Discovering Additional Ontological Categories for Words*

Michael Gabilondo
mgabilo@gmail.com

Fernando Gomez
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, USA
gomez@cs.ucf.edu

January 2010 (Revised February 2018)

Abstract

The unsupervised algorithm proposed in this paper discovers additional ontological categories for words which are underrepresented in the WordNet ontology. The basic approach begins with extracting local syntactic dependencies for each word by parsing a large corpus to construct feature vectors for those words. The words' feature vectors are used to construct feature vectors for upper-level ontological concepts by bootstrapping from the existing WordNet ontology. We use a similarity metric between these two types of feature vectors to discover ontological categories for each word. The algorithm discovered a total of 7420 additional ontological categories for 12721 words. Three human judges evaluated the correctness of a random subset of 200 out of the 7420 categorizations and found an average precision of 75.5% and average inter annotator agreement of 72.5%.

1 Introduction

The organization of the upper-level ontology of language is crucial for semantic interpretation. This is because the meaning of a verb is dependent in part on the ontological categories of its arguments' head nouns. For example, the verb "break" in the sentence "She broke the table" means to damage a physical object. This sense of the verb is chosen because the sense of "table" which was chosen is a physical object. Shifting the ontological category of the head noun to something that is not a physical object, as in "She broke the silence," also changes the meaning of the verb. If a sense of a word is

*(c) 2018 Michael Gabilondo, Fernando Gomez. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. https://creativecommons.org/licenses/by-nc-nd/4.0/

not correctly categorized in the ontology, interpretation of certain sentences will not be possible. For example, if someone does not know that “tavern” is a business and only knows that it is a building, the sentence “The tavern hired a new bartender” would not make sense because buildings are artifacts, not human agents that can hire people.

The WordNet ontology is widely used in natural language understanding applications. However, there are many words in WordNet that require additional ontological categorizations. In general, this is also a problem for all similarity metrics defined over WordNet, such as [12]. For example, “farm” is only categorized as a *workplace.n.01* and “publisher” is categorized as a *business_organization.n.01* and a *person.n.01*. From an intuitive point of view, we sense that business organizations and workplaces are similar. However, only *entity.n.01* subsumes both *business_organization.n.01* and *workplace.n.01*. Since *entity.n.01* is the most general concept in the ontology, any existing WordNet similarity metric used to calculate the similarity between these two concepts would return a very low score. Our algorithm can categorize “farm” under an additional concept (e.g., *business_organization.n.01*) which would make “publisher” and “farm” more similar.

Many words, particularly proper nouns, are also missing some common senses. For example, the word “Maria” has only two senses: “a dark region on the surface of the moon,” and “valuable timber tree of Panama.” In this case, “Maria” should also be categorized under *person.n.01*. Of course, the problem is not limited only to proper nouns. In addition, many words include uncommon senses. For example, “computer” has a sense that means “an expert at calculation.” However, in practice “computer” will almost always refer to a machine for performing calculations automatically.

The unsupervised algorithm proposed in this paper attempts to address the above concerns by discovering additional ontological categories for words in WordNet 3.0. The output is a new ontology in which upper-level concepts are labeled with a subset of concepts from the upper-level WordNet ontology. However, words in the new ontology are not categorized in the same way as words in WordNet. The categories chosen for each word reflect its primary senses as used in the naturally-occurring unlabeled text corpora that is used to train the algorithm. As an example of the program’s output, consider the word “principality”, which in WordNet is only categorized under *domain.n.02*, a subconcept of *location.n.01*. Our algorithm categorizes “principality” under *administrative_district.n.01*, which is subsumed by *location.n.01*, and *political_unit.n.01*, which is subsumed by *social_group.n.01*.

The rest of the paper is organized as follows. Some terminology and background is found in Section 2. An overview of our method is presented in Section 3. We present some work that is related to our method in Section 4. The details of the construction of word vectors are explained in Section 5. The details of the construction of concept vectors are explained in Section 6. We show how ontological categories are discovered for words in Section 7.

2 Terminology

The documentation for WordNet can be found in [9], but we give a brief introduction. The WordNet ontology is made up of *synsets*. A synset is a set of synonymous words that represents a concept. For example, the synset *pen.n.03*, which has the gloss, "a portable enclosure in which babies may be left to play," is {pen, playpen}. This means that "pen" or "playpen" can be used in some context to mean *pen.n.03*. A *sense* of a word w refers to one of the synsets that the word w is a member of.

Synsets are related to other synsets view hypernymy. Synset A is a hypernym (i.e., superconcept) of synset B if native speakers accept the statement, "B is a (kind of) A." Hyponymy is the inverse relationship. However, we will use the terms *subconcept* and *superconcept* instead of *hyponym* and *hypernym*. Also, we will favor the term *concept* or *ontological category* over *synset*. When referencing concepts in WordNet, we will use a notation like *xyz.n.03*; this instance is simply the label of the third ontological category to which the word "xyz" has been assigned, or equivalently, this is the label of the third sense of the word "xyz".

We will say that concept A is a *descendant* of concept B if it is possible to start at A and traverse only superconcepts to arrive at B (or to start at B and traverse only subconcepts to arrive at A). For example, *farm.n.01* is a subconcept of *workplace.n.01*, which in turn is a subconcept of *geographic_point.n.01*, and therefore *farm.n.01* is a descendant of *geographic_point.n.01*. The inverse relationship is *ancestor*, e.g., *geographic_point.n.01* is an ancestor of *farm.n.01*.

3 Our Method

The overall system architecture is shown in Figure 1. The approach first requires parsing a large untagged corpus to build feature vector representations that characterize the local syntactic context of each noun; these vectors are referred to as "word vectors." This approach assumes the *distributional hypothesis* of [5], which holds that words with similar meanings appear in similar grammatical contexts. For example, the words "rice" and "pasta" both share similar sets of verbs that they can be objects of: they are commonly "boiled," "eaten," "cooked" or "flavored," but not usually "calculated" or "knocked up". This stems from the fact that the verbs "eat," "cook" and "flavor" tend to select for words which are categorized under the ontological concept for *food* to appear as their objects' head nouns.

Using these word vectors, vectors that characterize the syntactic context of WordNet concepts are created, and are referred to as "concept vectors." More precisely, a concept vector for a concept c is a feature vector that characterizes the local syntactic context of words categorized exclusively under c. This is done by using WordNet ontology to find pairs of words that together unambiguously identify the target concept, and then creating a new feature vector by taking the common features of the two word vectors corresponding to the two words. For a given concept, many such feature vectors are created and merged to create the corresponding concept vector. For example, "port" is ambiguous and can mean a *location.n.01* or a *wine.n.01*, among other things. The word "wine" is also ambiguous because it can mean a *beverage.n.01* or (rarely) a *chromatic_color.n.01*. By taking the intersecting features of the word vectors of "port" and "wine", it is possible to isolate some of the features that represent the syntactic contexts of words categorized exclusively under *beverage.n.01*.

Since most concept vectors are extremely sparse and tend to represent low level

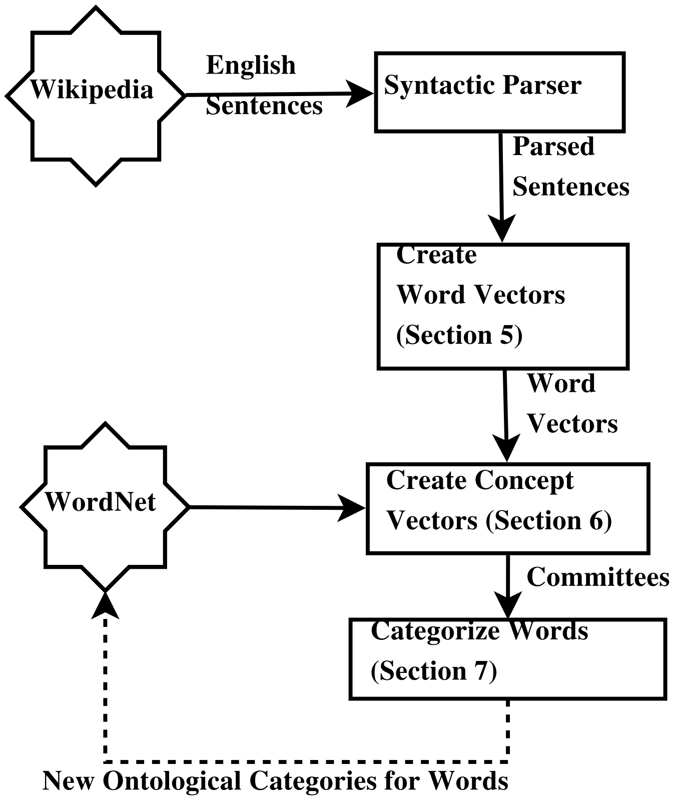
Wikipedia
English Sentences
Syntactic Parser
Parsed Sentences
Create Word Vectors (Section 5)
Word Vectors
WordNet
Create Concept Vectors (Section 6)
Committees
Categorize Words (Section 7)
New Ontological Categories for Words

Figure 1: Overall System Architecture

concepts, the concept vectors are clustered to create a set of upper-level concepts which form the upper-level of our new ontology. These clusters are referred to as "committees." Each committee has a label X, which is named after an upper-level concept in the WordNet ontology. Our clustering method ensures that for any concept c whose concept vector was used to form the committee X, c must be a direct or indirect sub-concept of X or c must equal X. This last statement means that the concept vectors of concepts from dissimilar parts of the ontology are not merged into the same committee even if those concept vectors are similar.

Following this step, the word vector corresponding to each word w is assigned to its most similar committees in order to discover the ontological categories or senses of w. The overlapping features of the word vector and the top ranking committee is removed from the word vector, and the word vector is again assigned to its most similar committees. The process repeats until the word vector is no longer similar to any committee or the number of features in the word vector is too small. Removing the overlapping features from the word vector allows for discovering the less frequent senses of the word which would otherwise be masked by the features of the dominant senses.

4 Related Work

This paper is partially motivated by some of the ideas presented in [4], which is a catalog of modifications to the upper level of the WordNet 2.1 ontology. The problems in that work were uncovered by a semantic interpreter which failed to correctly interpret certain sentences. One example of such a failure is the sentence, "The characters were engraved on the old coin." The verb "engrave" here means to carve, cut or etch into a physical material or surface. In both WordNet 2.1 and 3.0, the only sense for "coin" is categorized under *system_of_measurement.n.01*, which is an *abstraction.n.06* and not a *physical_object.n.01*. The verb predicate corresponding to the correct sense of "engrave" cannot allow for words only categorized as *abstraction.n.06* to appear as the head of the object, so interpreting the sentence is not possible.

The approach taken there was usually to tangle the hierarchy (i.e., introduce multiple inheritance), although sometimes concepts were moved from one part of the ontology to another. An example of tangling would be making *coin.n.01* also a subconcept of *physical_object.n.01*, which would resolve the problem above. This would fix the interpretation not only for the word "coin", but also for many other words which are categorized under *coin.n.01*, such as "nickel" or "fivepence". In this work, the hierarchy is not tangled and the upper level hierarchy remains the same; instead, it is words that are categorized under various ontological concepts. For example, "coin", "nickel", "fivepence" may be categorized under *physical_object.n.01* and *system_of_measurement.n.01*. These two approaches are roughly equivalent. The main differences between this work and [4] is that the former is unsupervised and the latter was mostly manual, although aided by an existing semantic interpreter. Overall, the goal of this work is to find similar problems in WordNet 3.0 and fix them automatically.

In general, the WordNet ontology must be organized in a way that reflects the ontology of natural language [4]. For example, it is possible to argue that in an ontology, the concept *process* should subsume the concept *physical object* because all physical objects have a temporal nature: they come into existence at one time instance and eventually come out of existence. However, this reasoning is problematic, as the ontology of natural language does make many distinctions between physical objects and entities that are conventionally thought of as processes, such as earthquakes. Physical objects like rocks may be "picked up" and "thrown" but earthquakes may not; earthquakes may be "experienced" or "forecast", but physical objects may not, at least not in the same sense.

The automatic acquisition of selectional preferences for verbs with respect to WordNet classes has also been an active area of research, with works such as [1] and [11]. The former work, which used a sense-tagged corpus to learn selectional preferences for verbs, revealed instances of the general problems described here; for example, words such as "temple" or "synagogue" which are only categorized as a *building.n.01* but not a *social_group.n.01* caused their system to fail to learn the correct selectional preferences for some verbs. In this particular case, the approach proposed here does categorize both "synagogue" and "temple" as an *institution.n.01*, which is a *social_group.n.01*.

Other approaches that attempt to acquire lexical relations from corpora include [13], [2] and [3]. These approaches use mainly lexico-syntactic patterns, as suggested by [6]. The latter work, VerbOcean, uses patterns to acquire different kinds of verb relations; the former two works use patterns to discover hypernymy relations between nouns and automatically construct ontologies. A typical example of such a pattern is NP_0 SUCH AS NP_1, which matches fragments like "local businesses such as taverns," and "hobbies such as stamp collecting." These fragments would be used as evidence for "tavern" being a subconcept of "business" and "stamp collecting" being a subconcept of "hobby."

Our method also shares similarities with some approaches that learn clusters of similar words but do not attempt to build concept hierarchies, as in [7] and especially [10]. Both of these works use the distributional hypothesis as we do to find similarities between words. The latter paper proposes an algorithm called Cluster by Committee (CBC), which is where we get the name "committee." The approach there is to first create a set of word vectors in much the same way we do, and then to create a set of tight clusters of word vectors called committees; each committee represents a distinct concept. As many committees as possible are created on the condition that no two committees are similar. Word vectors are assigned to the committees in order to discover senses for each word in a method that is similar to ours. Hence, it is reasonable to view our algorithm as as extension of CBC. The main differences between their algorithm and ours is in the way the committees are created and what they represent. Our committees correspond to WordNet ontological categories, so the output of our algorithm can be used to extend WordNet.

5 Representing Words

Following [10], each word w is represented by a feature vector, which will be referred to as a "word vector." Each feature corresponds to a local syntactic context in which a word may appear. Let $F_c(w)$ be the frequency with which word w occurs in context c. The value of the feature is the pointwise mutual information PMI(w, c) between the word w and the context c:

$$\text{PMI}(w,c) = log \frac{\frac{F_c(w)}{N}}{\frac{\sum\limits_i F_i(w)}{N} * \frac{\sum\limits_j F_c(j)}{N}} \tag{1}$$

In the above definition, the index i ranges over all contexts and the index j ranges over all words. $N = \sum_i \sum_j F_i(j)$ is the frequency with which all words occur in all contexts.

The following list presents each type of feature with an example feature extracted from the sentence, "Migratory birds from Florida State fly South in the Winter."

- SUBJECT-HEAD-OF(v). The target noun is the head of a simple noun phrase that is the subject of a verb v. The feature SUBJECT-HEAD-OF(fly) is extracted for the word "bird."

- OBJECT1-HEAD-OF(v). The target noun is the head of a simple noun phrase that is the first object of a verb v. The feature OBJECT1-HEAD-OF(fly) is extracted for the word "State."

- OBJECT2-HEAD-OF(v). The target noun is the head of a simple noun phrase that is the second object of a verb v. No feature of this type is extracted for any word in the sentence because there are no ditransitive verbs.

- MODIFIES-HEAD(n). The target noun acts as a modifier for a head noun n of a noun phrase. The feature MODIFIES-HEAD(State) is extracted for the word "Florida."

- HEAD-OF-PP-ATTACHED-TO-VERB(p, v). The target noun is the head of a simple noun phrase contained in a prepositional phrase headed by preposition p; that prepositional phrase is attached to the verb v. The feature HEAD-OF-PP-ATTACHED-TO-VERB(in, fly) is extracted for the word "Winter."

- HEAD-OF-PP-ATTACHED-TO-NP(p, n). The target noun is the head of a simple noun phrase contained in a prepositional phrase headed by preposition p; that prepositional phrase is attached to a simple noun phrase whose head is the noun n. The feature HEAD-OF-PP-ATTACHED-TO-NP(from, bird) is extracted for the word "State."

- HEAD-OF-NP-WITH-PP-ATTACHED(p, n). The target noun is the head of a simple noun phrase to which a prepositional phrase headed by prepositional p attaches; that prepositional phrase contains a simple noun phrase headed by noun n. The feature HEAD-OF-NP-WITH-PP-ATTACHED(from, State) is extracted for the word "bird."

A parser was necessary to find out if a given prepositional phrase attached to a noun phrase or a verb, and for identifying the subject-verb and object-verb relationships. To this end, we used a modified version of the Charniak parser to parse a large portion of Wikipedia. This modified parser made certain grammatical relationships explicit that the vanilla parser did not. The word vectors were constructed from over 8.5 million verb clauses.

6 Representing Concepts

A concept vector for a concept c is a feature vector that characterizes the local syntactic context of words categorized exclusively under c. This section presents the details of how we create concept vectors from the existing set of word vectors and the WordNet 3.0 ontology.

Our method requires a similarity measure between any two concepts c_1 and c_2 in the WordNet ontology. We use the *information content* based similarity measure from [8]. The information content $IC(c)$ of a concept c, originally introduced by [12], is defined as $IC(c) = -logp(c)$, where $p(c)$ is the estimated probability of a concept c occurring in some corpora. Intuitively, the more general upper-level concepts in the ontology have lower information content than the more specific lower-level concepts.

For example, *chair.n.01* has a higher information content than *furniture.n.01*. The *least common subsumer* $LCS(c_1, c_2)$ of two concepts c_1 and c_2 is defined as the information content of a concept which subsumes both c_1 and c_2 and has the highest information content out of all concepts that subsume both c_1 and c_2. The similarity between two concepts c_1 and c_2 is given by:

$$sim(c_1, c_2) = \frac{2 \times IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)} \tag{2}$$

Whenever we say that concepts c_1 and c_2 are *similar* it means that $sim(c_1, c_2) > \sigma$, for some large threshold σ between 0 and 1. If we say that c_1 and c_2 are *not similar* it means that $sim(c_1, c_2) < \sigma'$, for some low threshold σ' between 0 and 1. We also require the following two definitions.

Definition 1 *The union $\vec{v_1} \cup \vec{v_2}$ of two feature vectors $\vec{v_1}$ and $\vec{v_2}$ yields a third feature vector containing every feature of $\vec{v_1}$ and $\vec{v_2}$. If a feature is common to both vectors, the feature with the larger of the two values is chosen.*

Definition 2 *The intersection $\vec{v_1} \cap \vec{v_2}$ of two feature vectors $\vec{v_1}$ and $\vec{v_2}$ yields a third feature vector containing the common features of $\vec{v_1}$ and $\vec{v_2}$. If a common feature has different values in both vectors, the minimum of the two values is chosen.*

In order to determine if two word vectors can be intersected to unambiguously identify a target concept, consider two distinct words w_1 and w_2, and a target concept c. Assume that both of these words have a sense that is similar to c, so that the intersection of their word vectors will in part include the syntactic context of c. The rule for determining if the intersection is unambiguous is given by the following definition.

Definition 3 *The intersection of two word vectors $\vec{w_1}$ and $\vec{w_2}$ corresponding to two distinct words w_1 and w_2, respectively, is said to be unambiguous if any sense c' of w_1 that is not similar to c is also not similar to any sense of w_2.*

To see why this works, suppose this condition does not hold. Let c' be a sense of w_1 that is not similar to c, and c'' be a sense of w_2 that is similar to c'. By transitivity of similarity, c'' is also not similar to c. Hence, w_1 has a sense c' and w_2 has a sense c'' that are similar to each other but both not similar to the target concept c. Therefore, the intersection of the syntactic contexts of w_1 and w_2 will also include the contexts for c' and c'', both of which are not similar to the target concept c.

The algorithm below makes use of the words in WordNet synsets (i.e., concepts). When referring to the set of words that make up the synset c, we will write $synset(c)$. Let $L_1(c)$ be the set that contains the words in $synset(c)$. Let $L_2(c)$ be the set that contains the words in $L_1(c)$ and the words in $\{w \in synset(c') : c'$ is a descendant of c and $sim(c, c') > \sigma\}$. A concept vector $\vec{c}$ for every concept c in WordNet is constructed using Algorithm 1.

To illustrate the creation of concept vectors, consider the concepts *religion.n.01* (a religious belief) and *religion.n.02* (organized religion). The words "religion" and "faith" are in the synsets of both of these concepts. Hence, taking the intersecting

Algorithm 1 MAKE-CV(c)

```
1: Initialize $\vec{c}$ to an empty feature vector
2: for each pair of words $l_1, l_2 \in L_1(c) \times L_2(c)$ do
3:     if $l_1 \neq l_2$ and $\vec{l_1} \cap \vec{l_2}$ is unambiguous then
4:         $\vec{c} \leftarrow \vec{c} \cup (\vec{l_1} \cap \vec{l_2})$
5:     end if
6: end for
7: return $\vec{c}$
```

syntactic features of both of these words (i.e., taking the intersection of their word vectors) would not unambiguously identify either of these concepts.

Now consider "theism", which is word that is in the synset for *theism.n.01*, a sub-concept of *religion.n.01*. It is possible to take the intersection of the word vectors of "theism" and "faith" to get the context of *religion.n.01*. This is because no sense of "faith" that is dissimilar to *religion.n.01* is similar to any sense of "theism". Of course, the performance of this heuristic relies on the correctness of WordNet.

After the concept vectors are created, they are clustered to produce a set of committees (see Section 3.) The algorithm makes use of the similarity between two feature vectors $\vec{v_1}$ and $\vec{v_2}$, which is defined using the cosine similarity metric:

$$\text{COSINE-SIM}(\vec{v_1}, \vec{v_2}) = \frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\| \|\vec{v_2}\|} \tag{3}$$

In the algorithm below, $C(c)$ denotes the concept vector of concept c and $W(w)$ denotes the word vector of word w. CLUSTER(c) is a feature vector that is formed from taking the union of all concept vectors of concepts that are descendants of c along with the concept vector of c itself. FEATURE-COUNT(c) denotes the number of features of c.

Algorithm 2 MAKE-COMMITTEES(c)

```
1: for each subconcept $h$ of $c$ do
2:     $C(h) \leftarrow$ MAKE-COMMITTEES($h$)
3:     $sim \leftarrow$ COSINE-SIM($C(h)$, CLUSTER($c$))
4:     $fnum \leftarrow$ FEATURE-COUNT($C(h)$)
5:     if $sim \geq \theta$ or $fnum < \lambda$ then
6:         $C(c) \leftarrow C(c) \cup C(h)$
7:     else
8:         Add $C(h)$ with label $h$ to set of committees
9:     end if
10: end for
11: return $C(c)$
```

The algorithm visits concepts in a depth-first, post-order traversal, beginning with he most general concept *entity.n.01*. If the procedure is called on concept c, then for each subconcept h of c, it will do the following:

1. It will call itself using h as its argument. This call may result in committees with labels that are h or descendants of h added to the set of committees.

2. The recursive call returns a new concept vector $C(h)$ for h, which represents the centroid of a cluster that includes some concept vectors in set $\{C(h)\} \cup \{C(h') : h' \text{ is a descendant of } h\}$. The concept vectors which were chosen for the cluster represented by $C(h)$ were the concept vectors not yet chosen to be part of any committee.

3. If $C(h)$ is similar to $\text{CLUSTER}(c)$ or $C(h)$ has few features, then $C(h)$ is unioned with $C(c)$, and the result is stored back in $C(c)$. Otherwise, $C(h)$ is added to the set of committees. The procedure returns $C(c)$.

The algorithm has parameters λ and θ. Parameter λ is the minimum number of features a committee can have; increasing it also tends to create fewer committees, and those which do get created tend represent higher level concepts. Parameter θ controls how course grained the committees are allowed to be: the lower this value, the fewer committees get created and the more course grained they are.

In step 3, we use the heuristic of comparing the feature vectors $\text{CLUSTER}(c)$ and $C(h)$, and if they are dissimilar then $C(h)$ is added to the set of committees. The justification is that if these two vectors are similar, then the concept h is not well-distinguished from its sibling concepts, so in this case $C(h)$ and $C(c)$ are merged. If they are dissimilar, then h is well-distinguished from its siblings and so $C(h)$ should stand as its own committee. There is a tendency of upper-level WordNet concepts to have a set of subconcepts that have very little in common. For example, *artifact.n.01* has subconcepts *instrumentality.n.03*, *excavation.n.03*, *decoration.n.01*, etc., that do not have much in common besides being artifacts. Therefore, CLUSTER(*artifact.n.01*) is dissimilar to most feature vectors formed from the subconcepts of *artifact.n.01*, so many of its subconcepts will be used as the labels of new committees.

7 Categorizing Words

The algorithm below discovers ontological categories for each word w by assigning the word vector of w to its most similar committees. For each word w, do the following:

1. Let C be the set of committees,

$$c_{max} = \arg\max_{c \in C} \text{COSINE-SIM}(W(w), c)$$

   and $s_{max} = \text{COSINE-SIM}(c_{max}, W(w))$.

2. If $s_{max} > \beta$, then assign w to the ontological category represented by the label of c_{max}. This step assigns w to its most similar committee.

3. For each $c \in C$, if COSINE-SIM$(c, W(w)) > \beta'$, then assign w to the ontological category represented by the label of c. This step assigns w to all similar committees, but the parameter $\beta' > \beta$.
4. Remove from $W(w)$ the features of $W(w) \cap c_{max}$.
5. Repeat the above steps if FEATURE-COUNT$(W(w)) > \alpha$ and $s_{max} > \beta$.

We remove from the word vector of w the overlapping features between the word vector of w and the most similar assigned committee, as suggested by [10]. This allows the program to discover less common senses of w which are masked by dominating senses. Note that the word w is assigned to all similar committees, not just the top-most similar committee. However, a different threshold β', such that $\beta' > \beta$, is used for assigning w to the other committees which ranked lower than c_{max}. In our experiments, we chose $\beta = 0.067$ and $\beta' = 0.11$.

8 Evaluation

In order to evaluate the output of our program, we asked three human judges to evaluate if a set of random words were correctly categorized. The output is in the form of word-concept pairs; a given word-concept pair w, c is correct if native speakers accept the statement, "w can be a kind of c." For example, "chair can be a kind of furniture." An equivalent condition for correctness is, "w has a sense which is subsumed by c." A pair w, c was chosen only if

- $sim(c', c) < 0.30$, for all WordNet senses c' of w, where sim is given by Equation 2 and
- For all WordNet senses c' of w, c' is not a descendant of c.

The second condition filters out additional pairs that were not filtered by the first condition. These conditions ensure that the judges evaluate only novel ontological categorizations for words. A total of of 12721 words were categorized and there was a total of 18623 word-concept categorizations. Of these 18623, there were 7420 novel word-concept categorizations as determined by the rule above.

The judges were presented with 200 random novel word-concept pairs. For each pair, the judges were asked to evaluate whether any sense of the word could be subsumed by the concept. On average, **75.5%** of the pairs were judged to be correct. The average inter annotator agreement was **72.5%**.

Other approaches for discovering word senses or inducing ontologies have attempted automatic evaluation with respect to a gold standard such as WordNet. As pointed out by [10], such evaluations are problematic because the purpose of our work is to discover additional word categorizations that are not in WordNet.

9 Conclusion

The WordNet ontology is used in a variety of natural language understanding applications. The correct organization of the upper-level ontology is important for semantic interpretation, word sense disambiguation, the performance of any similarity metric implemented for that ontology and any other task where semantics are involved. We have presented a novel unsupervised approach for discovering additional ontological categorizations for words in WordNet. Evaluation by three human judges revealed an average of 75.5% of the additional categorizations were correct. The judges agreed with each other an average of 72.5% of the time. We plan to enrich the WordNet ontology with the discovered categorizations. One potential future direction is using our method to acquire domain specific categorizations for words, particularly by exploiting biomedical texts.

References

[1] Eneko Agirre and David Martinez. Learning class-to-class selectional preferences. In *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1117822.1117825.

[2] Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 120–126, Morristown, NJ, USA, 1999. Association for Computational Linguistics. ISBN 1-55860-609-3. doi: http://dx.doi.org/10.3115/1034678.1034705.

[3] Timothy Chklovski and Patrick Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 33–40, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[4] Fernando Gomez. Semantic interpretation and the upper-level ontology of wordnet. *Journal of Intelligent Systems*, 16(2):93–116, 2007.

[5] Zellig S. Harris. *Mathematical structures of language [by] Zellig Harris*. Interscience Publishers New York,, 1968. ISBN 0470353163.

[6] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/992133.992154.

[7] Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 268–275, Morristown, NJ, USA, 1990. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/981823.981857.

[8] Dekang Lin. An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8.

[9] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38 (11):39–41, 1995. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/219717.219748.

[10] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 613–619, 2002.

[11] Philip Resnik. Selectional preference and sense disambiguation, 1997.

[12] Philip 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.

[13] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, MA, 2005.