$$Similarity_{Resnik}(C_1, C_2) = IC(lcs(C_1, C_2)) \tag{4}$$

where IC is an information content of a concept and $lcs(C_1, C_2)$ is a lowest common subsumer of concepts C_1 and C_2.

The pseudo code implementing *InexactIntersect* operator is presented in Algorithm 2.

Algorithm 2 An algorithm implementing InexactIntersect operator

```
 1: Set inexactIntesect (Set source, Set target, double threshold)
 2: Set intersection;
 3: for each (sourceSynset in source) do
 4:    for each (targetSynset in target) do
 5:       double similarity=getResnikSimilarity(source, target);
 6:       if (similarity > threshold) then
 7:          if (sizeOf(source) > sizeOf(target)) then
 8:             addToSet(intersection, target);
 9:          else
10:             addToSet(intersection, source);
11:          end if
12:       end if
13:    end for
14: end for
15: return intersection
```

inexactIntersect function takes in input two sets of synsets and a threshold for internal similarity measure. It returns a set of synsets which is an inexact intersection of two input sets. For each pair of synsets in two input sets the Resnik similarity is computed (line 5). If the similarity exceeds a given threshold (line 6) the synset is memorized in the intersection. Note that line 7 ensure that we always save into intersection set the synsets from the smallest of two input sets. This allows us rule out double counting when computing cardinality of intersection set and ensure that intersection set is always smaller or equal to each of input sets.

Let us illustrate the use of ISII matcher on example presented in Section 4.2, namely comparison of (***head#4***, *chief#1*, *top_dog#1*) and (***head#13***, *principal#2*, *school_principal#1*, *head_teacher#1*) senses. The senses are used as annotations in the following glosses:

– minister#4 - the job of a head /head#4 of a government department;
– headship#2 - the position of head /head#4;
– principalship#1 - the post of principal /principal#2;

Thus, the sets of synsets associated with *head#4* and *head#13* are {*minister#4*, *headship#2*} and {*principalship#1*} respectively. Resnik matcher returns similarity score of 7.53 for *headship#2* and *principalship#1* pair and the score of 5.91 for *minister#4* and *principalship#1* pair. Given threshold of 4 we used in our experiments from Eq. 2 we obtain 0.5 similarity for *head#4* and *head#13* senses.

5 Evaluation results

For evaluation of matching quality we used Precision, Recall and F-Measure. Precision is the measure of soundness. It is calculated as:

$$Precision(P) = \frac{|A \cap B|}{|B|}$$

where A is a set of correct pairwise correspondences among the senses that should be merged and B is a set of pairwise correspondences as returned by the system.

Recall is the measure of completeness. It is calculated as:

$$Recall(R) = \frac{|A \cap B|}{|A|}$$

F-measure combines both Precision and Recall into single quality measure. It is calculated as:

$$F1 = \frac{2 * P * R}{P + R}$$

We used a dataset exploited in SemEval[1] coarse-grained English all words task. The dataset is validated by human lexicographers. The dataset contains 1108 nouns, 591 verbs, 262 adjectives and 208 adverbs. We split it into two equal parts namely training and testing datasets. The training dataset further was used for obtaining an optimal threshold combination for meta matcher while evaluation was performed on testing dataset.

First, we compared the results obtained from our meta matcher with simple baseline algorithm which merges all the senses that occur in the same lexicographical file in WordNet. Note that the measure is defined for nouns and verbs only since adjectives and adverbs lack meaningful separation into different lexicographical files. The results are presented on Figure 1.

As from the figure our algorithm significantly outperforms baseline in the case of nouns and draws in case of verbs. Relatively weak results in case of adjectives and adverbs can be explained by data sparseness. In particular, there are relatively few adjective and adverb annotations in the corpora POSC and ISII matchers exploit.

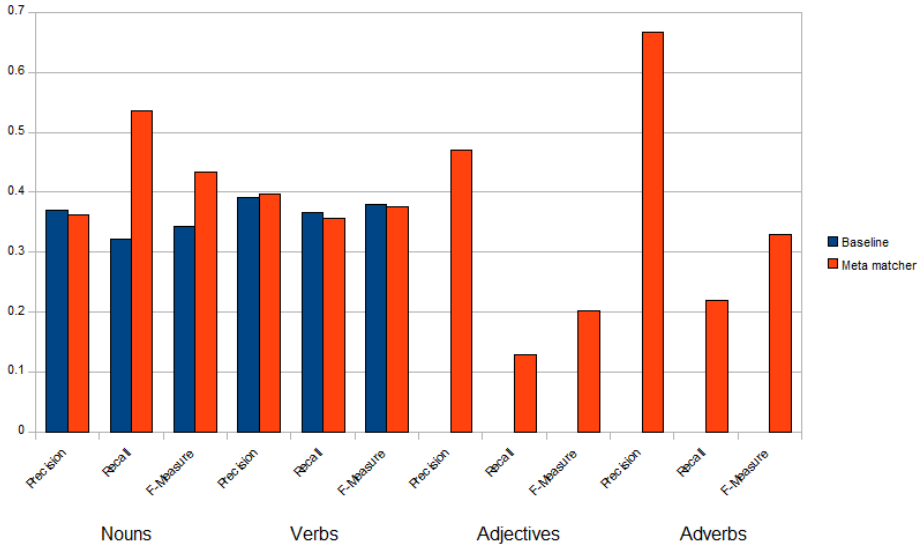

Fig. 1. Meta matcher vs. baseline matcher results

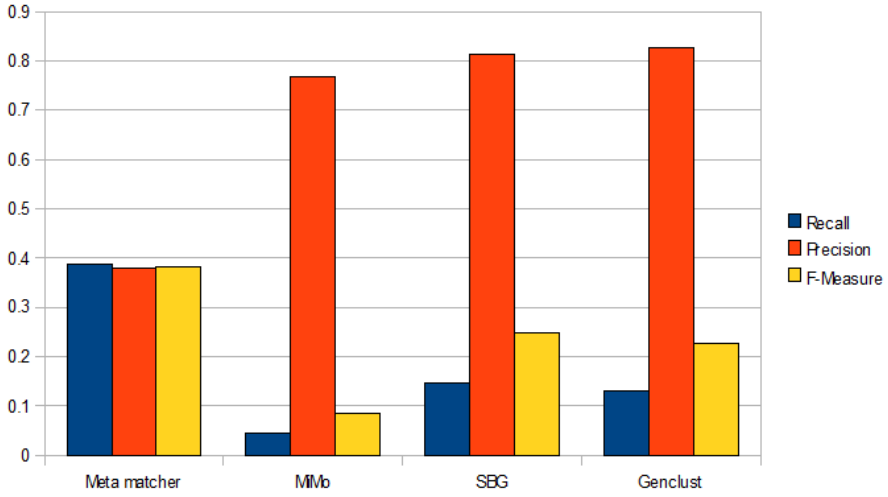

Fig. 2. Comparative evaluation results

We compared results of meta matcher with 3 other sense merging methods. In particular, we re-implemented sense merging algorithm [12], Genclust algorithm [18] and MiMo algorithm [13]. The results are presented on Figure 2.

Meta matcher significantly outperforms the other methods in terms of F-Measure. Application of meta matcher to the whole WordNet database allowed to reduce the number of synsets by 47.08%.

6 Related work

The problems of reducing polysemy in WordNet have received significant attention in the recent years. Many approaches to this problem relied on structural properties of WordNet for coarse-grained sense computation. In particular, domain extension to WordNet have been developed in [10]. In [17] two senses are to be merged based on Twin feature namely the case when two senses share more than one synonym word. In [18] generalization cluster (GenClust) algorithm was proposed. The algorithm merges senses based on coordinate term (cousin) relation as defined in WordNet. MiMo algorithm[13] applies a set of semantic rules for merging senses. Some of the rules exploit WordNet relations such as hyponym, pertainym, antonym and verb group. The other exploit structural properties like Twin and MaxMN (minimum and maximum distance from least common subsumer) features.

Semantic based grouping algorithm have been proposed in [12]. The algorithm exploits semantic relations of WordNet. In particular, two senses are considered to be merged if they are connected by a semantic semantic relation to the similar sets of senses. Semantic relations considered are synonym, hypernym, hyponym, coordinate term, and domain of synset. The application specific evaluation has been performed on two datasets of web search engines queries. Both datasets contain 10 queries. The reported precision of the algorithm is 100% and while recall is 66%.

The method presented in [15] is based on the exploiting mappings between WordNet and Oxford English dictionary (ODE). Both lexical and semantic methods are used to map WordNet's senses to more coarse-grained senses of ODE. In particular, well-known lexical overlap method [11] is used along with Structural Semantic Interconnections introduced in [16]. The method has 33.54 % of sense reduction, and the average degree of polysemy decreased from 6.65 to 3.32. The evaluation result shows that a coarse-grained inventory can improve performance of WSD. For example F-measure of Gambl [4] WSD system is improved from 65% to 70.84 %.

A supervised learning solution for merging WordNet senses based on support vector machines (SVM) was proposed in [20]. The features used for the system training included: WordNet relatedness measures [2], semantic and syntactic relations, cosine similarity of topic signatures [1], and the mapping to (more coarse-grained) ODE. F-Measure of SVM for nouns is 42.8% and for verbs is 43.19%.

[1] http://lcl.di.uniroma1.it/coarse-grained-aw/index.html

7 Conclusion and future work

We have presented an algorithm allowing to significantly reduce polysemy in WordNet while preserving the high quality of merging process as illustrated by high f-measure score. The future work includes (i) performing application specific and end-user oriented evaluation, i.e., measuring improvement from coarse-grained linguistic database in semantic search, classification and semantic matching tasks; (ii) and development of more effective combination strategies within meta matcher.

References

1. E. Agirre and O. Lopez de Lacalle. Publicly available topic signatures for all wordnet nominal senses. In *4rd International Conference on Language Resources and Evaluations (LREC)*, 2004.
2. A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47, March 2006.
3. G. Burnage and D. Dunlop. Encoding the british national corpus. In *English Language Corpora: Design, Analysis and Exploitation, Papers from the 13th international conference on English Language research on computerized corpora*, 1992.
4. B. Decadt, V. Hoste, W. Daelemans, and A. Van den Bosch. Gambl, genetic algorithm optimization of memory-based WSD. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
5. F. Giunchiglia, U. Kharkevich, and I. Zaihrayeu. Concept search. In *The 7th International Semantic Web Conference 2008 (ISWC2008)*, 2008.
6. F. Giunchiglia, M. Marchese, and I. Zaihrayeu. Encoding classifications into lightweight ontologies. *Journal of Data Semantics*, 8:57–81, 2007.
7. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *The Semantic Web: ISWC 2004: Third International Semantic Web Conference: Proceedings*, 2004.
8. F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX:1–38, 2007.
9. F. Giunchiglia, I. Zaihrayeu, and F. Farazi. Converting classifications into owl ontologies. In *3rd Asian Semantic Web Conference (ASWC2008)*, 2008.
10. A. Gliozzo, C. Strapparava, and I. Dagan. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, 18(3):275–299, 2004.
11. M. Lesk. Automatic word sense disambiguation: How to tell a pine cone from an ice cream cone. In *SIGDOC Conference*, 1986.
12. W. Meng, R. Hemayati, and C. Yu. Semantic-based grouping of search engine results using wordnet. In *Joint Conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management (APWeb/WAIM'07)*, 2007.
13. R. Mihalcea and D. Moldovan. Automatic generation of a coarse-grained wordnet. In *NAACL Workshop on WordNet*, 2001.
14. G. Miller. *WordNet: An electronic Lexical Database*. MIT Press, 1998.
15. R. Navigli. Meaningful clustering of senses helps boost word sense disambiguation performance. In *COLING-ACL 2006*, 2006.

16. R. Navigli and P. Velardi. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(7):1063–1074, 2004.
17. W. Peters and I. Peters. Automatic sense clustering in eurowordnet. In *Proceedings of LREC'2000*, 2000.
18. W. Peters, I. Peters, and P. Vossen. Automatic sense clustering in eurowordnet. In *Proceedings of LREC'1998*, 1998.
19. P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.
20. R. Snow, S. Prakash, D. Jurafsky, and A. Ng. Learning to merge word senses. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.