Constraints based Taxonomic Relation Classification

Quang Xuan Do **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{quangdo2,danr}@illinois.edu

Abstract

Determining whether two terms in text have an ancestor relation (e.g. *Toyota* and *car*) or a sibling relation (e.g. *Toyota* and *Honda*) is an essential component of textual inference in NLP applications such as Question Answering, Summarization, and Recognizing Textual Entailment. Significant work has been done on developing stationary knowledge sources that could potentially support these tasks, but these resources often suffer from low coverage, noise, and are inflexible when needed to support terms that are not identical to those placed in them, making their use as general purpose background knowledge resources difficult. In this paper, rather than building a stationary hierarchical structure of terms and relations, we describe a system that, given two terms, determines the taxonomic relation between them using a machine learning-based approach that makes use of existing resources. Moreover, we develop a global constraint optimization inference process and use it to leverage an existing knowledge base also to enforce relational constraints among terms and thus improve the classifier predictions. Our experimental evaluation shows that our approach significantly outperforms other systems built upon existing well-known knowledge sources.

1 Introduction

Taxonomic relations that are read off of structured ontological knowledge bases have been shown to play important roles in many computational linguistics tasks, such as document clustering (Hotho et al., 2003), navigating text databases (Chakrabarti et al., 1997), Question Answering (QA) (Saxena et al., 2007) and summarization (Vikas et al., 2008). It is clear that the recognition of taxonomic relation between terms in sentences is essential to support textual inference tasks such as Recognizing Textual Entailment (RTE) (Dagan et al., 2006). For example, it may be important to know that a *blue Toyota* is neither a *red Toyota* nor a *blue Honda*, but that all are cars, and even Japanese cars. Work in Textual Entailment has argued quite convincingly (MacCartney and Manning, 2008; MacCartney and Manning, 2009) that many such textual inferences are largely compositional and depend on the ability to recognize some basic taxonomic relations such as the ancestor or sibling relations between terms. To date, these taxonomic relations can be read off manually generated ontologies such as Wordnet that explicitly represent these, and there has also been some work trying to extend the manually built resources using automatic acquisition methods resulting in structured knowledge bases such as the Extended WordNet (Snow et al., 2006) and the YAGO ontology (Suchanek et al., 2007).

However, identifying when these relations hold using fixed stationary hierarchical structures may be impaired by noise in the resource and by uncertainty in mapping targeted terms to concepts in the structures. In addition, for knowledge sources derived using bootstrapping algorithms and distributional semantic models such as (Pantel and Pennacchiotti, 2006; Kozareva et al., 2008; Baroni and Lenci, 2010), there is typically a trade-off between precision and recall, resulting either in a relatively accurate resource with low coverage or a noisy re-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1099–1109,
MIT, Massachusetts, USA, 9-11 October 2010.

source with broader coverage. In the current work, we take a different approach, identifying directly whether a pair of terms hold a taxonomic relation.

Fixed resources, as we observe, are inflexible when dealing with targeted terms not being covered. This often happens when targeted terms have the same meaning, but different surface forms, than the terms used in the resources (e.g. *Toyota Camry* and *Camry*). We argue that it is essential to have a classifier that, given two terms, can build a semantic representation of the terms and determines the taxonomic relations between them. This classifier will make use of existing knowledge bases in multiple ways, but will provide significantly larger coverage and more precise results. We make use of a dynamic resource such as Wikipedia to guarantee increased coverage without changing our model and also perform normalization-to-Wikipedia to find appropriate Wikipedia replacements for outside-Wikipedia terms. Moreover, stationary resources are usually brittle because of the way most of them are built: using local relational patterns (e.g. (Hearst, 1992; Snow et al., 2005)). Infrequent terms are less likely to be covered, and some relations may not be supported well by these methods because their corresponding terms rarely appear in close proximity (e.g., an Israeli tennis player *Dudi Sela* and *Roger Federrer*). Our approach uses search techniques to gather relevant Wikipedia pages of input terms and performs a learning-based classification w.r.t. to the features extracted from these pages as a way to get around this brittleness.

Motivated by the needs of NLP applications such as RTE, QA, Summarization, and the compositionality argument alluded to above, we focus on identifying two fundamental types of taxonomic relations - *ancestor* and *sibling*. An ancestor relation and its directionality can help us infer that a statement with respect to the child (e.g. *cannabis*) holds for an ancestor (e.g. *drugs*) as in the following example, taken from a textual entailment challenge dataset:

> **T**: Nigeria's NDLEA has seized 80 metric tonnes of *cannabis* in one of its largest ever hauls, officials say.
>
> **H**: Nigeria seizes 80 tonnes of *drugs*.

Similarly, it is important to know of a sibling relation to infer that a statement about *Taiwan* may (without additional information) contradict a similar statement with respect to *Japan* since these are different countries, as in the following:

> **T**: A strong earthquake struck off the southern tip of *Taiwan* at 12:26 UTC, triggering a warning from Japan's Meteorological Agency that a 3.3 foot tsunami could be heading towards Basco, in the Philippines.
>
> **H**: An earthquake strikes *Japan*.

Several recent TE studies (Abad et al., 2010; Sammons et al., 2010) suggest to isolate TE phenomena, such as recognizing taxonomic relations, and study them separately; they discuss some of characteristics of phenomena such as contradiction from a similar perspective to ours, but do not provide a solution.

In this paper, we present **TA**xonomic **RE**lation **C**lassifier (TAREC), a system that classifies taxonomic relations between a given pair of terms using a machine learning based classifier. An integral part of TAREC is also our inference model that makes use of relational constraints to enforce coherency among several related predictions. TAREC does not aim at building or extracting a hierarchical structure of concepts and relations, but rather to directly recognize taxonomic relations given a pair of terms. Target terms are represented using vector of features that are extracted from retrieved corresponding Wikipedia pages. In addition, we make use of existing stationary ontologies to find related terms to the target terms, and classify those too. This allows us to make use of a constraint-based inference model (following (Roth and Yih, 2004; Roth and Yih, 2007) that enforces coherency of decisions across related pairs (e.g., if x is-a y and y is-a z, it cannot be that x is a sibling of z).

In the rest of the paper, after discussing related work in Section 2, we present an overview of TAREC in Section 3. The learning component and the inference model of TAREC are described in Sections 4 and 5. We experimentally evaluate TAREC in Section 6 and conclude our paper in Section 7.

2 Related Work

There are several works that aim at building taxonomies and ontologies which organize concepts and their taxonomic relations into hierarchical structures. (Snow et al., 2005; Snow et al., 2006) con-

structed classifiers to identify hypernym relationship between terms from dependency trees of large corpora. Terms with recognized hypernym relation are extracted and incorporated into a man-made lexical database, WordNet (Fellbaum, 1998), resulting in the *extended WordNet*, which has been augmented with over $400,000$ synsets. (Ponzetto and Strube, 2007) and (Suchanek et al., 2007) both mined Wikipedia to construct hierarchical structures of concepts and relations. While the former exploited Wikipedia category system as a conceptual network and extracted a taxonomy consisting of subsumption relations, the latter presented the YAGO ontology, which was automatically constructed by mining and combining Wikipedia and WordNet. A natural way to use these hierarchical structures to support taxonomic relation classification is to map targeted terms onto the hierarchies and check if they subsume each other or share a common subsumer. However, this approach is limited because constructed hierarchies may suffer from noise and require exact mapping (Section 6). TAREC overcomes these limitations by searching and selecting the top relevant articles in Wikipedia for each input term; taxonomic relations are then recognized based on the features extracted from these articles.

On the other hand, information extraction bootstrapping algorithms, such as (Pantel and Pennacchiotti, 2006; Kozareva et al., 2008), automatically harvest related terms on large corpora by starting with a few seeds of pre-specified relations (e.g. *is-a*, *part-of*). Bootstrapping algorithms rely on some scoring function to assess the quality of terms and additional patterns extracted during bootstrapping iterations. Similarly, but with a different focus, Open IE, (Banko and Etzioni, 2008; Davidov and Rappoport, 2008), deals with a large number of relations which are not pre-specified. Either way, the output of these algorithms is usually limited to a small number of high-quality terms while sacrificing coverage (or vice versa). Moreover, an Open IE system cannot control the extracted relations and this is essential when identifying taxonomic relations. Recently, (Baroni and Lenci, 2010) described a general framework of distributional semantic models that extracts significant contexts of given terms from large corpora. Consequently, a term can be represented by a vector of contexts in which it frequently appears. Any vector space model could then use the terms' vectors to cluster terms into categories. Sibling terms (e.g. *Honda*, *Toyota*), therefore, have very high chance to be clustered together. Nevertheless, this approach cannot recognize ancestor relations. In this paper, we compare TAREC with this framework only on recognizing sibling vs. no relation, in a strict experimental setting which pre-specifies the categories to which the terms belong.

3 An Overview of the TAREC Algorithm

3.1 Preliminaries

In the TAREC algorithm, a *term* refers to any mention in text, such as *mountain*, *George W. Bush*, *battle of Normandy*. TAREC does not aim at extracting terms and building a stationary hierarchical structure of terms, but rather recognize the taxonomic relation between any two given terms. TAREC focuses on classifying two fundamental types of taxonomic relations: *ancestor* and *sibling*. Determining whether two terms hold a taxonomic relation depends on a pragmatic decision of how far one wants to climb up a taxonomy to find a common subsumer. For example, *George W. Bush* is a child of *Presidents of the United States* as well as *people*, even more, that term could also be considered as a child of *mammals* or *organisms* w.r.t. the Wikipedia category system; in that sense, *George W. Bush* may be considered as a sibling of *oak* because they have *organisms* as a least common subsumer. TAREC makes use of a hierarchical structure as background knowledge and considers two terms to hold a taxonomic relation only if the relation can be recognized from information acquired by climbing up at most K levels from the representation of the target terms in the structure. It is also possible that the sibling relation can be recognized by clustering terms together by using vector space models. If so, two terms are siblings if they belong to the same cluster.

To cast the problem of identifying taxonomic relations between two terms x and y in a machine learning perspective, we model it as a multi-class classification problem. Table 1 defines four relations with some examples in our experiment data sets.

This paper focuses on studying a fundamental problem of recognizing taxonomic relations (given well-segmented terms) and leaves the orthogonal is-

Relation	Meaning	Examples	
		Term x	Term y
$x \leftarrow y$	x is an ancestor of y	actor food	Mel Gibson rice
$x \rightarrow y$	x is a child of y	Makalu Monopoly	mountain game
$x \leftrightarrow y$	x and y are siblings	Paris copper	London oxygen
$x \nleftrightarrow y$	x and y have no relation	Roja egg	C++ Vega

Table 1: Taxonomic relations and some examples in our data sets.

TAxonomic RElation Classifier (TAREC)

INPUT: A pair of terms (x, y)
A learned local classifier $\mathcal{R}$ (Sec. 4)
Wikipedia $\mathcal{W}$
OUTPUT: Taxonomic relation r^* between x and y
1. $(x, y) \leftarrow \text{NormalizeToWikipedia}(x, y, \mathcal{W})$
2. $\mathcal{Z} \leftarrow \text{GetAddionalTerms}(x, y)$ (Sec. 5.2)
3. $r^* = \text{ClassifyAndInference}(x, y, \mathcal{Z}, \mathcal{R}, \mathcal{W})$ (Sec. 5.1)
RETURN: r^*;

Figure 1: The TAREC algorithm.

sues of how to take contexts into account and how it should be used in applications to a future work.

3.2 The Overview of TAREC

Assume that we already have a learned local classifier that can classify taxonomic relations between any two terms. Given two terms, TAREC uses Wikipedia and the local classifier in an inference model to make a final prediction on the taxonomic relation between these two. To motivate the need for an inference model, beyond the local classifier itself, we observe that the presence of other terms in addition to the two input terms, can provide some natural constraints on the possible taxonomic relations and thus can be used to make the final prediction (which we also refer as global prediction) more coherent. In practice, we first train a local classifier (Section 4), then incorporate it into an inference model (Section 5) to classify taxonomic relations between terms. The TAREC algorithm consists of three steps and is summarized in Figure 1 and explained below.

1. Normalizing input terms to Wikipedia: Although most commonly used terms have corresponding Wikipedia articles, there are still a lot of terms with no corresponding Wikipedia articles. For a non-Wikipedia term, we make an attempt to find a replacement by using Web search. We wish to find a replacement such that the taxonomic relation is unchanged. For example, for input pair (*Lojze Kovačič*, *Rudi Šeligo*), there is no English Wikipedia page for *Lojze Kovačič*, but if we can find *Marjan Rožanc* and use it as a replacement of *Lojze Kovačič* (two terms are siblings and refer to two writers), we can continue classifying the taxonomic relation of the pair (*Marjan Rožanc*, *Rudi Šeligo*). This part of the algorithm was motivated by (Sarmento et al., 2007). We first make a query with the two input terms (e.g. "*Lojze Kovačič*" AND "*Rudi Šeligo*") to search for list-structure snippets in Web documents[1] such as "... $\langle delimiter \rangle$ c_a $\langle delimiter \rangle$ c_b $\langle delimiter \rangle$ c_c $\langle delimiter \rangle$ ..." (the two input terms should be among c_a, c_b, c_c, ...). The delimiter could be commas, periods, or asterisks[2]. For snippets that contain the patterns of interest, we extract c_a, c_b, c_c etc. as replacement candidates. To reduce noise, we empirically constrain the list to contain at least 4 terms that are no longer than 20 characters each. The candidates are ranked based on their occurrence frequency. The top candidate with Wikipedia pages is used as a replacement.

2. Getting additional terms (Section 5.2): TAREC leverage an existing knowledge base to extract additional terms related to the input terms, to be used in the inference model in step 3.

3. Making global prediction with relational constraints (Section 5.1): TAREC performs several local predictions using the local classifier $\mathcal{R}$ (Section 4) on the two input terms and these terms with the additional ones. The global prediction is then inferred by enforcing relational constraints among the terms' relations.

4 Learning Taxonomic Relations

The local classifier of TAREC is trained on the pairs of terms with correct taxonomic relation labels (some examples are showed in Table 1). The trained classifier when applied on a new input pair of terms will return a real valued number which can be interpreted as the probability of the predicted label. In this section, we describe the learning features used

[1] We use http://developer.yahoo.com/search/web/

[2] Periods and asterisks capture enumerations.

Title/Term	Text	Categories
President of the United States	*The President of the United States is the head of state and head of government of the United States and is the highest political official in the United States by influence and recognition. The President leads the executive branch of the federal government and is one of only two elected members of the executive branch...*	*Presidents of the United States, Presidency of the United States*
George W. Bush	*George Walker Bush; born July 6, 1946) served as the 43rd President of the United States from 2001 to 2009. He was the 46th Governor of Texas from 1995 to 2000 before being sworn in as President on January 20, 2001...*	*Children of Presidents of the United States, Governors of Texas, Presidents of the United States, Texas Republicans...*
Gerald Ford	*Gerald Rudolff Ford (born Leslie Lynch King, Jr.) (July 14, 1913 December 26, 2006) was the 38th President of the United States, serving from 1974 to 1977, and the 40th Vice President of the United States serving from 1973 to 1974.*	*Presidents of the United States, Vice Presidents of the United States, Republican Party (United States) presidential nominees...*

Table 2: Examples of texts and categories of Wikipedia articles.

by our local taxonomic relation classifier.

Given two input terms, we first build a semantic representation for each term by using a local search engine[3] to retrieve a list of top articles in Wikipedia that are relevant to the term. To do this, we use the following procedure: (1) Using both terms to make a query (e.g. *"George W. Bush"* AND *"Bill Clinton"*) to search in Wikipedia ; (2) Extracting important keywords in the titles and categories of the retrieved articles using TF-IDF (e.g. *president*, *politician*); (3) Combining each input term with the extracted keywords (e.g. *"George W. Bush"* AND *"president"* AND *"politician"*) to create a final query used to search for the term's relevant articles in Wikipedia. This is motivated by the assumption that the real world applications calling TAREC typically does so with two terms that are related in some sense, so our procedure is designed to exploit that. For example, it's more likely that term *Ford* in the pair (*George W. Bush*, *Ford*) refers to the former president of the United States, *Gerald Ford*, than the founder of Ford Motor Company, *Henry Ford*.

Once we have a semantic representation of each term, in the form of the extracted articles, we extract from it features that we use as the representation of the two input terms in our learning algorithm. It is worth noting that a Wikipedia page usually consists of a title (i.e. the term), a body text, and a list of categories to which the page belongs. Table 2 shows some Wikipedia articles. From now on, we use *the titles of* x, *the texts of* x, and *the categories of* x to refer to the titles, texts, and categories of the associated articles in the representation of x. Below are the learning features extracted for input pair (x,y).

Bags-of-words Similarity: We use cosine similarity metric to measure the degree of similarity between bags of words. We define four bags-of-words features as the degree of similarity between the texts

Degree of similarity
texts(x) vs. categories(y)
categories(x) vs. texts(y)
texts(x) vs. texts(y)
categories(x) vs. categories(y)

Table 3: Bag-of-word features of the pair of terms (x,y); texts(.) and categories(.) are two functions that extract associated texts and categories from the semantic representation of x and y.

and categories associated with two input terms x and y in Table 3. To collect categories of a term, we take the categories of its associated articles and go up K levels in the Wikipedia category system. In our experiments, we use abstracts of Wikipedia articles instead of whole texts.

Association Information: This features represents a measure of association between the terms by considering their information overlap. We capture this feature by the pointwise mutual information (pmi) which quantifies the discrepancy between the probability of two terms appearing together versus the probability of each term appearing independently[4]. The pmi of two terms x and y is estimated as follows:

$$pmi(x,y) = log\frac{p(x,y)}{p(x)p(y)} = log\frac{Nf(x,y)}{f(x)f(y)},$$

where N is the total number of Wikipedia articles, and $f(.)$ is the function which counts the number of appearances of its argument.

Overlap Ratios: The overlap ratio features capture the fact that the titles of a term usually overlap with the categories of its descendants. We measure this overlap as the ratio of the number of common phrases used in the titles of one term and the categories of the other term. In our context, a phrase is

[3]E.g. http://lucene.apache.org/

[4]pmi is different than mutual information. The former applies to specific outcomes, while the latter is to measure the mutual dependence of two random variables.

considered to be a common phrase if it appears in the titles of one term and the categories of the other term and it is also of the following types: (1) the whole string of a category, or (2) the head in the root form of a category, or (3) the post-modifier of a category. We use the Noun Group Parser from (Suchanek et al., 2007) to extract the head and post-modifier from a category. For example, one of the categories of an article about *Chicago* is *Cities in Illinois*. This category can be parsed into a head in its root form *City*, and a post-modifier *Illinois*. Given term pair *(City, Chicago)*, we observe that *City* matches the head of the category *Cities in Illinois* of term *Chicago*. This is a strong indication that *Chicago* is a child of *City*.

We also use a feature that captures the overlap ratio of *common phrases* between the categories of two input terms. For this feature, we do not use the *post-modifier* of the categories. We use Jaccard similarity coefficient to measure these overlaps ratios.

5 Inference with Relational Constraints

Once we have a local multi-class classifier that maps a given pair of terms to one of the four possible relations, we use a constraint-based optimization algorithm to improve this prediction. The key insight behind the way we model the inference model is that if we consider more than two terms, there are logical constraints that restrict the possible relations among them. For instance, *George W. Bush* cannot be an ancestor or sibling of *president* if we are confident that *president* is an ancestor of *Bill Clinton*, and *Bill Clinton* is a sibling of *George W. Bush*. We call the combination of terms and their relations a *term network*. Figure 2 shows some n-term networks consisting of two input terms (x, y), and additional terms z, w, v.

The aforementioned observations show that if we can obtain additional terms that are related to the two target terms, we can enforce such coherency relational constraints and make a global prediction that would improve the prediction of the taxonomic relation between the two given terms. Our inference model follows constraint-based formulations that were introduced in the NLP community and were shown to be very effective in exploiting declarative background knowledge (Roth and Yih, 2004; Denis and Baldridge, 2007; Punyakanok et al., 2008; Chang et al., 2008).

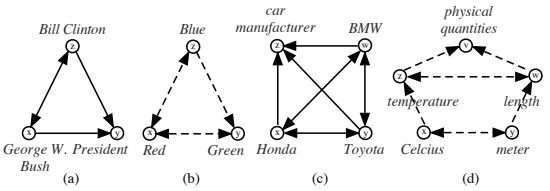

Figure 2: Examples of n-term networks with two input term x and y. (a) and (c) show valid combinations of edges, whereas (b) and (d) are two relational constraints. For simplicity, we do not draw *no relation* edges in (d).

5.1 Enforcing Coherency through Inference

Let x, y be two input terms, and $\mathcal{Z} = \{z_1, z_2, ..., z_m\}$ be a set of additional terms. For a subset $Z \in \mathcal{Z}$, we construct a set of term networks whose nodes are x, y and all elements in Z, and the edge, e, between every two nodes is one of four taxonomic relations whose weight, $w(e)$, is given by a local classifier (Section 4). If $l = |Z|$, there are $n = 2 + l$ nodes in each network, and $4^{\left[\frac{1}{2}n(n-1)\right]}$ term networks can be constructed. In our experiments we only use 3-term networks (i.e. $l = 1$). For example, for the input pair (*red*, *green*) and $\mathcal{Z} = \{blue, yellow\}$, we can construct 64 networks for the triple $\langle red, green, Z = \{blue\}\rangle$ and 64 networks for $\langle red, green, Z = \{yellow\}\rangle$ by trying all possible relations between the terms.

A *relational constraint* is defined as a term network consisting of only its "illegitimate" edge settings, those that belongs to a pre-defined list of invalid edge combinations. For example, Figure 2b shows an invalid network where *red* is a sibling of both *green* and *blue*, and *green* is an ancestor of *blue*. In Figure 2d, *Celcius* and *meter* cannot be siblings because they are children of two sibling terms *temperature* and *length*. The relational constraints used in our experiments are manually constructed.

Let $\mathcal{C}$ be a list of relational constraints. Equation (1) defines the network scoring function, which is a linear combination of the edge weights, $w(e)$, and the penalties, ρ_k, of term networks matching constraint $C_k \in \mathcal{C}$.

$$score(t) = \sum_{e \in t} w(e) - \sum_{k=1}^{|\mathcal{C}|} \rho_k d_{C_k}(t) \qquad (1)$$

function $d_{C_k}(t)$ indicates if t matches C_k. In our work, we use relational constraints as hard con-

YAGO Query Patterns

INPUT: term "x"

OUTPUT: lists of ancestors, siblings, and children of "x"

Pattern 1	Pattern 2	Pattern 3
"x" MEANS ?A	"x" MEANS ?A	"x" MEANS ?D
?A SUBCLASSOF ?B	?A TYPE ?B	?E TYPE ?D
?C SUBCLASSOF ?B	?C TYPE ?B	

RETURN: ?B, ?C, ?E as lists of ancestors, siblings, and children, respectively.

Figure 3: Our YAGO query patterns used to obtain related terms for "x".

straints and set their penalty ρ_k to ∞. For a set of term networks formed by $\langle x, y, Z \rangle$ and all possible relations between the terms, we select the best network, $t^* = \text{argmax}_t score(t)$.

After picking the best term network t^* for every $Z \in \mathcal{Z}$, we make the final decision on the taxonomic relation between x and y. Let r denote the relation between x and y in a particular t^* (e.g. $r = x \leftrightarrow y$.) The set of all t^* is divided into 4 groups with respect to r (e.g. a group of all t^* having $r = x \leftrightarrow y$, a group of all t^* having $r = x \leftarrow y$.) We denote a group with term networks holding r as the relation between x and y by $\mathcal{T}_r$. To choose the best taxonomic relation, r^*, of x and y, we solve the objective function defined in Equation 2.

$$r^* = \text{argmax}_r \frac{1}{|\mathcal{T}_r|} \sum_{t^* \in \mathcal{T}_r} \lambda_{t^*} score(t^*) \qquad (2)$$

where λ_t is the weight of term network t, defined as the occurrence probability of t (regarding only its edges' setting) in the training data, which is augmented with additional terms. Equation (2) finds the best taxonomic relation of two input terms by computing the average score of every group of the best term networks representing a particular relation of two input terms.

5.2 Extracting Related Terms

In the inference model, we need to obtain other terms that are related to the two input terms. Hereafter, we refer to additional terms as *related terms*. The related term space is a space of direct ancestors, siblings and children in a particular resource.

We propose an approach that uses the YAGO ontology (Suchanek et al., 2007) to provide related terms. It is worth noting that YAGO is chosen over the Wikipedia category system used in our work because YAGO is a clean ontology built by carefully combining Wikipedia and WordNet.[5]

In YAGO model, all objects (e.g. *cities*, *people*, etc.) are represented as *entities*. To map our input terms to entities in YAGO, we use the MEANS relation defined in the YAGO ontology. Furthermore, similar entities are grouped into *classes*. This allows us to obtain direct ancestors of an entity by using the TYPE relation which gives the entity's classes. Furthermore, we can get ancestors of a class with the SUBCLASSOF relation[6]. By using three relations MEANS, TYPE and SUBCLASSOF in YAGO model, we can obtain Proposals for direct ancestors, siblings, and children, if any, for any input term. We then evaluate our classifier on all pairs, and run the inference to improve the prediction using the coherency constraints. Figure 3 presents three patterns that we used to query related terms from YAGO.

6 Experimental Study

In this section, we evaluate TAREC against several systems built upon existing well-known knowledge sources. The resources are either hierarchical structures or extracted by using distributional semantic models. We also perform several experimental analyses to understand TAREC's behavior in details.

6.1 Comparison to Hierarchical Structures

We create and use two main data sets in our experiments. **Dataset-I** is generated from 40 semantic classes of about 11,000 instances. The original semantic classes and instances were manually constructed with a limited amount of manual post-filtering and were used to evaluate information extraction tasks in (Paşca, 2007; Paşca and Van Durme, 2008) (we refer to this original data as **OrgData-I**). This dataset contains both terms with Wikipedia pages (e.g. *George W. Bush*) and non-Wikipedia terms (e.g. *hindu mysticism*). Pairs of terms are generated by randomly pairing semantic class names and instances. We generate disjoint training and test sets of 8,000 and 12,000 pairs of terms, respectively. We call the test set of this

[5]However, YAGO by itself is weaker than our approach in identifying taxonomic relations (see Section 6.)

[6]These relations are defined in the YAGO ontology.

dataset **Test-I**. **Dataset-II** is generated from 44 semantic classes of more than 10,000 instances used in (Vyas and Pantel, 2009)[7]. The original semantic classes and instances were extracted from Wikipedia lists. This data, therefore, only contains terms with corresponding Wikipedia pages. We also generate disjoint training and test sets of 8,000 and 12,000 pairs of terms, respectively, and call the test set of this dataset **Test-II**.[8]

Several semantic class names in the original data are written in short forms (e.g. *chemicalelem*, *proglanguage*). We expand these names to some meaningful names which are used by all systems in our experiments. For example, *terroristgroup* is expanded to *terrorist group*, *terrorism*. Table 1 shows some pairs of terms which are generated. Four types of taxonomic relations are covered with balanced numbers of examples in all data sets. To evaluate our systems, we use a snapshot of Wikipedia from July, 2008. After cleaning and removing articles without categories (except redirect pages), 5,503,763 articles remain. We index these articles using Lucene[9]. As a learning algorithm, we use a regularized averaged Perceptron (Freund and Schapire, 1999).

We compare TAREC with three systems that we built using recently developed large-scale hierarchical structures. **Strube07** is built on the latest version of a taxonomy, T_{Strube}, which was derived from Wikipedia (Ponzetto and Strube, 2007). It is worth noting that the structure of T_{Strube} is similar to the page structure of Wikipedia. For a fair comparison, we first generate a semantic representation for each input term by following the same procedure used in TAREC described in Section 4. The titles and categories of the articles in the representation of each input term are then extracted. Only titles and their corresponding categories that are in T_{Strube} are considered. A term is an ancestor of the other if at least one of its titles is in the categories of the other term. If two terms share a common category, they are considered siblings; and no relation, otherwise. The ancestor relation is checked first, then sibling, and finally no relation. **Snow06** uses the *extended WordNet* (Snow et al., 2006). Words in the extended WordNet can be common nouns or proper nouns. Given two input terms, we first map them onto the hierarchical structure of the extended WordNet by exact string matching. A term is an ancestor of the other if it can be found as an hypernym after going up K levels in the hierarchy from the other term. If two terms share a common subsumer within some levels, then they are considered as siblings. Otherwise, there is no relation between the two input terms. Similar to Strube07, we first check ancestor, then sibling, and finally no relation. **Yago07** uses the YAGO ontology (Suchanek et al., 2007) as its main source of background knowledge. Because the YAGO ontology is a combination of Wikipedia and WordNet, this system is expected to perform well at recognizing taxonomic relations. To access a term's ancestors and siblings, we use patterns 1 and 2 in Figure 3 to map a term to the ontology and move up on the ontology. The relation identification process is then similar to those of Snow06 and Strube07. If an input term is not recognized by these systems, they return *no relation*.

	Test-I	**Test-II**
Strube07	24.32	25.63
Snow06	41.97	36.26
Yago07	65.93	70.63
TAREC (local)	81.89	84.7
TAREC	**85.34**	**86.98**

Table 4: Evaluating and comparing performances, in accuracy, of the systems on **Test-I** and **Test-II**. TAREC (local) uses only our local classifier to identify taxonomic relations by choosing the relation with highest confidence.

Our overall algorithm, **TAREC**, is described in Figure 1. We manually construct a pre-defined list of 35 relational constraints to use in the inference model. We also evaluate our local classifier (Section 4), which is referred as **TAREC (local)**. To make classification decision with TAREC (local), for a pair of terms, we choose the predicted relation with highest confidence returned by the classifier.

In all systems compared, we vary the value of K[10] from 1 to 4. The best result of each system is reported. Table 4 shows the comparison of all systems evaluated on both Test-I and Test-II. Our systems, as shown, significantly outperform the other

[7]There were 50 semantic classes in the original dataset. We grouped some semantically similar classes for the purpose of classifying taxonomic relations.

[8]Published at http://cogcomp.cs.illinois.edu/page/software

[9]http://lucene.apache.org, version 2.3.2

[10]See Section 3.1 for the meaning of K.

systems. In Table 4, the improvement of TAREC over TAREC (local) on Test-I shows the contribution of both the normalization procedure (that is, going outside Wikipedia terms) and the global inference model to the classification decisions, whereas the improvement on Test-II shows only the contribution of the inference model, because Test-II contains only terms with corresponding Wikipedia articles.

Observing the results we see that our algorithms is doing significantly better that fixed taxonomies based algorithms. This is true both for TAREC (local) and for TAREC. We believe that our machine learning based classifier is very flexible in extracting features of the two input terms and thus in predicting their taxonomic Relation. On the other hand, other system rely heavily on string matching techniques to map input terms to their respective ontologies, and these are very inflexible and brittle. This clearly shows one limitation of using existing structured resources to classify taxonomic relations.

We do not use special tactics to handle polysemous terms. However, our procedure of building semantic representations for input terms described in Section 4 ties the senses of the two input terms and thus, implicitly, may get some sense information. We do not use this procedure in Snow06 because WordNet and Wikipedia are two different knowledge bases. We also do not use this procedure in Yago07 because in YAGO, a term is mapped onto the ontology by using the MEANS operator (in Pattern 1, Figure 3). This cannot follow our procedure.

6.2 Comparison to Harvested Knowledge

As we discussed in Section 2, the output of bootstrapping-based algorithms is usually limited to a small number of high-quality terms while sacrificing coverage (or vice versa). For example, the full Espresso algorithm in (Pantel and Pennacchiotti, 2006) extracted 69,156 instances of *is-a* relation with 36.2% precision. Similarly, (Kozareva et al., 2008) evaluated only a small number (a few hundreds) of harvested instances. Recently, (Baroni and Lenci, 2010) proposed a general framework to extract properties of input terms. Their **TypeDM** model harvested 5,000 significant properties for each term out of 20,410 noun terms. For example, the properties of *marine* include $\langle own, bomb\rangle$, $\langle use, gun\rangle$. Using vector space models we could measure the similarity between terms using their property vectors. However, since the information available in TypeDM does not support predicting the ancestor relation between terms, we only evaluate TypeDM in classifying sibling vs. no relation. We do this by giving a list of semantic classes using the following procedure: (1) For each semantic class, use some seeds to compute a centroid vector from the seeds' vectors in TypeDM, (2) each term in an input pair is classified into its best semantic class based on the cosine similarity between its vector and the centroid vector of the category, (3) two terms are siblings if they are classified into the same category; and have no relation, otherwise. Out of 20,410 noun terms in TypeDM, there are only 345 terms overlapping with the instances in OrgData-I and belonging to 10 significant semantic classes. For each semantic class, we randomly pick 5 instances as its seeds to make a centroid vector. The rest of the overlapping instances are randomly paired to make a dataset of 4,000 pairs of terms balanced in the number of sibling and no relation pairs. On this dataset, TypeDM achieves the accuracy of 79.75%. TAREC (local), with the local classifier trained on the training set (with 4 relation classes) of Dataset-I, gives 78.35% of accuracy. The full TAREC system with relational constraints achieves 82.65%. We also re-train and evaluate the local classifier of TAREC on the same training set but without ancestor relation pairs. This local classifier has an accuracy of 81.08%.

These results show that although the full TAREC system gives better performance, TypeDM is very competitive in recognizing sibling vs. no relation. However, TypeDM can only work in a limited setting where semantic classes are given in advance, which is not practical in real-world applications; and of course, TypeDM does not help to recognize ancestor relations between two terms.

6.3 Experimental Analysis

In this section, we discuss some experimental analyses to better understand our systems.

Precision and Recall: We want to study TAREC on individual taxonomic relations using Precision and Recall. Table 5 shows that TAREC performs very well on ancestor relation. Sibling and no relation are the most difficult relations to classify. In the same experimental setting on Test-I, Yago07

TAREC	Test-I		Test-II	
	Prec	Rec	Prec	Rec
$x \leftarrow y$	95.82	88.01	96.46	88.48
$x \rightarrow y$	94.61	89.29	96.15	88.86
$x \leftrightarrow y$	79.23	84.01	83.15	81.87
$x \nleftrightarrow y$	73.94	79.9	75.54	88.27
Average	85.9	85.3	87.83	86.87

Table 5: Performance of TAREC on individual taxonomic relation.

	Wiki	**WordNet**	**non-Wiki**
Strube07	24.59	24.13	21.18
Snow06	41.23	46.91	34.46
Yago07	69.95	70.42	34.26
TAREC (local)	89.37	89.72	31.22
TAREC	**91.03**	**91.2**	**45.21**

Table 6: Performance of the systems on special data sets, in accuracy. On the non-Wikipedia test set, TAREC (local) simply returns sibling relation.

achieves 79.34% and 66.03% of average Precision and Recall, respectively. These numbers on Test-II are 81.33% and 70.44%.

Special Data Sets: We evaluate all systems that use hierarchical structures as background knowledge on three special data sets derived from Test-I. From 12,000 pairs in Test-I, we created a test set, **Wiki**, consisting of $10,456$ pairs with all terms in Wikipedia. We use the rest of $1,544$ pairs with at least one non-Wikipedia term to build a **non-Wiki** test set. The third dataset, **WordNet**, contains $8,625$ pairs with all terms in WordNet and Wikipedia. Table 6 shows the performance of the systems on these data sets. Unsurprisingly, Yago07 gets better results on Wiki than on Test-I. Snow06, as expected, gives better performance on the WordNet test set. TAREC still significantly outperforms these systems. The improvement of TAREC over TAREC (local) on the Wiki and WordNet test sets shows the contribution of the inference model, whereas the improvement on the non-Wikipedia test set shows the contribution of normalizing input terms to Wikipedia.

Contribution of Related Terms in Inference: We evaluate TAREC when the inference procedure is fed by related terms that are generated using a "gold standard" source instead of YAGO. To do this, we use the original data which was used to generate Test-I. For each term in the examples of Test-I, we get its ancestors, siblings, and children, if any, from the original data and use them as related terms in the inference model. This system is referred as **TAREC (Gold Infer.)**. Table 7 shows the results of the two systems on different K as the number of levels to go up on the Wikipedia category system. We see that TAREC gets better results when doing inference with better related terms. In this experiment, the two systems use the same number of related terms.

	K=1	K=2	K=3	K=4
TAREC	82.93	85.34	85.23	83.95
TAREC (Gold Infer.)	83.46	86.18	85.9	84.93

Table 7: Evaluating TAREC with different sources providing related terms to do inference.

7 Conclusions

We studied an important component of many computational linguistics tasks: given two target terms, determine that taxonomic relation between them. We have argued that static structured knowledge bases cannot support this task well enough, and provided empirical support for this claim. We have developed TAREC, a novel algorithm that leverages information from existing knowledge sources and uses machine learning and a constraint-based inference model to mitigate the noise and the level of uncertainty inherent in these resources. Our evaluations show that TAREC significantly outperforms other systems built upon existing well-known knowledge sources. Our approach generalizes and handles non-Wikipedia term well across semantic classes. Our future work will include an evaluation of TAREC in the context of textual inference applications.

Acknowledgments

The authors thank Mark Sammons, Vivek Srikumar, James Clarke and the anonymous reviewers for their insightful comments and suggestions. University of Illinois at Urbana-Champaign gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract No. FA8750-09-C-0181. The first author also thanks the Vietnam Education Foundation (VEF) for its sponsorship. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the VEF, DARPA, AFRL, or the US government.

References

A. Abad, L. Bentivogli, I. Dagan, D. Giampiccolo, S. Mirkin, E. Pianta, and A. Stern. 2010. A resource for investigating the impact of anaphora and coreference on inference. In *LREC*.

M. Banko and O. Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL-HLT*.

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36.

S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. 1997. Using taxonomy, discriminants, and signatures for navigating in text databases. In *VLDB*.

M. Chang, L. Ratinov, and D. Roth. 2008. Constraints as prior knowledge. In *ICML Workshop on Prior Knowledge for Text and Language Processing*.

D. Davidov and A. Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *ACL*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *NAACL*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*.

M. A. Hearst. 1992. Acquisition of hyponyms from large text corpora. In *COLING*.

A. Hotho, S. Staab, and G. Stumme. 2003. Ontologies improve text document clustering. In *ICDM*.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL-HLT*.

B. MacCartney and C. D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *COLING*.

B. MacCartney and C. D. Manning. 2009. An extended model of natural logic. In *IWCS-8*.

M. Paşca and B. Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL-HLT*.

M. Paşca. 2007. Organizing and searching the world wide web of facts step two: Harnessing the wisdom of the crowds. In *WWW*.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, pages 113–120.

S. P. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from wikipedia. *AAAI*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

M. Sammons, V.G. Vydiswaran, and D. Roth. 2010. Ask not what textual entailment can do for you... In *ACL*.

L. Sarmento, V. Jijkuon, M. de Rijke, and E. Oliveira. 2007. "more like these": growing entity classes from seeds. In *CIKM*.

A. K. Saxena, G. V. Sambhu, S. Kaushik, and L. V. Subramaniam. 2007. Iitd-ibmirl system for question answering using pattern matching, semantic type and semantic category recognition. In *TREC*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*.

F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*.

O. Vikas, A. K. Meshram, G. Meena, and A. Gupta. 2008. Multiple document summarization using principal component analysis incorporating semantic vector space model. In *Computational Linguistics and Chinese Language Processing*.

V. Vyas and P. Pantel. 2009. Semi-automatic entity set refinement. In *NAACL-HLT*.

D. Yarowsky. 1995. Unsupervised woed sense disambiguation rivaling supervied methods. In *Proceedings of ACL-95*.