Relational Constraint-based Taxonomic Relation Identification

Abstract

Determining whether two concepts in text have an ancestor relation (e.g. *Toyota* and *car*) or a sibling relation (e.g. *Toyota* and *Honda*) is often essential to support many tasks in computational linguistics. Significant effort has been devoted to developing structured knowledge bases that could potentially support these tasks. But these resources usually suffer from noise and uncertainty, making their use as background knowledge difficult. We argue that, given two concepts, the problem of determining what taxonomic relation holds between them should be addressed in a machine learning fashion. More importantly, we leverage an existing knowledge base to enforce relational constraints among concepts in a global inference process to accurately identify taxonomic relations. Experiments show that our approach significantly outperforms other systems built upon existing well-known structured knowledge bases.

1 Introduction

Structured knowledge bases like taxonomies and ontologies play important roles in many computational linguistics tasks, such as (Hotho et al., 2003; Chakrabarti et al., 1997). Recently, the Textual Entailment challenge (Dagan et al., 2006) has shown that taxonomic relations can greatly help textual inference which requires the use of large amounts of background knowledge. 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, (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.

In this paper, we focus on identifying two general types of taxonomic relations - *ancestor* and *sibling*. An ancestor relation and its directionality can help us 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 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 an 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 work has been devoted to acquiring structured taxonomies and ontologies resulting in structured knowledge bases such as the *augmented WordNet* (Snow et al., 2006) and YAGO (Suchanek et al., 2007), which represent taxonomic information of individual concepts. We believe that these acquired structured knowledge bases are important resources to support classification of taxonomic relations. However, these resources usually suffer from noise and, especially, uncertainty with respect to each concept, making it difficult to support robust classification of taxonomic relations between two given concepts.

In this paper, we present our system to identify taxonomic relations using machine learning-based approach. More importantly, we describe our constraint-based inference model that incorporates relational constraints as prior knowledge to accurately identify taxonomic relations. Furthermore, we present a novel approach to leveraging an existing knowledge base, which is by itself

weaker than our system in identifying taxonomic relations, to provide useful information to our inference model.

The rest of this paper is organized as follows: Sec. 2 gives an overview of our approach. We then present our learning component in Sec. 3. Sec. 4 describes our inference model that leverages global relational constraints to infer taxonomic relations. Sec. 5 presents our experimental study, and Sec. 6 describes related work.

2 Taxonomic Relation Identification: An Overview

First, we give an overview of our **T**axonomic **RE**lation **I**dentification (**TREI**) algorithm addressing the problem of identifying taxonomic relation between concepts. We define a *concept* as a single word or phrase that refers to an entity, such as *mountain*, *George W. Bush*, *Empire State building*. We focus on classifying four fundamental taxonomic relations between two concept X and Y: (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 $X \leftrightarrow Y$; and (4) X and Y have *no relation*, denoted by $X \nleftrightarrow Y$.

Given two concepts, we use Wikipedia and a learned classifier in an inference model to predict the kind of taxonomic relation between these two. As a motivation for the inference model, while the learned classifier by itself can predict the relation between two concepts, we observe that in the presence of additional related concepts, we can create concept networks of relations among the input and additional concepts where relational constraints can be enforced to make a coherent final prediction. We call the outcome of the inference model the global prediction.

In the big picture of our approach, we first learn a classifier to identify taxonomic relations (Sec. 3) to use in our inference model (Sec. 4). The input of the learning component is a set of annotated examples, each of which consists of two concepts and the taxonomic relation between them.

Our inference model is presented in Figure 1, which consists of three steps.

Taxonomic RElation Identification (TREI)

INPUT: A concept pair (X, Y)
A learned taxonomic relation classifier $\mathcal{R}$
Wikipedia $\mathcal{W}$
OUTPUT: Taxonomic relation r^* between X and Y

1. $(X, Y) \leftarrow NormalizeToWikipedia(X, Y)$
2. $\mathcal{Z} \leftarrow ExtractRelatedConcepts(X, Y)$
3. $r^* = ConstraintbasedInference(X, Y, \mathcal{Z}, \mathcal{R}, \mathcal{W})$

RETURN: r^*;

Figure 1: TREI inference model.

1. Normalizing input concepts to Wikipedia: Although most commonly used concepts have corresponding Wikipedia articles, there are still a lot of concepts having no corresponding Wikipedia articles. For input concepts that we can find corresponding Wikipedia pages, we leave them as are. For a non-Wikipedia concept, we search the Web and try to find a replacement for it. 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 concepts are siblings and refer to two writers), we can continue the process of identifying the taxonomic relation of the pair (*Marjan Rožanc*, *Rudi Šeligo*). Our method was motivated by (Sarmento et al., 2007). We first make a query with two input concepts (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 concepts should be among c_a, c_b, c_c, ...). The delimiter can 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 a list 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 having Wikipedia pages is used as a replacement.

2. Extracting additional related concepts: In this step, we leverage an existing knowledge

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

[2] Periods and asterisks capture enumerations.

base to extract additional concepts related to the input concepts (Sec. 4.2,) which are all used in the inference model in the next step.

3. Making global prediction using relational constraints: We use Wikipedia, the learned local classifier, and the extracted additional concepts to enforce relational constraints among the concept relations to make final global prediction on the taxonomic relation between two input concepts (Sec. 4.1.)

3 Learning Taxonomic Relations

In this section, we describe our local taxonomic relation classifier.

For each input concept in the input pair, we first build its Wikipedia-article representation which is a list of top relevant articles in Wikipedia retrieved by a local search engine[3]. To do this, we use the following procedure: (1) Using both concepts 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 concept with the extracted keywords (e.g. *"George W. Bush"* AND *"president"* AND *"politician"*) to create a final query used to search for the concept's relevant articles in Wikipedia. This procedure was motivated by the assumption that real world applications usually have a need to know the taxonomic relation between two concepts which are related and in the same sense. For example, if the input pair is (*George W. Bush*, *Ford*), it's more likely that concept *Ford* refers to the former president of the United States, *Gerald Ford*, than the founder of Ford Motor Company, *Henry Ford*.

After having Wikipedia-article representation for each concept, we extract learning features of two input concepts from the articles in the representations. All these features are used to train a local taxonomic relation classifier. It is worth noting that a Wikipedia page usually consists of a title (i.e. the concept), a body text, and a list of categories to which the page belongs. Table 1 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. The learning features extracted for input pair (X,Y) are described below.

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 similarity features associated with any two given concepts X and Y: (1) the degree of similarity between the texts of X and the categories of Y, (2) the similarity between the categories of X and the texts of Y, (3) the similarity between the texts of X and the texts 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.

Association Information: Intuitively, association information is the overlap information that is shared by any two concepts. We capture this feature by the pointwise mutual information which quantifies the discrepancy between the probability of two concepts appearing together versus the probability of each concept appearing independently. The pointwise mutual information for two concepts X and Y is defined as following.

$$PMI(X,Y) = log\frac{p(X,Y)}{p(X)p(Y)} = log\frac{Nf(X,Y)}{f(X)f(Y)}$$

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

Overlap Ratios: The overlap ratio features 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 one concept and the categories of the other concept. In our context, a phrase is considered to be a *common phrase* if it appears in the titles of one concept and the categories of the other concept and is one of the following: (1) the *whole string* of a category, or (2) the *head* in its 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*,

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

Title/Concept	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 1: Examples of texts and categories of Wikipedia articles.

and a post-modifier *Illinois*. Given concept pair *(City, Chicago)*, we observe that *City* matches the head of the category *Cities in Illinois* of concept *Chicago*. Thus, we have 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 concepts. For this feature, we do not use the *post-modifier* of the categories. We use Jaccard similarity coefficient to measure these overlaps ratios.

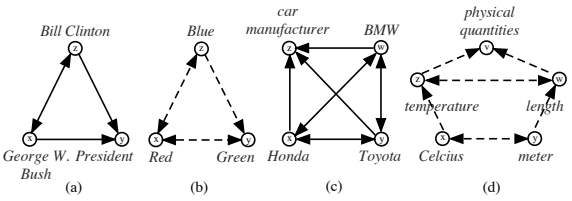

Figure 2: Examples of n-node concept networks with two input concept x and y. Networks (a) and (c) show valid combinations of edges, whereas (b) and (d) are two relational constraints. For simplicity, we do not draw *no relation* edges in (d).

4 Inference with Relational Constraints

The key insight of the inference model is that if we have more than two concepts, only some kinds of relations between them are possible. 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 concepts and their relations a *concept network*. Figure 2 shows some n-node concept networks consisting of two input concepts (x, y), and additional concepts z, w, v.

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 a global prediction on the taxonomic relation of two given concepts. 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).

4.1 Constraints as Prior Knowledge

Let x, y be two input concepts, and $\mathcal{Z} = \{z_1, z_2, ..., z_m\}$ 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 (Sec. 3). If $l = |Z|$, there are $n = 2 + l$ nodes in each network, and $4^{\left[\frac{1}{2}n(n-1)\right]}$ concept networks can be constructed. We only focus on 3-node concept networks (i.e. $l = 1$). For example, for input pair (*red*, *green*) and $\mathcal{Z} = \{blue, yellow\}$, we can construct 64 networks for triple $\langle red, green, Z = \{blue\}\rangle$ and 64 networks for $\langle red, green, Z = \{yellow\}\rangle$ by trying all possible relations between the concepts.

A *relational constraint* is defined as a concept network, by regarding only its edges' setting, that belongs to a pre-defined list of invalid edges' 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 concepts *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 concept networks matching constraint $C_k \in \mathcal{C}$.

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

function $d_{C_k}(t)$ indicates if t matches C_k. In our work, we use relational constraints as hard constraints and set their penalty ρ_k to ∞. For a set of concept networks formed by $\langle x, y, Z \rangle$ and all possible relations between the concepts, we select the best network, $t^* = \text{argmax}_t score(t)$.

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 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 concept 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.

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

where λ_t is the weight of concept network t, defined as the occurrence probability of t (regarding only its edges' setting) in the training data, which is augmented with additional concepts. Equation (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 of two input concepts.

4.2 Extracting Related Concepts

To apply our constraint-based inference model, we need to acquire concepts additional to the input pair. From now on, we refer to additional concepts as *related concepts*. The related concept space is defined as 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 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.[4]

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, respectively.

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

In YAGO model, all objects (e.g. *cities*, *people*, etc.) are represented as *entities*. To map our input concepts to entities in YAGO, we use MEANS relation defined in YAGO ontoloy. 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[5]. 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. Figure 3 presents three patterns that we used to query related concepts from YAGO ontology.

5 Experimental Study

In this section, we first describe the datasets. After that, we present the experiments that show the performance of our systems and other systems built upon existing large-scale resources. We also give a deeper understanding about our systems by several experimental analyses.

5.1 Datasets

We create and use two main datasets in our experiments.

The first dataset is generated from 40 semantic classes of almost 11,000 instances. The original semantic classes and instances were manually constructed without much manual post-filtering and were used to evaluate information extraction tasks as in (Paşca, 2007; Paşca and Van Durme, 2008). This dataset, therefore, contains concepts

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

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

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 2: Some examples in our data set.

having Wikipedia pages, such as *George W. Bush*, and also non-Wikipedia concepts, such as *hindu mysticism*. We create learning examples for our task by randomly pairing the semantic classes and instances. We generate disjoint training and test sets of 8,000 and 12,000 examples, respectively. We call the test set of this dataset **Test-I**.

The second dataset is generated from 44 semantic classes of more than 10,000 instances used in (Vyas and Pantel, 2009)[6]. The original semantic classes and instances were extracted from Wikipedia lists. This data, therefore, only contains concepts having Wikipedia pages (e.g. *pianist*, *roger federer*). We also generate disjoint training and test sets of 8,000 and 12,000 examples and call the test set of this dataset **Test-II**[7].

Several semantic class names in the original data are written in short forms (e.g. *chemicalelem*, *proglanguage*). We, therefore, 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 2 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 balanced numbers of examples in all datasets.

To evaluate our systems, 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 using Lucene[8]. As a learning algorithm, we use a regularized averaged Perceptron (Freund and Schapire, 1999).

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

[7]Omitting the dataset shared URL for blind review.

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

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. We compare our system with three systems that we built upon different existing resources.

Strube07 uses a large scale taxonomy, T_{Strube}, which was derived from Wikipedia (Ponzetto and Strube, 2007). The taxonomy was created by applying several lexical matching and methods based on connectivity in the network to the category system in Wikipedia. We use the latest available version of the taxonomy. It is worth noting that the taxonomy in Strube07, T_{Strube}, was extracted from and is similar to the page structure of Wikipedia (Ponzetto and Strube, 2007). We, therefore, first build the Wikipedia-article representation for each input concept by following the same procedure used in our work described in Sec. 3. The titles and categories of the articles in the representation of each input concept are then extracted. Only titles and their corresponding categories that are in T_{Strube} are considered. A concept is an ancestor of the other if at least one of its titles is in the categories of the other concept. If two concepts sharing at least one category, we predict that they are siblings. Otherwise, there is no relation between two input concepts. In the relation identification process, the ancestor relation is preferred, then sibling, and finally no relation.

Snow06 uses the *augmented WordNet* (Snow et al., 2006). Words in the augmented WordNet can be common nouns or proper nouns. Given two input concepts, we first map them onto the augmented WordNet by exact string matching. A concept is an ancestor of the other if it can be found as an hypernym after going up some levels in the hierarchical structure of the augmented WordNet from the other concept. If two concepts sharing a common subsumption by going up some levels from them, then they are considered as siblings. Otherwise, there is no relation between the two input concepts. The order of relation checking process is similar to Strube07: first ancestor, then sibling, and finally no relation. We use the Java WordNet library[9] to work with the augmented WordNet.

[9]http://sourceforge.net/projects/jwordnet/

	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 3: 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 4: Performance of our systems on individual taxonomic relation.

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 Figure 3 to map a concept to the ontology and move up on the ontology. The relation identification process is then similar to that of Snow06.

If an input concept is a non-Wikipedia concept, all these three systems simply pick *no relation* as the prediction.

Our overall algorithm, **TREI**, is described in Figure 1. We manually construct a pre-defined list of 35 relational constraints to use in the inference model of TREI. We also evaluate our local taxonomic relation classifier (Sec. 3), 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 3 shows the comparison of all systems evaluated on both Test-I and Test-II. Our systems, as shown, significantly outperforms other systems.

We do not have special tactics to handle polysemous concepts. However, our procedure of building Wikipedia-article representations for input concepts described in Sec. 3 ties the senses of two input concepts to each other. 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 a concept is mapped onto the YAGO ontology by using the MEANS operator as in Pattern 1 in Figure 3. This cannot follow our proposed procedure.

Observing the results shows that one of the main reasons that our systems, TREI (local) and TREI, perform much better than other systems is that while our machine learning-based classifier is very flexible in extracting features of two input concepts and predicting their taxonomic relation, other systems rely heavily on string matching techniques, which are very inflexible, to decide the relation. This clearly shows the limitation of using existing structured resources to identify taxonomic relations.

In Table 3, the improvement of TREI over TREI (local) on Test-I shows the contribution of both normalization procedure and our inference model that makes global prediction, whereas the improvement on Test-II shows the contribution of the inference model solely because Test-II contains only concepts having corresponding Wikipedia articles.

5.3 Experimental Analysis

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

Performances on Individual Relation: This experiment shows the performance of our best system on individual taxonomic relation. Table 4 presents the results in precision and recall. The result shows that our system performs very well on ancestor relation. Sibling and no relation are the most difficult relations to identify.

Evaluating with Special Datasets: We evaluate all systems on three special datasets derived from Test-I. From 12,000 examples of Test-I, we create the **Wikipedia** test set consisting of $10,456$ examples, each of which has two concepts in Wikipedia. We use the rest of $1,544$ exam-

	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 5: Performance of the systems on special datasets, in accuracy. For examples in the non-Wikipedia test set, TREI (local) simply returns sibling relation.

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

Table 6: Evaluating TREI with different sources providing related concepts to do inference. K is the number of levels to go up on the Wikipedia category system.

ples with at least one non-Wikipedia concept to make the **non-Wikipedia** test set. Furthermore, if we only consider examples with all concepts in WordNet and Wikipedia, there are $8,625$ examples of interest. We use these examples to make the **WordNet** test set. Table 5 shows the comparison of the systems on these datasets. In this experiment, Yago07 gets better results on the Wikipedia test set than when evaluated with Test-I. Snow06, as expected, gets better results on the WordNet test set. TREI still significantly outperforms other systems. The improvement of TREI over TREI (local) on the Wikipedia 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 concepts to Wikipedia.

Contribution of Related Concepts in Inference: We evaluate TREI when provided with related concepts from another source different than YAGO. To do this, we use the original data which was used to generate Test-I. For each concept in the examples of Test-I, we get its ancestors, siblings, children, if any, from the original data and use them as related concepts in the inference model. This system is referred as **TREI (Gold Infer.)**. Table 6 shows the results of TREI and TREI (Gold Infer.) on different K as the number of levels to go up on the Wikipedia category system. 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 of identifying taxonomic relation between two given concepts. However, there are several work which acquires structured taxonomies and ontologies.

In (Snow et al., 2005) and (Snow et al., 2006), the authors constructed classifiers to identify hypernym relationship between concepts. The classifiers is then used to augment WordNet (Fellbaum, 1998) with over $400,000$ synsets.

(Ponzetto and Strube, 2007) generated a taxonomy using Wikipedia as the knowledge source. They used the Wikipedia category system as a conceptual network consisting of subsumption (*isa*) relations. (Suchanek et al., 2007) presented the the YAGO ontology, which was automatically constructed using both Wikipedia and WordNet. The YAGO approach built on the *infoboxes* and *category pages* in Wikipedia and linked to a clean taxonomy of concepts in WordNet. YAGO provides many useful relations such as *subClassOf*, *bornOnDate*, and *type*. Our work differs from this line of work because we do not build a knowledge base, but rather directly identify taxonomic relations between two given concepts.

7 Conclusions and Future Work

We studied an important component of many computational linguistics tasks – identifying taxonomic relations between pairs of concepts. We argued and provided experimental evidences that static structured knowledge bases cannot support this task well enough. We developed TREI, a novel algorithm that leverages information from existing knowledge sources and uses machine learning and a constraint-based inference model to get around the noise and the level of uncertainty inherent in these resources. The experimental study showed that TREI is significantly better than other systems which were built upon existing well-known structured resources. Our approach generalized well across semantic classes and handled well non-Wikipedia concepts. Our future research will include an evaluation of TREI in the context of textual inference applications.

References

Chakrabarti, Soumen, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. 1997. Using taxonomy, discriminants, and signatures for navigating in text databases. In *VLDB*.

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

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

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

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

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

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

Hotho, Andreas, Steffen Staab, and Gerd Stumme. 2003. Ontologies improve text document clustering. In *ICDM*.

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

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

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

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

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

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

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

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

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*.

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

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