Which Noun Phrases Denote Which Concepts?

Jayant Krishnamurthy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jayantk@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cmu.edu

Abstract

Resolving polysemy and synonymy is required for high-quality information extraction. We present ConceptResolver, a component for the Never-Ending Language Learner (NELL) (Carlson et al., 2010) that handles both phenomena by identifying the latent concepts that noun phrases refer to. ConceptResolver performs both word sense induction and synonym resolution on relations extracted from text using an ontology and a small amount of labeled data. Domain knowledge (the ontology) guides concept creation by defining a set of possible semantic types for concepts. Word sense induction is performed by inferring a set of semantic types for each noun phrase. Synonym detection exploits redundant information to train several domain-specific synonym classifiers in a semi-supervised fashion. When ConceptResolver is run on NELL's knowledge base, 87% of the word senses it creates correspond to real-world concepts, and 85% of noun phrases that it suggests refer to the same concept are indeed synonyms.

1 Introduction

Many information extraction systems construct knowledge bases by extracting structured assertions from free text (e.g., NELL (Carlson et al., 2010), TextRunner (Banko et al., 2007)). A major limitation of many of these systems is that they fail to distinguish between noun phrases and the underlying concepts they refer to. As a result, a polysemous phrase like "apple" will refer sometimes to the concept *Apple Computer (the company)*, and other times to the concept *apple (the fruit)*. Furthermore, two synonymous noun phrases like "apple" and "Apple Computer" can refer to the same underlying concept. The result of ignoring this many-to-many mapping between noun phrases and underlying concepts (see Figure 1) is confusion about the meaning of extracted information. To minimize such confusion, a system must separately represent noun phrases, the underlying concepts to which they can refer, and the many-to-many "can refer to" relation between them.

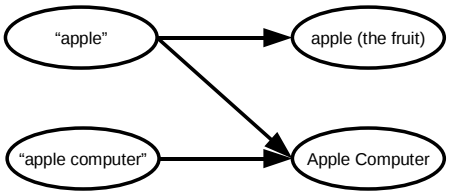

Figure 1: An example mapping from noun phrases (left) to a set of underlying concepts (right). Arrows indicate which noun phrases can refer to which concepts.

```
[eli lilly, lilly]
[kaspersky labs, kaspersky lab, kaspersky]
[careerbuilder, careerbuilder.com]
[l 3 communications, level 3 communications]
[cellular, u.s. cellular]
[jc penney, jc penny]
[nielsen media research, nielsen company]
[universal studios, universal music group, universal]
[amr corporation, amr]
[intel corp, intel corp., intel corporation, intel]

[emmitt smith, chris canty]
[albert pujols, pujols]
[carlos boozer, dennis martinez]
[jason hirsh, taylor buchholz]
[chris snyder, ryan roberts]
[j.p. losman, losman, jp losman]
[san francisco giants, francisco rodriguez]
[andruw jones, andruw]
[aaron heilman, bret boone]
[roberto clemente, clemente]
```

Figure 2: A random sample of concepts created by ConceptResolver. The first 10 concepts are from **company**, while the second 10 are from **athlete**.

The relations extracted by systems like NELL actually apply to concepts, not to noun phrases. Say

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 570–580,
Portland, Oregon, June 19-24, 2011.

the system extracts the relation **ceoOf**(x_1, x_2) between the noun phrases x_1 and x_2. The correct interpretation of this extracted relation is that there exist concepts c_1 and c_2 such that x_1 can refer to c_1, x_2 can refer to c_2, and **ceoOf**(c_1, c_2). If the original relation were **ceoOf**("steve", "apple"), then c_1 would be *Steve Jobs*, and c_2 would be *Apple Computer*. A similar interpretation holds for one-place category predicates like **person**(x_1). We define *concept discovery* as the problem of (1) identifying concepts like c_1 and c_2 from extracted predicates like **ceoOf**(x_1, x_2) and (2) mapping noun phrases like x_1, x_2 to the concepts they can refer to.

The main input to ConceptResolver is a set of extracted category and relation instances over noun phrases, like **person**(x_1) and **ceoOf**(x_1, x_2), produced by running NELL. Here, any individual noun phrase x_i can be labeled with multiple categories and relations. The output of ConceptResolver is a set of concepts, $\{c_1, c_2, ..., c_n\}$, and a mapping from each noun phrase in the input to the set of concepts it can refer to. Like many other systems (Miller, 1995; Yates and Etzioni, 2007; Lin and Pantel, 2002), ConceptResolver represents each output concept c_i as a set of synonymous noun phrases, i.e., $c_i = \{x_{i1}, x_{i2}, ..., x_{im}\}$. For example, Figure 2 shows several concepts output by ConceptResolver; each concept clearly reveals which noun phrases can refer to it. Each concept also has a semantic type that corresponds to a category in ConceptResolver's ontology; for instance, the first 10 concepts in Figure 2 belong to the category **company**.

Previous approaches to concept discovery use little prior knowledge, clustering noun phrases based on co-occurrence statistics (Pantel and Lin, 2002). In comparison, ConceptResolver uses a knowledge-rich approach. In addition to the extracted relations, ConceptResolver takes as input two other sources of information: an ontology, and a small number of labeled synonyms. The ontology contains a schema for the relation and category predicates found in the input instances, including properties of predicates like type restrictions on its domain and range. The category predicates are used to assign semantic types to each concept, and the properties of relation predicates are used to create evidence for synonym resolution. The labeled synonyms are used as training data during synonym resolution, where they are used to train a semi-supervised classifier.

1. **Induce Word Senses**
   i. Use extracted category instances to create one or more senses per noun phrase.
   ii. Use argument type constraints to produce relation evidence for synonym resolution.
2. **Cluster Synonymous Senses**
   For each category C defined in the ontology:
   i. Train a semi-supervised classifier to predict synonymy.
   ii. Cluster word senses with semantic type C using classifier's predictions.
   iii. Output sense clusters as concepts with semantic type C.

Figure 3: High-level outline of ConceptResolver's algorithm.

ConceptResolver discovers concepts using the process outlined in Figure 3. It first performs word sense induction, using the extracted category instances to create one or more unambiguous word senses for each noun phrase in the knowledge base. Each word sense is a copy of the original noun phrase paired with a semantic type (a category) that restricts the concepts it can refer to. ConceptResolver then performs synonym resolution on these word senses. This step treats the senses of each semantic type independently, first training a synonym classifier then clustering the senses based on the classifier's decisions. The result of this process is clusters of synonymous word senses, which are output as concepts. Concepts inherit the semantic type of the word senses they contain.

We evaluate ConceptResolver using a subset of NELL's knowledge base, presenting separate results for the concepts of each semantic type. The evaluation shows that, on average, 87% of the word senses created by ConceptResolver correspond to real-world concepts. We additionally find that, on average, 85% of the noun phrases in each concept refer to the same real-world entity.

2 Prior Work

Previous work on concept discovery has focused on the subproblems of word sense induction and synonym resolution. Word sense induction is typically performed using unsupervised clustering. In the SemEval word sense induction and disambigua-

tion task (Agirre and Soroa, 2007; Manandhar et al., 2010), all of the submissions in 2007 created senses by clustering the contexts each word occurs in, and the 2010 event explicitly disallowed the use of external resources like ontologies. Other systems cluster words to find both word senses and concepts (Pantel and Lin, 2002; Lin and Pantel, 2002). ConceptResolver's category-based approach is quite different from these clustering approaches. Snow et al. (2006) describe a system which adds new word senses to WordNet. However, Snow et al. assume the existence of an oracle which provides the senses of each word. In contrast, ConceptResolver automatically determines the number of senses for each word.

Synonym resolution on relations extracted from web text has been previously studied by Resolver (Yates and Etzioni, 2007), which finds synonyms in relation triples extracted by TextRunner (Banko et al., 2007). In contrast to our system, Resolver is unsupervised and does not have a schema for the relations. Due to different inputs, ConceptResolver and Resolver are not precisely comparable. However, our evaluation shows that ConceptResolver has higher synonym resolution precision than Resolver, which we attribute to our semi-supervised approach and the known relation schema.

Synonym resolution also arises in record linkage (Winkler, 1999; Ravikumar and Cohen, 2004) and citation matching (Bhattacharya and Getoor, 2007; Bhattacharya and Getoor, 2006; Poon and Domingos, 2007). As with word sense induction, many approaches to these problems are unsupervised. A problem with these algorithms is that they require the authors to define domain-specific similarity heuristics to achieve good performance. Other synonym resolution work is fully supervised (Singla and Domingos, 2006; McCallum and Wellner, 2004; Snow et al., 2007), training models using manually constructed sets of synonyms. These approaches use large amounts of labeled data, which can be difficult to create. ConceptResolver's approach lies between these two extremes: we label a small number of synonyms (10 pairs), then use semi-supervised training to learn a similarity function. We think our technique is a good compromise, as it avoids much of the manual effort of the other approaches: tuning the similarity function in one case, and labeling a large amount of data in the other

ConceptResolver uses a novel algorithm for semi-supervised clustering which is conceptually similar to other work in the area. Like other approaches (Basu et al., 2004; Xing et al., 2003; Klein et al., 2002), we learn a similarity measure for clustering based on a set of must-link and cannot-link constraints. Unlike prior work, our algorithm exploits multiple views of the data to improve the similarity measure. As far as we know, ConceptResolver is the first application of semi-supervised clustering to relational data – where the items being clustered are connected by relations (Getoor and Diehl, 2005). Interestingly, the relational setting also provides us with the independent views that are beneficial to semi-supervised training.

Concept discovery is also related to coreference resolution (Ng, 2008; Poon and Domingos, 2008). The difference between the two problems is that coreference resolution finds noun phrases that refer to the same concept within a specific document. We think the concepts produced by a system like ConceptResolver could be used to improve coreference resolution by providing prior knowledge about noun phrases that can refer to the same concept. This knowledge could be especially helpful for cross-document coreference resolution systems (Haghighi and Klein, 2010), which actually represent concepts and track mentions of them across documents.

3 Background: Never-Ending Language Learner

ConceptResolver is designed as a component for the Never-Ending Language Learner (NELL) (Carlson et al., 2010). In this section, we provide some pertinent background information about NELL that influenced the design of ConceptResolver [1].

NELL is an information extraction system that has been running 24x7 for over a year, using coupled semi-supervised learning to populate an ontology from unstructured text found on the web. The ontology defines two types of predicates: *categories* (e.g., **company** and **CEO**) and *relations* (e.g., **ceoOfCompany**). Categories are single-argument predicates, and relations are two-argument predicates.

[1] More information about NELL, including browsable and downloadable versions of its knowledge base, is available from `http://rtw.ml.cmu.edu`.

NELL's *knowledge base* contains both definitions for predicates and extracted instances of each predicate. At present, NELL's knowledge base defines approximately 500 predicates and contains over half a million extracted instances of these predicates with an accuracy of approximately 0.85.

Relations between predicates are an important component of NELL's ontology. For ConceptResolver, the most important relations are **domain** and **range**, which define argument types for each relation predicate. For example, the first argument of **ceoOfCompany** must be a **CEO** and the second argument must be a **company**. Argument type restrictions inform ConceptResolver's word sense induction process (Section 4.1).

Multiple sources of information are used to populate each predicate with high precision. The system runs four independent extractors for each predicate: the first uses web co-occurrence statistics, the second uses HTML structures on webpages, the third uses the morphological structure of the noun phrase itself, and the fourth exploits empirical regularities within the knowledge base. These subcomponents are described in more detail by Carlson et al. (2010) and Wang and Cohen (2007). NELL learns using a bootstrapping process, iteratively re-training these extractors using instances in the knowledge base, then adding some predictions of the learners to the knowledge base. This iterative learning process can be viewed as a discrete approximation to EM which does not explicitly instantiate every latent variable.

As in other information extraction systems, the category and relation instances extracted by NELL contain polysemous and synonymous noun phrases. ConceptResolver was developed to reduce the impact of these phenomena.

4 ConceptResolver

This section describes ConceptResolver, our new component which creates concepts from NELL's extractions. It uses a two-step procedure, first creating one or more senses for each noun phrase, then clustering synonymous senses to create concepts.

4.1 Word Sense Induction

ConceptResolver induces word senses using a simple assumption about noun phrases and concepts. If a noun phrase has multiple senses, the senses should be distinguishable from context. People can determine the sense of an ambiguous word given just a few surrounding words (Kaplan, 1955). We hypothesize that local context enables sense disambiguation by defining the semantic type of the ambiguous word. ConceptResolver makes the simplifying assumption that *all* word senses can be distinguished on the basis of semantic type. As the category predicates in NELL's ontology define a set of possible semantic types, this assumption is equivalent to the *one-sense-per-category assumption*: a noun phrase refers to at most one concept in each category of NELL's ontology. For example, this means that a noun phrase can refer to a **company** and a **fruit**, but not multiple companies.

ConceptResolver uses the extracted category assertions to define word senses. Each word sense is represented as a tuple containing a noun phrase and a category. In synonym resolution, the category acts like a type constraint, and only senses with the same category type can be synonymous. To create senses, the system interprets each extracted category predicate $c(x)$ as evidence that category c contains a concept denoted by noun phrase x. Because it assumes that there is at most one such concept, ConceptResolver creates one sense of x for each extracted category predicate. As a concrete example, say the input assertions contain **company**("apple") and **fruit**("apple"). Sense induction creates two senses for "apple": ("apple", company) and ("apple", fruit).

The second step of sense induction produces evidence for synonym resolution by creating relations between word senses. These relations are created from input relations and the ontology's argument type constraints. Each extracted relation is mapped to all possible sense relations that satisfy the argument type constraints. For example, the noun phrase relation **ceoOfCompany**("steve jobs", "apple") would map to **ceoOfCompany**(("steve jobs", ceo), ("apple", company)). It would not map to a similar relation with ("apple", fruit), however, as ("apple", fruit) is not in the range of **ceoOfCompany**. This process is effective because the relations in the ontology have restrictive domains and ranges, so only a small fraction of sense pairs satisfy the argument type restrictions. It is also not vital that this mapping be perfect, as the sense relations are only

used as evidence for synonym resolution. The final output of sense induction is a sense-disambiguated knowledge base, where each noun phrase has been converted into one or more word senses, and relations hold between pairs of senses.

4.2 Synonym Resolution

After mapping each noun phrase to one or more senses (each with a distinct category type), ConceptResolver performs semi-supervised clustering to find synonymous senses. As only senses with the same category type can be synonymous, our synonym resolution algorithm treats senses of each type independently. For each category, ConceptResolver trains a semi-supervised synonym classifier then uses its predictions to cluster word senses.

Our key insight is that semantic relations and string attributes provide independent views of the data: we can predict that two noun phrases are synonymous either based on the similarity of their text strings, or based on similarity in the relations NELL has extracted about them. As a concrete example, we can decide that ("apple computer", company) and ("apple", company) are synonymous because the text string "apple" is similar to "apple computer," or because we have learned that ("steve jobs", ceo) is the CEO of both companies. ConceptResolver exploits these two independent views using co-training (Blum and Mitchell, 1998) to produce an accurate synonym classifier using only a handful of labels.

4.2.1 Co-Training the Synonym Classifier

For each category, ConceptResolver co-trains a pair of synonym classifiers using a handful of labeled synonymous senses and a large number of automatically created unlabeled sense pairs. Co-training is a semi-supervised learning algorithm for data sets where each instance can be classified from two (or more) independent sets of features. That is, the features of each instance x_i can be partitioned into two views, $x_i = (x_i^1, x_i^2)$, and there exist functions in each view, f^1, f^2, such that $f^1(x_i^1) = f^2(x_i^2) = y_i$. The co-training algorithm uses a bootstrapping procedure to train f^1, f^2 using a small set of labeled examples L and a large pool of unlabeled examples U. The training process repeatedly trains each classifier on the labeled examples, then allows each classifier to label some examples in the unlabeled data pool. Co-training also has PAC-style theoretical guarantees which show that it can learn classifiers with arbitrarily high accuracy under appropriate conditions (Blum and Mitchell, 1998).

Figure 4 provides high-level pseudocode for co-training in the context of ConceptResolver. In ConceptResolver, an instance x_i is a pair of senses (e.g., <("apple", company), ("microsoft", company)>), the two views x_i^1 and x_i^2 are derived from string attributes and semantic relations, and the output y_i is whether the senses are synonyms. (The features of each view are described later in this section.) L is initialized with a small number of labeled sense pairs. Ideally, U would contain all pairs of senses in the category, but this set grows quadratically in category size. Therefore, ConceptResolver uses the canopies algorithm (McCallum et al., 2000) to initialize U with a subset of the sense pairs that are more likely to be synonymous.

Both the string similarity classifier and the relation classifier are trained using L_2-regularized logistic regression. The regularization parameter λ is automatically selected on each iteration by searching for a value which maximizes the loglikelihood of a validation set, which is constructed by randomly sampling 25% of L on each iteration. λ is re-selected on each iteration because the initial labeled data set is extremely small, so the initial validation set is not necessarily representative of the actual data. In our experiments, the initial validation set contains only 15 instances.

The string similarity classifier bases its decision on the original noun phrase which mapped to each sense. We use several string similarity measures as features, including SoftTFIDF (Cohen et al., 2003), Level 2 JaroWinkler (Cohen et al., 2003), Fellegi-Sunter (Fellegi and Sunter, 1969), and Monge-Elkan (Monge and Elkan, 1996). The first three algorithms produce similarity scores by matching words in the two phrases and the fourth is an edit distance. We also use a heuristic abbreviation detection algorithm (Schwartz and Hearst, 2003) and convert its output into a score by dividing the length of the detected abbreviation by the total length of the string.

The relation classifier's features capture several intuitive ways to determine that two items are synonyms from the items they are related to. The relation view contains three features for each relation

For each category C:

1. Initialize labeled data L with 10 positive and 50 negative examples (pairs of senses)
2. Initialize unlabeled data U by running canopies (McCallum et al., 2000) on all senses in C.
3. Repeat 50 times:
   i. Train the string similarity classifier on L
   ii. Train the relation classifier on L
   iii. Label U with each classifier
   iv. Add the most confident 5 positive and 25 negative predictions of both classifiers to L

Figure 4: The co-training algorithm for learning synonym classifiers.

r whose domain is compatible with the current category. Consider the sense pair (s, t), and let $r(s)$ denote s's values for relation r (i.e., $r(s) = \{v : r(s, v)\}$). For each relation r, we instantiate the following features:

- **(Senses which share values are synonyms)** The percent of values of r shared by both s and t, that is $\frac{|r(s) \cap r(t)|}{|r(s) \cup r(t)|}$.
- **(Senses with different values are not synonyms)** The percent of values of r not shared by s and t, or $1 - \frac{|r(s) \cap r(t)|}{|r(s) \cup r(t)|}$. The feature is set to 0 if either $r(s)$ or $r(t)$ is empty. This feature is only instantiated if the ontology specifies that r has at most one value per sense.
- **(Some relations indicate synonymy)** A boolean feature which is true if $t \in r(s)$ or $s \in r(t)$.

The output of co-training is a pair of classifiers for each category. We combine their predictions using the assumption that the two views X^1, X^2 are conditionally independent given Y. As we trained both classifiers using logistic regression, we have models for the probabilities $P(Y|X^1)$ and $P(Y|X^2)$. The conditional independence assumption implies that we can combine their predictions using the formula:

$$P(Y = 1|X^1, X^2) = \frac{P(Y = 1|X^1)P(Y = 1|X^2)P(Y = 0)}{\sum_{y=0,1} P(Y = y|X^1)P(Y = y|X^2)(1 - P(Y = y))}$$

The above formula involves a prior term, $P(Y)$, because the underlying classifiers are discriminative. We set $P(Y = 1) = .5$ in our experiments as this setting reduces our dependence on the (typically poorly calibrated) probability estimates of logistic regression. We also limited the probability predictions of each classifier to lie in $[.01, .99]$ to avoid divide-by-zero errors. The probability $P(Y|X^1, X^2)$ is the final synonym classifier which is used for agglomerative clustering.

4.2.2 Agglomerative Clustering

The second step of our algorithm runs agglomerative clustering to enforce transitivity constraints on the predictions of the co-trained synonym classifier. As noted in previous works (Snow et al., 2006), synonymy is a transitive relation. If a and b are synonyms, and b and c are synonyms, then a and c must also be synonyms. Unfortunately, co-training is not guaranteed to learn a function that satisfies these transitivity constraints. We enforce the constraints by running agglomerative clustering, as clusterings of instances trivially satisfy the transitivity property.

ConceptResolver uses the clustering algorithm described by Snow et al. (2006), which defines a probabilistic model for clustering and a procedure to (locally) maximize the likelihood of the final clustering. The algorithm is essentially bottom-up agglomerative clustering of word senses using a similarity score derived from $P(Y|X^1, X^2)$. The similarity score for two senses is defined as:

$$\log \frac{P(Y = 0)P(Y = 1|X^1, X^2)}{P(Y = 1)P(Y = 0|X^1, X^2)}$$

The similarity score for two clusters is the sum of the similarity scores for all pairs of senses. The agglomerative clustering algorithm iteratively merges the two most similar clusters, stopping when the score of the best possible pair is below 0. The clusters of word senses produced by this process are the concepts for each category.

5 Evaluation

We perform several experiments to measure ConceptResolver's performance at each of its respective tasks. The first experiment evaluates word sense induction using Freebase as a canonical set of concepts. The second experiment evaluates synonym resolution by comparing ConceptResolver's sense clusters to a gold standard clustering.

For both experiments, we used a knowledge base created by running 140 iterations of NELL. We preprocessed this knowledge base by removing all noun

phrases with zero extracted relations. As ConceptResolver treats the instances of each category predicate independently, we chose 7 categories from NELL's ontology to use in the evaluation. The categories were selected on the basis of the number of extracted relations that ConceptResolver could use to detect synonyms. The number of noun phrases in each category is shown in Table 2. We manually labeled 10 pairs of synonymous senses for each of these categories. The system automatically synthesized 50 negative examples from the positive examples by assuming each pair represents a distinct concept, so senses in different pairs are not synonyms.

5.1 Word Sense Induction Evaluation

Our first experiment evaluates the performance of ConceptResolver's category-based word sense induction. We estimate two quantities: (1) *sense precision*, the fraction of senses created by our system that correspond to real-world entities, and (2) *sense recall*, the fraction of real-world entities that ConceptResolver creates senses for. Sense recall is only measured over entities which are represented by a noun phrase in ConceptResolver's input assertions – it is a measure of ConceptResolver's ability to create senses for the noun phrases it is given. Sense precision is directly determined by how frequently NELL's extractors propose correct senses for noun phrases, while sense recall is related to the correctness of the one-sense-per-category assumption.

Precision and recall were evaluated by comparing the senses created by ConceptResolver to concepts in Freebase (Bollacker et al., 2008). We sampled 100 noun phrases from each category and matched each noun phrase to a set of Freebase concepts. We interpret each matching Freebase concept as a sense of the noun phrase. We chose Freebase because it had good coverage for our evaluation categories.

To align ConceptResolver's senses with Freebase, we first matched each of our categories with a set of similar Freebase categories[2]. We then used a combination of Freebase's search API and Mechanical Turk to align noun phrases with Freebase concepts: we searched for the noun phrase in Freebase, then had Mechanical Turk workers label which of the top 10 resulting Freebase concepts the noun phrase could refer to. After obtaining the list of matching Freebase concepts for each noun phrase, we computed sense precision as the number of noun phrases matching ≥ 1 Freebase concept divided by 100, the total number of noun phrases. Sense recall is the reciprocal of the average number of Freebase concepts per noun phrase. Noun phrases matching 0 Freebase concepts were not included in this computation.

Category	Precision	Recall	Freebase concepts per Phrase
athlete	0.95	0.56	1.76
city	0.97	0.25	3.86
coach	0.86	0.94	1.06
company	0.85	0.41	2.41
country	0.74	0.56	1.77
sportsteam	0.89	0.30	3.28
stadiumoreventvenue	0.83	0.61	1.63

Table 1: ConceptResolver's word sense induction performance

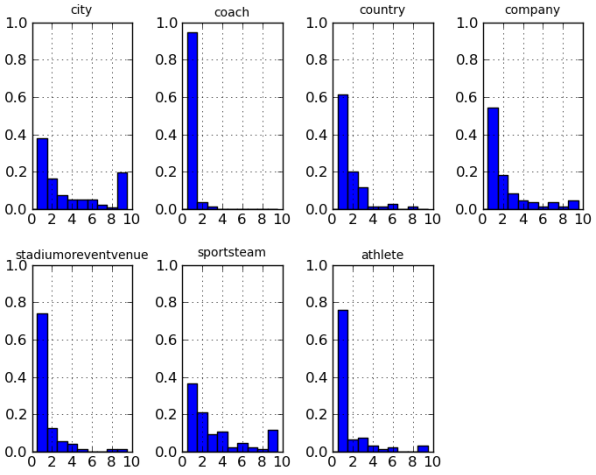

Figure 5: Empirical distribution of the number of Freebase concepts per noun phrase in each category

The results of the evaluation in Table 1 show that ConceptResolver's word sense induction works quite well for many categories. Most categories have high precision, while recall varies by category. Categories like **coach** are relatively unambiguous, with almost exactly 1 sense per noun phrase. Other categories have almost 4 senses per noun phrase. However, this average is somewhat misleading. Figure 5 shows the distribution of the number of concepts per noun phrase in each category. The distribution shows that most noun phrases are unambiguous, but a small number of noun phrases have a large number of senses. In many cases, these noun phrases

[2]In Freebase, concepts are called Topics and categories are called Types. For clarity, we use our terminology throughout.

are generic terms for many items in the category; for example, "palace" in **stadiumoreventvenue** refers to 10 Freebase concepts. Freebase's category definitions are also overly technical in some cases – for example, Freebase's version of **company** has a concept for each registered corporation. This definition means that some companies like *Volkswagen* have more than one concept (in this case, 9 concepts). These results suggest that the one-sense-per-category assumption holds for most noun phrases.

An important footnote to this evaluation is that the categories in NELL's ontology are somewhat arbitrary, and that creating subcategories would improve sense recall. For example, we could define subcategories of **sportsteam** for various sports (e.g., **football team**); these new categories would allow ConceptResolver to distinguish between teams with the same name that play different sports. Creating subcategories could improve performance in categories with a high level of polysemy.

5.2 Synonym Resolution Evaluation

Our second experiment evaluates synonym resolution by comparing the concepts created by ConceptResolver to a gold standard set of concepts. Although this experiment is mainly designed to evaluate ConceptResolver's ability to detect synonyms, it is somewhat affected by the word sense induction process. Specifically, the gold standard clustering contains noun phrases that refer to multiple concepts within the same category. (It is unclear how to create a gold standard clustering without allowing such mappings.) The word sense induction process produces only one of these mappings, which limits maximum possible recall in this experiment.

For this experiment, we report two different measures of clustering performance. The first measure is the precision and recall of pairwise synonym decisions, typically known as cluster precision and recall. We dub this the clustering metric. We also adopt the precision/recall measure from Resolver (Yates and Etzioni, 2007), which we dub the Resolver metric. The Resolver metric aligns each proposed cluster containing $\geq$ 2 senses with a gold standard cluster (i.e., a real-world concept) by selecting the cluster that a plurality of the senses in the proposed cluster refer to. Precision is then the fraction of senses in the proposed cluster which are also in the gold standard cluster; recall is computed analogously by swapping the roles of the proposed and gold standard clusters. Resolver precision can be interpreted as the probability that a randomly sampled sense (in a cluster with at least 2 senses) is in a cluster representing its true meaning. Incorrect senses were removed from the data set before evaluating precision; however, these senses may still affect performance by influencing the clustering process.

Precision was evaluated by sampling 100 random concepts proposed by ConceptResolver, then manually scoring each concept using both of the metrics above. This process mimics aligning each sampled concept with its best possible match in a gold standard clustering, then measuring precision with respect to the gold standard.

Recall was evaluated by comparing the system's output to a manually constructed set of concepts for each category. To create this set, we randomly sampled noun phrases from each category and manually matched each noun phrase to one or more real-world entities. We then found other noun phrases which referred to each entity and created a concept for each entity with at least one unambiguous reference. This process can create multiple senses for a noun phrase, depending on the real-world entities represented in the input assertions. We only included concepts containing at least 2 senses in the test set, as singleton concepts do not contribute to either recall metric. The size of each recall test set is listed in Table 2; we created smaller test sets for categories where synonyms were harder to find. Incorrectly categorized noun phrases were not included in the gold standard as they do not correspond to any real-world entities.

Table 2 shows the performance of ConceptResolver on each evaluation category. For each category, we also report the baseline recall achieved by placing each sense in its own cluster. ConceptResolver has high precision for several of the categories. Other categories like **athlete** and **city** have somewhat lower precision. To make this difference concrete, Figure 2 (first page) shows a random sample of 10 concepts from both **company** and **athlete**. Recall varies even more widely across categories, partly because the categories have varying levels of polysemy, and partly due to differences in average concept size. The differences in average concept size are reflected in the baseline recall numbers.

Category	# of Phrases	Recall Set Size	Resolver Metric				Clustering Metric			
			Precision	Recall	F1	Baseline Recall	Precision	Recall	F1	Baseline Recall
athlete	3886	80	0.69	0.69	0.69	0.46	0.41	0.45	0.43	0.00
city	5710	50	0.66	0.52	0.58	0.42	0.30	0.10	0.15	0.00
coach	889	60	0.90	0.93	0.91	0.43	0.83	0.88	0.85	0.00
company	3553	60	0.93	0.71	0.81	0.39	0.79	0.44	0.57	0.00
country	693	60	0.98	0.50	0.66	0.30	0.94	0.15	0.26	0.00
sportsteam	2085	100	0.95	0.48	0.64	0.29	0.87	0.15	0.26	0.00
stadiumoreventvenue	1662	100	0.84	0.73	0.78	0.39	0.65	0.49	0.56	0.00

Table 2: Synonym resolution performance of ConceptResolver

We attribute the differences in precision across categories to the different relations available for each category. For example, none of the relations for **athlete** uniquely identify a single athlete, and therefore synonymy cannot be accurately represented in the relation view. Adding more relations to NELL's ontology may improve performance in these cases.

We note that the synonym resolution portion of ConceptResolver is tuned for precision, and that perfect recall is not necessarily attainable. Many word senses participate in only one relation, which may not provide enough evidence to detect synonymy. As NELL continually extracts more knowledge, it is reasonable for ConceptResolver to abstain from these decisions until more evidence is available.

6 Discussion

In order for information extraction systems to accurately represent knowledge, they must represent noun phrases, concepts, and the many-to-many mapping from noun phrases to concepts they denote. We present ConceptResolver, a system which takes extracted relations between noun phrases and identifies latent concepts that the noun phrases refer to. Two lessons from ConceptResolver are that (1) ontologies aid word sense induction, as the senses of polysemous words tend to have distinct semantic types, and (2) redundant information, in the form of string similarity and extracted relations, helps train accurate synonym classifiers.

An interesting aspect of ConceptResolver is that its performance should improve as NELL's ontology and knowledge base grow in size. Defining finer-grained categories will improve performance at word sense induction, as more precise categories will contain fewer ambiguous noun phrases. Both extracting more relation instances and adding new relations to the ontology will improve synonym resolution. These scaling properties allow manual effort to be spent on high-level ontology operations, not on labeling individual instances. We are interested in observing ConceptResolver's performance as NELL's ontology and knowledge base grow.

For simplicity of exposition, we have implicitly assumed thus far that the categories in NELL's ontology are mutually exclusive. However, the ontology contains compatible categories like **male** and **politician**, where a single concept can belong to both categories. In these situations, the one-sense-per-category assumption may create too many word senses. We currently address this problem with a heuristic post-processing step: we merge all pairs of concepts that belong to compatible categories and share at least one referring noun phrase. This heuristic typically works well, however there are problems. An example of a problematic case is "obama," which NELL believes is a **male**, **female**, and **politician**. In this case, the heuristic cannot decide which "obama" (the male or female) is the politician. As such cases are fairly rare, we have not developed a more sophisticated solution to this problem.

ConceptResolver has been integrated into NELL's continual learning process. NELL's current set of concepts can be viewed through the knowledge base browser on NELL's website, `http://rtw.ml.cmu.edu`.

Acknowledgments

This work is supported in part by DARPA (under contract numbers FA8750-08-1-0009 and AF8750-09-C-0179) and by Google. We also gratefully acknowledge the contributions of our colleagues on the NELL project, Jamie Callan for the ClueWeb09 web crawl and Yahoo! for use of their M45 computing cluster. Finally, we thank the anonymous reviewers for their helpful comments.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 7–12.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676.

Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68.

Indrajit Bhattacharya and Lise Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 47–58.

Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1).

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration*, pages 73–78, August.

Ivan P. Fellegi and Alan B. Sunter. 1969. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210.

Lise Getoor and Christopher P. Diehl. 2005. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7:3–12.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, June.

Abraham Kaplan. 1955. An experimental study of ambiguity and context. *Mechanical Translation*, 2:39–46.

Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. 2002. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 307–314.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics - Volume 1*, pages 1–7.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems 18*.

Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Alvaro Monge and Charles Elkan. 1996. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 640–649.

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

Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence - Volume 1*, pages 913–918.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 650–659.

Pradeep Ravikumar and William W. Cohen. 2004. A hierarchical graphical model for record linkage. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 454–461.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on BIOCOMPUTING 2003*, pages 451–462.

Parag Singla and Pedro Domingos. 2006. Entity resolution with markov logic. In *Proceedings of the Sixth International Conference on Data Mining*, pages 572–582.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Morristown, NJ, USA.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to merge word senses. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1005–1014, June.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of the Seventh IEEE International Conference on Data Mining*, pages 342–350.

William E. Winkler. 1999. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. 2003. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 17*, pages 505–512.

Alexander Yates and Oren Etzioni. 2007. Unsupervised resolution of objects and relations on the web. In *Proceedings of the 2007 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.