On-the-fly Constraint-based Taxonomic Relation Identification

Abstract

Determining whether two concepts in text have an ancestor relation or a sibling relation is often essential to support textual inferences such as identifying that *"Jim drives a Honda"* contradicts *"Jim drives a Toyota"* but both imply that *"Jim drives a Japanese car"*. Significant effort has been devoted to developing structured knowledge bases that could potentially support these tasks, but these lack sufficient coverage and, more significantly, from noise and unavoidable uncertainty, making their use in textual inference difficult. We argue that, given two concepts, the problem of determining what taxonomic relation holds between them should be addressed by fusing information from multiple knowledge sources both structured and unstructured. We present an approach that makes use of machine learning and a constrained optimization based decision algorithm to combine information from structured sources and unstructured text, and show that it significantly improves taxonomic relation identification relative to existing structured knowledge sources.

1 Introduction

Textual inference requires the use of large amounts of background knowledge. For example, it may be important to know that a *blue Toyota* is not a *red Toyota* nor a *blue Honda* but that all are cars, and even Japanese made cars. Word in Textual Entailment (Dagan et al., 2006; Haghighi et al., 2005; Braz et al., 2005), has argued quite convincingly, (e.g. (Maccartney and Manning, 2008)), that many inferences are largely compositional and depend on the ability of models to recognize taxonomic relations between noun phrases and entities. For example, it is often necessary to know of an *ancestor* relation and its directionality in order to deduce 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 the textual entailment challenge data set:

> **T**: Nigeria's National Drug Law Enforcement Agency (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 often important to know of a *sibling* relation to infer that a statement about *Taiwan* may (without additional information)*contradict* an identical statement with respect to *Japan* since these are *different* countries, as in the following example:

> **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*.

A lot of work has been devoted to acquiring semantic taxonomies and ontologies (Snow et al., 2006; Suchanek et al., 2007) resulting in structured knowledge sources such as augmented WordNet and Yago which represent taxonomic information about individual concepts (Sec. 6). However, as we show, these suffer from limited coverage and, more importantly, need to represent uncertainty with respect to each concept, making it difficult to support robust classification of taxonomic relations between two given concepts.

Identifying and classifying taxonomic relations between two given concepts serves a different purpose and is distinct from that of Open Information Extraction (Banko et al., 2007), On-Demand Information Extraction (Sekine, 2006) and other effort to recognize *easy to find* facts in a given corpus (Davidov and Rappoport, 2008; Paşca

and Van Durme, 2008)–capitalizing on local co-occurrence of concepts to generate databases of open-ended facts. It is also different from the supervised relation extraction (Roth and Yih, 2004) effort which requires additional supervised data to learn new relations.

We believe that these acquired structured knowledge bases and ontologies are important resources in order to support robust classification of taxonomic relations but, as we show, are not sufficient. There is a need to fuse information from multiple sources, including unstructured text (e.g., the web) in order to determine the taxonomic relation between a pair of concepts.

We present an approach that makes use of machine learning and a constrained optimization based decision algorithm to combine information from structured sources and unstructured text, and show that it significantly improves taxonomic relation identification relative to existing structured knowledge sources.

Our key technical contribution is a constraint-based framework that makes use of relational constraints in a global inference process that accurately identifies taxonomic relations. Our inference algorithm makes use of an accurate classifier we develop, which returns, for a given pair of concepts, a distribution over possible taxonomic relations; this classifier is applied multiple times, on an automatically generated network of concepts that are related to the target pair, and the constrained optimization technique is used to force these decisions to cohere across the network. This results in improving the accuracy of each of the predicted relations in the network. Our basic classifier makes use of Wikipedia and its category structure. On one hand, this guarantees growing coverage but, on the other, necessitates taking into account the non-uniformity and level of noise in this resource. Our algorithmic approach therefore treats Wikipedia and its category structure as an open resource and uses statistical text mining techniques to gather robust information. And, while Wikipedia has broad coverage, there is a need to go beyond it. We suggest a simple but efficient technique to accomplish this using web search, and show its effectiveness when at least one of the target concepts is not mentioned in Wikipedia.

The contributions of this paper are:

1. The development of a robust and accurate machine learning-based approach to classify taxonomic relations.
2. A constraint-based inference model that incorporates prior knowledge to accurately identify concept taxonomic relations.
3. An approach to leveraging an existing knowledge base, which by itself is weaker than our relation identifier, to provide useful information for the inference model.

In an extensive experimental study we show that our approach significantly improves taxonomic relation identification relative to existing structured knowledge sources.

The next section gives an overview of our algorithmic approach. In Sec. 3 we present the constraint-based inference model that makes used of global relational constraints to infer concepts' taxonomic relations. As shown, in order to generate a network of related concepts for the inference process we leverage the YAGO ontology. Our machine learning-based component is used to make local prediction on taxonomic relations and is described in Sec. 4. Sec. 5 presents our experimental evaluation, and Sec. 6 describes some of the related work.

2 Overview of the Algorithm

In this section, we present our overall algorithm addressing the problem of identifying taxonomic relation of concepts.

Inputs of our algorithm are a pair of target concepts (X, Y) and a trained taxonomic relation classifier $\mathcal{R}$, which is used to provide a distribution of confidences over all possible taxonomic relations between any two concepts. The algorithm's output is the taxonomic relation of X and Y, which can be (1) X is an *ancestor* of Y, denoted by $X \leftarrow Y$; (2) X is a *child* of Y, denoted by $X \rightarrow Y$; (3) X and Y are *siblings*, denoted by $X \leftrightarrow Y$; or (4) X and Y have *no relation*, denoted by $X \nleftrightarrow Y$.

Figure 1 presents our overall algorithm to identify taxonomic relation of X and Y. The algorithm consists of two main components (1) a machine learning-based algorithm to learn a local taxonomic relation classifier; and (2) a constraint-based inference model that makes use of related concepts extracted for X and Y to make final decision on taxonomic relation of X and Y. Both components use the same Wikipedia index as a knowledge source to extract features of input concepts

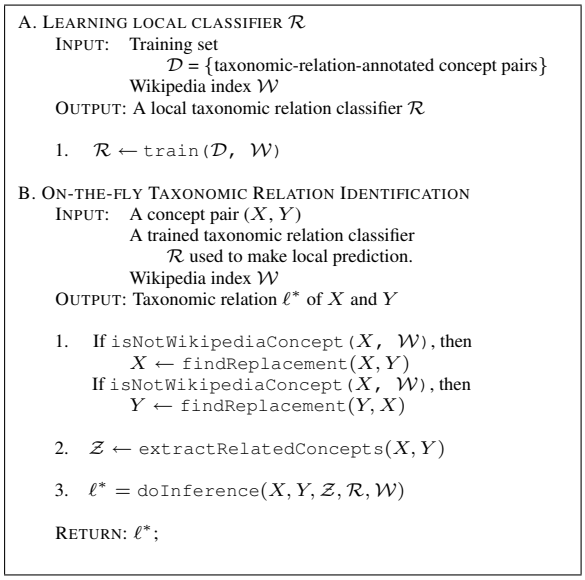

Figure 1: On-the-fly taxonomic relation identification algorithm.

and related concepts to learn the local classifier and do inference as well.

The identification component (B) makes prediction on the taxonomic relation between X and Y using $\mathcal{R}$ as its local classifier. Given two concepts X, and Y, function `isNotWikipediaConcept` determines if they are *Wikipedia concepts* or *non-Wikipedia concepts* by searching the concept space in Wikipedia. If a concept is *non-Wikipedia* (i.e. it does not have a Wikipedia page), the algorithm tries to find a replacement for it by performing a web search via function `findReplacement`. Replacing concepts are expected to be *Wikipedia concepts* and in the same semantic class with the input concepts. After that, function `extractRelatedConcepts` returns list $\mathcal{Z}$ of additional concepts which are related to X and Y. The algorithm passes the two input concepts, local classifier $\mathcal{R}$ and list $\mathcal{Z}$ to function `doInference`, which solves a constraint-based optimization problem and returns ℓ^*, the taxonomic relation of X and Y.

Functions `extractRelatedConcepts` and `doInference` are described in details in Sec. 3.

The learning component (A) trains a local classifier $\mathcal{R}$ for the taxonomic relation identification, and is described in Sec. 4.

Although most commonly used concepts can be found in Wikipedia, there is still a need to cover the *non-Wikipedia concepts* to improve the coverage of the algorithm. Briefly, function `findReplacement`(X, Y) takes a *non-Wikipedia concept* X and a supporting concept Y as its input. The function searches the web to find a *Wikipedia concept* X' which is in the same semantic class of X to be its replacement. Our method was motivated by (Sarmento et al., 2007). In our work, we use the Yahoo! Web Search APIs[1] to search for list structures in web documents such as "... $\langle delimiter \rangle$ c$_a$ $\langle delimiter \rangle$ c$_b$ $\langle delimiter \rangle$ c$_c$ $\langle delimiter \rangle$ ..." (X and Y are among c$_a$, c$_b$, c$_c$, ...). For text snippets that contain the patterns of interest, we extract c_a, c_b, etc. as replacement candidates. To reduce noise, we constrain the list structure to contain at least 4 concepts that are no longer than 20 characters each. The candidates are ranked based on their occurrence frequency. The top candidate in the Wikipedia concept space is used as replacement.

3 Inference with Relational Constraints

Analyzing concepts and the relations between them reveals several relational constraints among the relations identified for the target pair and those identified for related concepts. For instance, *George W. Bush* cannot be an ancestor or sibling of *president* if we are confident that concept *president* is an ancestor of *Bill Clinton*, and *Bill Clinton* is a sibling of *George W. Bush*. Another example would be that if concepts *red* and *green* are known to be siblings, and *blue* is also known to be a sibling of *red*, the prediction that *green* is an ancestor of *blue* should be invalid. We call the combination of concepts and their relations *concept network*. Fig. 2 shows some n-node concept networks consisting of two input concepts (x, y), and additional concepts z, w, v. Fig. 2(b) and 2(d) illustrate two *relational constraints* which will be defined later. In general, n concepts can be involved to construct n-node concept networks ($n > 2$). In this paper, we focus on 3-node concept networks consisting of the two target concepts and an additional one. However, our formalization applies to general n-node concept networks.

The aforementioned observations show that, if we can obtain additional concepts that are related to the two input concepts, we can enforce such relational constraints as prior knowledge to make final decision. Our inference model follows constraint-based formulations that were in-

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

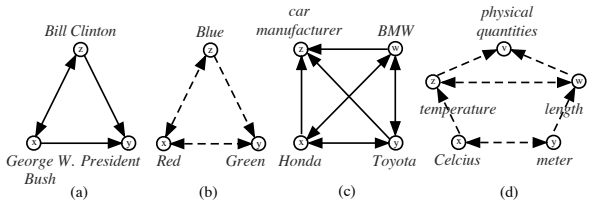

Figure 2: Examples of n-node concept networks formed by two input concepts x, y, and related concepts z, w and v. Concept networks (a) and (c) show valid combinations of edges, whereas (b) and (d) demonstrate invalid combinations. (b) and (d) are two relational constraints that we do not allow. For simplicity, we do not draw *no relation* edges in (d).

troduced in the NLP community and were shown to be very effective in exploiting declarative background knowledge as a way to achieve significant improvement in performance (Roth and Yih, 2004; Punyakanok et al., 2005b; Chang et al., 2009; Denis and Baldridge, 2007; Punyakanok et al., 2005a). While it is also possible to inject prior knowledge *indirectly*, by adding more features, it has been argued (e.g., (Roth and Yih, 2005; Chang et al., 2008)) that a *direct* way, in the form of soft or hard constraints, is beneficial due to better expressivity and simpler learning. Our model follows this approach.

3.1 Constraints as Prior Knowledge

Let x, y be the two input concepts let $\mathcal{Z} = \{z_1, z_2, ..., z_m\}$ be a set of additional concepts. For a subset $Z \in \mathcal{Z}$, we construct a set of concept 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 (see Sec. 4). If $m = |Z|$, there will be $n = m + 2$ nodes in each network, and $4^{\left[\frac{1}{2}n(n-1)\right]}$ concept networks in total. With $m = 1$, each network consists of 3 nodes, and there are 64 concept networks.

A *relational constraint* is defined as an invalid concept network (regarding only its edges' setting) that is not allowed to happen (e.g. Fig. 2(b), Fig. 2(d)).

Let $\mathcal{C}$ be a list of relational constraints. Following the approaches to inject prior knowledge directly in (Chang et al., 2008), we incorporate relational constraints into our model to choose the best network t^*, which also gives the relation between x and y, from a set of concept networks constructed from $\langle x, y, Z \rangle$. The scoring function is a linear combination of the edge weights $w(e)$ and the penalties ρ_k of concept networks violating constraint $C_k \in \mathcal{C}$.

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

In Eqn. 1, function $d_{C_k}(t)$ measures the degree that concept network t violates constraint C_k. In our work, we use relational constraints as hard constraints and set their penalty ρ_k to ∞. Objective function 1 allows us to pick the best setting of all edges connecting concepts in the networks with respect to $\mathcal{C}$. After picking the best concept network t^* for every $Z \in \mathcal{Z}$, we make the final decision on the taxonomic relation between x and y. Let ℓ denote the relation (i.e. edge) between x and y in t^*. The set of all t^* is divided into groups with respect to ℓ. We denote these groups as $\mathcal{T}_\ell$. To choose the best taxonomic realtion, ℓ^*, of x and y, we solve the objective function defined in Eqn. 2.

$$\begin{aligned} \ell^* &= \text{argmax}_\ell \frac{1}{|\mathcal{T}_\ell|} \sum_{t \in \mathcal{T}_\ell} \lambda_t score(t) \qquad (2) \\ &= \text{argmax}_\ell \frac{1}{|\mathcal{T}_\ell|} \sum_{t \in \mathcal{T}_\ell} \lambda_t \left(\sum_{e \in t} w(e) - \sum_{k=1}^{|\mathcal{C}|} \rho_k d_{C_k}(t) \right) \end{aligned}$$

where λ_t is the weight of concept network t, defined as the occurrence probability of t (regarding only its edges' setting) in training data, which is augmented with additional concepts. Eqn. (2) finds the best taxonomic relation of two input concepts by computing the average score of every group of the best concept networks representing a particular relation with respect to a set of relational constraints. The relational constraints used in our experiments are manually constructed. We consider only 3-node concept networks (i.e. $|Z| = 1$). In general, n-node concept networks can be selected as relational constraints and used in our inference model. But one can always decompose n-node networks into 3-node networks to allow greedy search in the concept network space.

3.2 Related Concepts Extraction

To apply our constraint-based inference model, we need to acquire concepts additional to the input pair. In principal, with strong relational constraints and a reasonable local taxonomic relation classifier, one can use random concepts as additional concepts to do inference. However, there is a high chance that a random additional concepts will simply produce *no relation* with two input concepts, and eventually will not help the inference model. Our experiment (see Sec. 5

YAGO QUERY PATTERNS
INPUT: concept "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, resp.

Figure 3: Our YAGO query patterns used to obtain related concepts for "X".

shows that the proposed inference model gets better results when doing inference with additional concepts that are more relevant to input concepts. From now on, we refer to additional concepts as *related concepts*.

We, therefore, propose an approach that makes use of YAGO ontology (Suchanek et al., 2007) to provide related concepts. 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 lexical database WordNet.[2]

In YAGO model, all objects (e.g. *cities*, *people*, etc.) are represented as *entities*. Following the YAGO model, our input concepts are considered to be words. To map our input concepts to entities in YAGO, we use MEANS relation. Furthermore, similar entities are grouped into *classes*. This allows us to obtain direct ancestors of an entity by using TYPE relation which gives the entity's *classes*. Furthermore, we can get ancestors of a *class* with SUBCLASSOF relation. By using three relations MEANS, TYPE and SUBCLASSOF in YAGO model, we can obtain direct ancestors, siblings, and children, if any, of an input concept. Fig. 3 presents three patterns that we used to query related concepts from YAGO. For concepts having no children, such as *honda civic*, *bill clinton*, we simple ignore their children.

4 Learning Taxonomic Relations

In this section, we describe our machine learning-based classifier for taxonomic relations of concepts. For each input concept in the input pair, we first build its article representation by searching for a list of relevant articles in Wikipedia. To do this, we first use both concepts to search the Wikipedia index, and then extract important keywords in the titles and categories of the returned articles. The extracted keywords are combine with each input concept to create a final query that is used to search for a list of relevant articles in Wikipedia. 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 of concept X. As a learning algorithm, we use a regularized averaged Perceptron (Freund and Schapire, 1999). Below are features extracted for pair (X,Y) using their lists of relevant articles.

Bags of words: We define four bag-of-word features associated with any two given concepts X and Y: (1) the degree of similarity between the texts of concept X and the categories of concept Y, (2) the similarity between the categories of X and the texts of Y, (3) the similarity between the text of X and the text of Y, and (4) the similarity between the categories of X and the categories of Y. To collect categories of a concept, 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. We use cosine similarity metric to measure the degree of similarity between text fragments.

Association information: Intuitively, association information is the overlap information that is shared by two concepts. We capture this feature by the pointwise mutual information which gives the ratio between the number of times two concepts appeared together and the number of times they were expected to appear, independently.

Overlap ratios: This feature captures the fact that the titles of a concept usually overlap with the categories of its descendants. We measure this overlap as the ratio of *common phrases* used in the titles of concept X and the categories of concept Y. In our context, a phrase is considered to be a *common phrase* if it appears in the titles of X and the categories of Y and is one of the following: (1) the *whole string* of a category of Y, or (2) the *head* in its root form of a category of Y, or (3) the *post-modifier* of a category of Y.

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 *Illi-*

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

nois. Given two entity pairs *(Illinois, Chicago)* and *(City, Chicago)*, we can recognize that *Illinois* and *City* match the category *Cities in Illinois* of concept *Chicago*. Therefore, we have a strong indication that *Chicago* is a descendant of both *Illinois* and *City*.

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

5 Experimental Study

In this section, we first describe the datasets. Follow that, we present the experiments to evaluate our overall system and compare it with other systems using existing large-scale resources. We also give a deeper understanding about our system by performing several experimental analyses.

5.1 Datasets

In our experiments, two main datasets are used. The first dataset is generated from 40 semantic classes of almost 11,000 instances. The original semantic classes were manually constructed[3] and was used to evaluate information extraction tasks as in (Paşca, 2007; Paşca and Van Durme, 2008). The second dataset is generated from 44 semantic classes of more than 10,000 instances used in (Vyas and Pantel, 2009)[4]. The original classes of the second dataset were extracted from Wikipedia lists. In both datasets, we have both types of closed word semantic class (e.g. *chemical element*, *country*) and open word semantic class (e.g. *basic food*, *hurricane*). Moreover, there are classes with proper nouns (e.g. *actor* with *Mel Gibson*) and classes with common nouns (e.g. *basic food* with *rice*, *milk*).

Several semantic class names in the original data are written in a short form (e.g. *chemicalelem*, *proglanguage*). We, therefore, expand each semantic class name in the original dataset to some meaningful names which are used by all systems in our experiments. For example, *terroristgroup* is expanded to *terrorist group*, *torrorism*.

We refer to both name of a semantic classes and its instances as concepts. An example in the problem of taxonomic relation identification is a pair of two concepts (X, Y) such as (*city*, *Dubai*), (*lead*, *aluminum*). We pair the semantic classes and instances in the original data set to create training and testing examples. For each original dataset, we randomly generate disjoint training and test sets of $12,000$ and $8,000$ examples, respectively. We call the test set from the first and the second datasets **Test-I** and **Test-II**, respecitively. Table 1 shows some generated examples. All types of taxonomic relations including $X \leftarrow Y$, $X \rightarrow Y$, $X \leftrightarrow Y$, and $X \nleftrightarrow Y$ are covered with a balanced number of examples.

Relation	Concept X	Concept Y
$X \leftarrow Y$	actor	Mel Gibson
	food	rice
$X \rightarrow Y$	Makalu	mountain
	Monopoly	game
$X \leftrightarrow Y$	Paris	London
	copper	oxygen
$X \nleftrightarrow Y$	Roja	C++
	egg	Vega

Table 1: Some examples in our data set.

To evaluate our system, we use a snapshot of Wikipedia from July, 2008. After cleaning up and removing articles without a category (except redirect pages), and administrative articles such as *Template*, *Image* and *Portal* pages, there are 5,503,763 articles remained. We index these articles by using Lucene[5].

5.2 Overall Results and Comparison

To the best of our knowledge, no prior work directly targets the problem of on-the-fly taxonomic relation identification. Most prior work which relates to our problem focuses on building lexical taxonomy or knowledge ontology (Snow et al., 2006; Ponzetto and Strube, 2007; Suchanek et al., 2007). We compare our system with three offline knowledge bases which are built upon different existing resources.

Strube07 uses a large scale taxonomy which was derived from Wikipedia (Ponzetto and Strube, 2007), as the background knowledge. The taxonomy was created by applying several lexical matching and methods based on connectivity in the network to the category system in Wikipedia. The taxonomy used in this system is in the latest version from March, 2008[6]. **Snow06** uses the *augmented WordNet* (Snow et al., 2005; Snow et

[3]Private communication.

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

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

[6]Private communication.

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

Table 2: Evaluating and comparing performances, in accuracy, of the systems on **Test-I** and **Test-II**. TREI (local) uses only our local classifier to identify taxonomic relations by choosing the relation with highest confidence.

TREI	Test-I		Test-II	
	Prec	Rec	Prec	Rec
$X \leftarrow Y$	95.81	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

Table 3: Performance of our systems on individual taxonomic relation.

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

Table 4: Performance of the systems on special datasets, in accuracy. For all concept pairs from **non-Wikipedi**, TREI (local) simply returns sibling relation.

al., 2006) as background knowledge. To build the *augmented WordNet*, the authors first identified lexico-syntactic patterns indicative of hypernymy from corpora and use them to extract candidate noun pairs that may hold the hypernym relation. Starting from WordNet-2.1 (Fellbaum, 1998), the latest version of the augmented WordNet has augmented 400,000 synsets. Words that are added into the augmented WordNet can be common nouns or proper nouns. **Yago07** uses YAGO ontology (Suchanek et al., 2007) as the main source of background knowledge. Because YAGO ontology is a combination of Wikipedia and WordNet, this system is expected to be powerful in recognizing concept relationships. To access a concept's ancestors and siblings, we use patterns 1 and 2 in Fig. 3 to move up on the ontology from a concept. Our overall algorithm, **TREI**, is described in Fig. 1. We also evaluate our local taxonomic relation classifier (Sec. 4), which is referred as **TREI (local)**. To make classification decision with TREI (local), for a pair of concepts, we choose the predicted relation with highest confidence returned by the classifier.

In all systems compared, we vary the value of K from 1 to 4 as the levels to go up to collect information in the taxonomy or category tree. The best result of each system is reported. Table 2 shows the comparison of all systems evaluated on both **Test-I** and **Test-II**. Our systems, as shown, significantly outperform other systems implemented using existing sophisticated resources.

5.3 Experimental Analysis

In this section, we give some experimental analyses to understand other aspects of our system.

Performances on Individual Relation: This experiment shows the performance of our best system in Tab. 2 on individual taxonomic relation. Tab. 3 shows the results in precision and recall.

Evaluating on Special Datasets: We evaluate the systems on different dataset derived from **Test-I**. **Test-I** consists of $10,456$ concepts pairs with all concepts from Wikipedia and $1,544$ concept pairs with at least one *non-Wikipedia* concept. Furthermore, if we only consider concepts pairs with all concepts from WordNet and Wikipedia, there are $8,625$ pairs of interest. Tab. 4 shows the comparison of the systems on these dataset which are denoted as **Wikipedia**, **non-Wikipedia** and **WordNet**. In this experiment, Yago07 gets better results on **Wikipedia** concept pairs than when evaluated with **Test-I**. Snow06, as expected, gets better results on **WordNet** concept pairs. However, TREI still significantly outperforms other systems. The improvement of TREI over TREI (local) on the **Wikipedia** and **WordNet** datasets shows the contribution of the constraint-based inference model. Whereas, the improvement on **non-Wikipedia** dataset shows the contribution of function `findReplacement` (Sec. 2).

Contribution of Related Concepts in Inference: In this experiment, we show the results of TREI when using another source than YAGO to provide related concepts to the constraint-based inference model. We use the original data from (Paşca and Van Durme, 2008), which was used to generate **Test-I**. For each concept in an input pair from **Test-I**, we get its ancestors, siblings, chil-

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

Table 5: Evaluating TREI with different sources providing related concepts to do inference. K is the number of levels to go up on the Wikipedia category.

dren, if any, from the original data and use them in the inference process. This system is referred as TREI (Gold Inf.). Tab. 5 present the results of TREI and TREI (Gold Inf.) on different K as the number of levels to go up on the Wikipedia category. We see that **TREI** gets better results when doing inference with better related concepts. In this experiment, two systems use the same number of related concepts.

6 Related Work

To the best of our knowledge, no previous study has directly addressed the problem we study here in this form: given two concepts, identify the relation between them. However, there are several lines of related work that we would like to mention and compare to.

In (Snow et al., 2005), the authors construct a hypernym-only classifier between concepts. The classifier builds on dependency path patterns discovered in sentences that contain noun pairs in hypernym and hyponym relations. The best system presented there is a hypernym-only classifier that is trained with hundreds of thousands of dependency paths extracted from Wikipedia. The system outperforms the best WordNet classifier in identifying coordinated terms,, improving the relative F-score by over 54%. More recently, (Snow et al., 2006) proposed a method to train their (m,n)-cousin relationship classifier (defined similarly to our *sibling* relation) to significantly increase coverage and improve the F-score by 23% over the WordNet-2.1 hypernym classifier. The classifier is then applied to build the *augmented WordNet* by augmenting WordNet-2.1 with over $400,000$ synsets. These works are different than ours because they largely build on the redundancy of information in the web—many related terms appear often enough in some simple explicit form in close proximity in a sentence. In our work, we relax this fact and identify relations between any two input concepts, exhibiting significantly better coverage and accuracy.

Other related works attempts to build relational knowledge bases (semantic taxonomies and ontologies) that represent entities and relations among them. E.g. (Ponzetto and Strube, 2007) generates a taxonomy using Wikipedia as the knowledge source. They use the category system of Wikipedia as a conceptual network consisting of the subsumption (*isa*) relation. Similarly (Suchanek et al., 2007) presents the the YAGO ontology, which is automatically constructed using both Wikipedia and WordNet to mine entities and their relations. The YAGO approach builds on the *infoboxes* and *category pages* in Wikipedia and links to the clean taxonomy of concepts in WordNet. YAGO provides many useful relations between entities such as *subClassOf*, *bornOnDate*, *locatedIn*, and *type*. One could possibly use the taxonomy in (Ponzetto and Strube, 2007) or the YAGO ontology, with some additional work, to find relation of two input concepts. Our work differs from this line of work in that we do not build an offline knowledge base, but rather directly identify relations between input concepts. Nevertheless, as we show, our approach has both better coverage and better accuracy.

7 Conclusions and Future Work

We study an important component of textual inference–identifying identifying taxonomic relations between pairs of concepts. We argued and provided experimental evidence that static structured knowledge bases cannot support this task well enough and developed a novel approach for this problem. We developed TREI, an algorithm that fuses information from existing knowledge sources and uses machine learning and a constrained optimization technique to get around the noise and the level of uncertainty inherent in these resources. The experimental study shows that our system is significantly better than other systems that use existing well-known structured resources, showing that our approach generalizes well across semantic classes and handles well concepts that are not mentioned in Wikipedia. The key lesson from the success of our approach has to do with our combined learning and global inference approach. Furthermore, we demonstrate an effective approach to leveraging existing knowledge bases for this inference process. Our future research will include an evaluation of TREI in the context a textual inference application.

References

M. Banko, M. Cafarella, M. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *IJCAI*.

R. Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment in natural language. In *AAAI*.

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

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *NAACL*.

I. Dagan, O. Glickman, and B. Magnini, editors. 2006. *The PASCAL Recognising Textual Entailment Challenge*. Springer-Verlag, Berlin.

Dmitry Davidov and Ari Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *ACL-HLT*.

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.

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

A. Haghighi, A. Ng, and C. Manning. 2005. Robust textual inference via graph matching. In *HLT-EMNLP*.

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

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*.

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

V. Punyakanok, D. Roth, and W. Yih. 2005a. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005b. Learning and inference over constrained output. In *IJCAI*.

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

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *ICML*.

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

S. Sekine. 2006. On-demand information extraction. In *ACL*.

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

Rion Snow, Daniel Jurafsky, and Andrew 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*.

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