# Constraint-based Inference towards Identification of Relational Knowledge


## ABSTRACT

We present a novel approach to identifying relations between pairs of concepts. We focus on identifying relations that are essential to support textual inference: determining whether two concepts have an ancestor relation, a sibling relation, or no relation. We develop a machine learning based approach that makes use of Wikipedia as a main source for background knowledge, but we also propose an effective approach to improve the coverage of our method and support inference between concepts that are not mentioned in Wikipedia. Our key innovation is that, in order to accurately determine the relations between concepts $C_1$ and $C_2$, we consider an automatically generated collection of related concepts, evaluate all pairwise relations, and use constraint based inference to force them to cohere, thus improving the local prediction of the pairwise relation identification. We demonstrate that the inference technique significantly enhances the local prediction methods and consequently exhibit large improvements over existing methods.



## Categories and Subject Descriptors

H.4 [**Knowledge Management**]: Information Extraction

## General Terms

Relation identification

## Keywords

Concept, relation, classification, Wikipedia


## 1. INTRODUCTION

Many data and knowledge management problems require some sort of textual inference. These range from expansion and query interpretation in information retrieval to query schema matching to question answering. Textual Inference, in turn, 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.

The fundamental problem expressed here is that of recognizing basic relations between two given concepts. This view of recognizing relations is different from the Open Information Extraction effort [1] and On-Demand Information Extraction [20], which aim to extract large databases of open-ended facts and build the two concepts occurring together in some local context. It is also different from the effort on supervised relation extraction[18] which requires additional supervised data to learn new relations.

Moreover, a textual inference system that needs to know if concepts $A$ and $B$ are related, and how, cannot benefit from the large body of literature on knowledge acquisition that has extracted all *easy to find* facts in a given corpus [2, 5, 15], since it will know of $A$ and $B$ only if they have occurred in an explicit way and in close proximity in a sentence. Indeed, we know of no successful application of the large scale existential knowledge acquisition efforts to textual inference.

In the context of Textual Inference [4, 9, 3], it has been argued, e.g., in [12] that many inferences are largely compositional and depend on the ability to recognize specific relations between concepts such as entities, noun phrases, etc. 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*). This is illustrated in the following example, taken from the RTE4 test data:

> **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 *contradict* an identical statement with respect to *Japan* (at least, without additional information) since these are *different* countries. This is illustrated in the following example from RTE4 test data.

> **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*.

This paper proposes to address the problem of relation identification and classification in a form that is directly applicable to textual inference. Specifically, our system accepts two input concepts as arguments (these could be entities or noun phrases) and identifies the relation between them . For example, we identify that *global warming* and *food crisis* are in a *sibling* relation, and the concept of *economic problems* is in an *ancestor* relation with both of them.

We focus here on the *ancestor* relation and the *sibling* relation, that were identified as key relations also in [12] (the *sibling* relation is called an *alternation* there, and our *ancestor* relation is called *forward entailment* and *backward*



*WWW2010*, April 26-30, 2010, Raleigh, North Carolina.

.

*entailment*). Following the logic developed there, we expect that the resource we develop in this work can be used compositionally to support robust textual inference.

We develop an approach that makes use of Wikipedia that given a pair of concepts, can, on the fly, recognize and classify the relation between these concepts. While Wikipedia is vast resource that is rapidly being updated, our approach needs to take into account that it is developed by a large number of people, and is thus very noisy and non-uniform. Our algorithmic approach therefore treats Wikipedia and its category structure as an open resource and uses statistical text mining techniques to get robust information from this noisy resource. For example, the concept *Ford* appears many times in Wikipedia, and is part of a large number of categories. As a *president*, mentions of *Ford* are inconsistent with mentioned of other presidents. However, it appears also in other senses, related to the *car* industry, for example. We need to disambiguate it and determine which category it belongs to and, within this category, which specific concept is intended. In order to disambiguate it, we make use of the context provided by the concept pair—*Ford* in (*Ford*, *Nixon*) is probably different than the one in (*Ford*, *Chevrolet*) and even from the one in (*Ford*, *Iacocca*).

Textual inference is driven by background knowledge. Therefore, the notion of *prominence* is an essential notion in supporting textual inference. Most people *know* that *Michael Jordan* is a former NBA player, and do not know the *Michael Jordan* who goes to school with the authors. Consequently, unless additional knowledge is given, a textual inferences system should assume that *Michael Jordan* ia a basketball player. This is the reason we use Wikipedia as our background knowledge source. Moreover, under the assumption that in textual infernce applications we are in search of some notion of "common sense" knowledge we make use of a notion of prominence with respect to a given text collection (in this case, with respect to Wikipedia itself). Clearly, while Wikipedia has broad coverage that is sufficient for many application, one may want to go beyond it in some cases. We suggest a simple technique to makes use of the web to do that, and show its effectiveness when at least one of the target concepts is not mentioned in wikipedia.

We measure the performance of our system over a large number of pairs chosen from over 40 semantic classes. We compare it with other large scale efforts to identify relations between concepts. For example we show that, even when all concepts are covered by the *extended WordNet* [22], our system still significantly outperforms that system.

The key contributions of this paper are (i) the definition of a relation identification problem that is directly relevant to supporting textual inference and (ii) the development of a robust and accurate machine learning based approach to address this problem. We show that this approach has significantly better coverage and accuracy than other approaches that can only identify relations between concepts that occur in an explicit way and in close proximity or other Wikipedia-based approaches. Notably, our algorithmic approach is trained with a small number of annotated examples and generalizes well across semantic classes.

## 2. RELATED WORK

In [21], the authors construct a hypernym-only classifier to identify hypernym relation between concepts. The classifier builds on dependency path patterns discovered in sentences that contain noun pairs in hypernym and hyponym relations. The best system in their work is a hypernym-only classifier additionally trained with hundred thousands of dependency paths extracted from Wikipedia corpus. The system outperforms the best WordNet classifier in identifying coordinated terms by relatively improving F-score by over 54%. More recently, [22] proposed a method to train their ($m$,$n$)-cousin relationship classifier (defined similarly to our *sibling* relation) to significantly increase the coverage and improve the F-score by 23% over the WordNet-2.1 hypernym classifier. The classifier is then applied to build the *extended WordNet* by augmenting WordNet-2.1 with over $400,000$ synsets. However, these work are very different with ours because mostly they use the fact that due to redundancy of information in the web, a lot of related terms with appear often enough in some simple explicit form in close proximity. In our work, we relax the fact to consider

Other related work focuses on building relational knowledge bases such as large-scale semantic taxonomy and ontology that represent entities and their relations. In [16], the author generate a taxonomy using Wikipedia as the knowledge source. The authors use the cateogory system in Wikipedia as a conceptual network to derive a taxonomy containing a large amount of subsumption, i.e. *isa* relations. In the same line of work, the authors in [23] present YAGO ontology, which is constructed automatically. Wikipedia and WordNet are used as the main source of knowledge to mine entities and relations for YAGO. The YAGO approach builds on the *infoboxes* and *category pages* in Wikipedia, and links to the clean taxonomy of concept from WordNet. YAGO can provide many useful relations between entities such as *subClassOf*, *bornOnDate*, *locatedIn*, and *type*. Using the taxonomy in [16] or YAGO ontology, with some additional work, one may be able to get the answer for relation of two input concepts. However, there are some basic differences between our work and YAGO ontology. The most important difference is that our work is not building an offline knowledge base, but targeting on identifying relations between input concepts. We are now focusing on recognizing if two input concepts hold *ancestor*, *sibling*, *children* or *no relation* to each other. We develop a machine learning based approach that utilizes relational constraints to facilitate the inference process that help identifying relations efficiently.

## 3. IDENTIFYING RELATIONAL KNOWLEDGE

Our approach is a machine learning based algorithm. We first present an algorithm to disambiguate given concepts. We then extract features of disambiguated concepts and train a supervised multi-class classifer using Wikipedia as the background knowledge to identify relations between concepts. Given a pair of concepts ($X$, $Y$), we focus on addressing the following relation types:

1. $X$ is an ancestor of $Y$, denoted by $X \leftarrow Y$.
2. $Y$ is an ancestor of $X$, denoted by $X \rightarrow Y$.
3. $X$ and $Y$ are siblings, denoted by $X \leftrightarrow Y$.
4. $X$ and $Y$ have no relation, denoted by $X \nleftrightarrow Y$.

We also propose a simple and effective method to improve the coverage of our system by going beyond Wikipedia to

identify relations between concepts that are not mentioned in Wikipedia.

## 3.1 Concept Disambiguation

By posing the problem as identifying relations, we have to deal with the problem of disambiguating given concepts. The more accurate the disambiguation is, the more precise the relation recognition. For example, the concept *ford* in the concept pair *(bush, ford)* may refer to several concepts, such as *Ford Motor Company*, president *Gerald Ford*, or *Ford* city in Wisconsin - USA. Our approach is motivated by observing that the concept *bush* can be used to disambiguate *ford* and vice versa. Eventually, we can put two concepts *George W. Bush* and *Gerald Ford* in the top retrieved lists of *bush* and *ford*.

We use Explicit Semantic Analysis (ESA) [8] to generate a list of top semantically relevant articles in Wikipedia for both given concepts. For each article in the top list, we extract and keep its categories. All categories are then tokenized to create a bag of tokens relevant to the two given concepts. We use TF-IDF score to weigh the importance of each token. Only the top important tokens, which are different from the two input entities, are kept to use in the next step of our algorithm. Each given concept is then concatenated with the top weighed tokens to create a new query. We use the new queries to search titles and texts of Wikipedia articles to get the top relevant articles. These lists of articles later are used to provide features for our relation classifier.

Figure 1 shows the pseudo codes of our concept disambiguation algorithm. In this algorithm, function $ESA(query)$ retrieves the most semantically relevant articles in Wikipedia to $query$. All categories of a $list$ of articles are extracted by function $getCategories(list)$. Function $search(query)$ retrieves relevant articles in Wikipedia by searching $query$ in the articles' titles and texts.

```
CONCEPT DISAMBIGUATION ALGORITHM
    INPUT: Two concepts X and Y
    OUTPUT: A list of relevant articles in Wikipedia.
            for each concept.

    Query q ← Concatenating X and Y;
    L = ESA(q); // return a list of relevant articles to X and Y

    C = getCategories(L);
    T = tokenize(C);
    T_top = top^K_tfidf(T); // pick top K tokens

    Query q_x ← Concatenating x and tokens from T_top;
    Query q_y ← Concatenating y and tokens from T_top;
    A_x = search(q_x);
    A_y = search(q_y);

    RETURN: A_x and A_y;
```

**Figure 1: Pseudo code of our concept disambiguation algorithm. The algorithm takes as input two concepts. The algorithm disambiguates two given concepts and return a list of relevant articles in Wikipedia for each concept.**

## 3.2 Learning Concept Relations

Our goal is to design a supervised learning approach that identifies relational knowledge between concepts. We want our learning algorithm to be as general as possible so that by training on examples having concepts in some semantic classes, we are still able to classify examples in other semantic classes with high accuracy. To do this, we first select expressive features that will be associated with given concept pairs. These features are necessarily independent of the semantic class of the given concepts. Recall that, after the concept disambiguation step, each concept of a given pair is associated with a list of relevant articles in Wikipedia. These articles give lists of titles, texts, and categories for the corresponding concepts. From now on, we use *the titles of concept* $X$ to refer to the titles of the articles associated with concept $X$; similarly for *the texts of concept* $X$, and *the categories of concept* $X$. As learning algorithm, we use a regularized averaged Perceptron [7]. The features selected for our learning problem include the word usage of the associated articles, the association information between two concepts, and the overlap ratios of important information between the concepts and the articles. We describe these features below.

### 3.2.1 Word Usage

One of the most important features in recognizing relations between concepts is the word usage in associated articles. An article in Wikipedia typically contains three main components: title, text, and categories. The title distinguishes a concept from other concepts in Wikipedia. The text describes the concept. The categories classify the concept into one or more concept groups which can be further categorized. To collect the categories for a concept, we take the categories of its associated articles go up to $K$ level in the Wikipedia category system. In our experiments, we use abstracts of Wikipedia articles instead of whole texts.

By analyzing Wikipedia articles, one could observe that the word usage of the texts describing a concept often overlaps with the word usage of the categories of its inferior concepts. For example, the texts describing the concept *Presidents of the United States* overlap with the categories of *George W. Bush*, and *Gerald Ford* in the tokens *president, united, states* and so on (see Table 1.)

On the other hand, two sibling concepts often have overlap not only in their texts, but also in their categories. For example, concepts *George W. Bush* and *Gerald Ford* both use *presidents, united, states* in their texts, as well as *presidents, united, states, republican* in their categories.

We define four word usage features associated with any two given concepts $X$ and $Y$: the degree of similarity in word usage between the texts of concept $X$ and the categories of concept $Y$, the similarity between the categories of $X$ and the texts of $Y$, the similarity between the text of $X$ and the text of $Y$, and the similarity in word usage between the categories of $X$ and the categories of $Y$.

We now have to measure the degree of similarity in word usage for the features. There are several ways to calculate degree of similarity between two texts [13]. In this paper, we use the cosine similarity to measure the degree of similarity of word usage of two text fragments. Equation (1) shows the formula of the cosine similarity metric applying on the term vectors $T_1$ and $T_2$ of two text fragments.

| 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 the concepts from Wikipedia: *President of the United States*, *George W. Bush* and *Gerald Ford*.**

$$Sim(T_1, T_2) = cos\theta(T_1, T_2) = \frac{T_1 \cdot T_2}{\|T_1\| \, \|T_2\|} = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{j=1}^{N} x_j} \sqrt{\sum_{k=1}^{N} y_k}} \quad (1)$$

where $N$ is the vocabulary size, and $x_i$, $y_i$ are two indicator values in $T_1$ and $T_2$, respectively.

### 3.2.2 Association Information

Another useful feature that can help to recognize the relation between two concepts is the association information between them. We capture the association information between two concepts $X$ and $Y$ by measuring the pointwise mutual information, defined in equation (2). In this paper, the association information is measured in document level.

$$PMI(X, Y) = log \frac{p(X, Y)}{p(X)p(Y)} = log \frac{\frac{f(X,Y)}{N}}{\frac{f(X)}{N} \frac{f(Y)}{N}} = log \frac{N f(X, Y)}{f(X) f(Y)} \quad (2)$$

where $X$ and $Y$ are two input entities, $N$ is the total number of document in Wikipedia, and $f(.)$ is a function returning document frequency.

### 3.2.3 Overlap Ratio

Wikipedia articles are not only good in giving definition and explanation about concepts, they are also very helpful in categorizing concepts into groups and topics. We want to capture the fact that the titles of a concept often has an overlap with the categories of its descendants. For examples, the title (and also the concept itself) *Presidents of the United States* overlap with one of the categories of the concepts *George W. Bush* and *Gerald Ford* (see Table 1.) This phenomenon is a good feature to recognize the ancestor relation. 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 as a *common phrase* if it appears in the titles of $X$ and the categories of $Y$ and it is one of the following things:

- the *whole string* of a category of $Y$, or
- the *head* in its root form of a category of $Y$, or
- the *post-modifier* of a category of $Y$.

In our work, we use the Noun Group Parser from [23] to extract the *head* and *post-modifer* 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-modifer *Illinois*. 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*. This is a good indication that *Chicago* is a descendant of both *Illinois* and *City*.

To collect the categories for a concept, we collect the categories of its associated articles within $K$ level up in the Wikipedia category system. We measure the overlap ratio between the titles of concept $X$ and the categories of concept $Y$ by using the Jaccard similarity coefficient as shown in equation (3).

$$\sigma_{ttl}^{K}(X, Y) = \frac{\left|L_X \cap \mathcal{P}_Y^K\right|}{\left|L_X \cup \mathcal{P}_Y^K\right|} \quad (3)$$

where $L_x$ is the set of the titles of $X$, and $\mathcal{P}_y$ is the union of the category strings, the heads of the categories, and the post-modifiers of the categories of $Y$.

Similarly, we also use the ratio $\sigma_{ttl}^{K}(Y, X)$ as one of our features.

We also observe that there are many cases where two concepts are siblings or have no relation but they still have an overlap between their titles and their categories. To handle this, we add one more feature which is the overlap ratio of *common phrases* between the categories of two given entities. However, to capture the sibling relation, we do not use the *post-modifier* of the categories, because *post-modifier* does not help us to recognize sibling relation (e.g. *Cities in Illinois* and *Mountains In Illinois* have the same *post-modifier*, which is *Illinois*, but a city and a mountain cannot be sibling.)

The Jaccard similarity coefficient for overlap ratio between the categories of two concepts $X$ and $Y$ is showed in equation (4).

$$\sigma_{cat}^{K}(X, Y) = \frac{\left|\mathcal{Q}_X^K \cap \mathcal{Q}_Y^K\right|}{\left|\mathcal{Q}_X^K \cup \mathcal{Q}_Y^K\right|} \quad (4)$$

where $\mathcal{Q}_{(.)}^{K}$ is the union of the category strings and the heads of the categories. The categories are collected by going up to $K$ level in the Wikipedia category system.

All of the features described above are used in training our multi-class classifier to recognize relation between concepts.

## 3.3 Going Beyond Wikipedia

In our work, Wikipedia is the main source of background knowledge used to recognize concept relations. Although most commonly used concepts are mentioned in Wikipedia as we have argued above, there is still a need to identify relations between other concepts that do not have relevant articles in Wikipedia. We call these concepts *non-Wikipedia* concepts. In this section, we show how to improve the coverage of our approach to concepts outside Wikipedia. The idea is that we find, for each such concept, a replacement that is in Wikipedia. Our method was motivated by [19]. In their work, they address the set expansion problem. In order to identify concept pairs that belong to lists, they look for structures like *"... $ne_a$, $ne_b$ and $ne_c$ ..."* where $ne_a$, $ne_b$, etc. are named entities. E.g. *"I've lived in NY, Paris, and Amsterdam."* . Another possibility would be *"... $ne_a$, $ne_b$ or $ne_c$ ..."* . However, in their work, the authors use the texts from Wikipedia to look for entities in the desired structures. We are trying to cover concepts beyond Wikipedia. Therefore, we use a modified version of their structures and use Web search engines to search for concepts in textual lists. In our work, we use Yahoo! Web Search API[1]. We use Web search enignes to search for the conjunction of two given concepts (e.g. *"bass"* AND *"trout"* and look for the list structures *"... $\langle DELIMITER\rangle$ $c_a$ $\langle DELIMITER\rangle$ $c_b$ $\langle DELIMITER\rangle$ $c_c$ $\langle DELIMITER\rangle$ ..."* in the top snippets returned. The delimiters used in our experiments are comma(,), puntuation(.), and asterisk(*). For sentences that contain our structures, we extract $c_a$, $c_b$, etc. as candidate concepts which are in the same semantic class with our original given concepts. To reduce noise from concepts extracted from the snippets, we constrain the list structure to contain at least 4 concepts and each concept cannot be longer than 20 characters. Once a list of concept candidates was extracted, the concepts are ranked based on their frequency of occurrence. The highest ranked concepts are used to replace the non-Wikipedia concepts in the original given pair. The highest ranked candidates are tried with the concept disambiguation algorithm until we find the concepts mentioned in Wikipedia.

Note that unlike other pattern-based methods discussed in sec. 2 that search for concepts in close proximity, we use concepts in Wikipedia as anchors for the search, thus increasing the likelihood of covering less commonly used concepts.

# 4. INFERENCE WITH RELATIONAL CONSTRAINTS

From our real-case observations, we notice that there are several relational constraints among concept relations that can be used to enforce the relation between two input concepts identified by the system. For instance, concept *George W. Bush* cannot be an ancestor or sibling of concept *president* if we are confident that concept *president* is an ancestor of concept *Bill Clinton*, and *Bill Clinton* is a sibling of concept *George W. Bush*. Another example would be that concepts *red* and *green* are known to be siblings, and concept *blue* is also known as a sibling of *red*, the prediction identifying 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 3-vertex concept network examples including two input concepts ($x$, $y$), and an additional concept $z$. Fig. 2(b) illustrates a relational constraint.

In general, $n$ concepts can be involved to construct $n$-vertex concept network ($n > 2$). In this paper, we focus on observing 3-vertex concept networks with two input concepts and an additional one. However, our formalization can be easily applied to general $n$-vertex concept networks. The problem of determining optimal value of $n$ is out of scope of this paper.

From the observations, if we can obtain additional concepts to the two input concepts, we can enforce such relational constraints as prior knowledge to do inference that guides the final decision of the relation identifier. In literature, there are several models that take advantage of prior knowledge to achieve significant improvement in performance (CITATIONS XXX YYY). There are two main approaches to inject prior knowledge. The first approach incorporates prior knowledge *indrectly*; by adding more features (CITATIONS XXX YYY). In the other hand, the other approach incorporate prior knowledge *directly* under the form of hard or soft constraints (CITATIONS XXX YYY). In our work, we want our model to incorporate prior knowledge directly, therefore our model is closely related to the latter approach. We first present our inference model that incorporates prior knowledge. After that, we propose an approach that allow us to obtain additional concepts which are related to two input concepts.

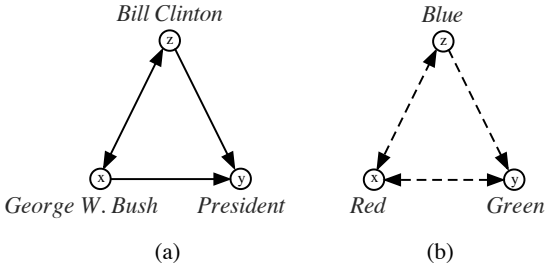


**Figure 2: Examples of 3-vertex concept networks formed by two input concepts $x$, $y$, and a related concept $z$. The relation between concepts is predicted by a local classifier (see Sec. 3.2). The left concept network (a) shows a valid combination of three edges. The right concept network (b) demonstrate an invalid combination of the edges. The right concept network forms a relational constraint that we do not allow to happen.**

## 4.1 Constraints as Prior Knowledge

Let two input concepts $(x, y)$, and a set of additional concepts $\mathcal{Z}^* = \{z_1, z_2, ..., z_m\}$. For a subset $\mathcal{Z} \in \mathcal{Z}^*$, a set of concept networks is constructed. Each concept network contains two input concept $x$, and $y$, and all additional concepts in $\mathcal{Z}$. Every two concepts in a concept network are connected by their relation. Four relations of interest in this paper are used. A local relation classifier is used to give weights for 4 relation classes of an edge in a concept network. We use $e$ and $w(e)$ to denote an edge in a concept network and its corresponding weight, respectively. If $n$ is the number of concepts in a network ($n > 2$), there will be

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

$\left[\frac{1}{2}n(n-1)\right]^4$ concept networks can be constructed due to 4 relations of every 2 concepts. In particular, with $n = 3$, we have 64 concept networks.

A relational constraint is defined as an invalid concept networks that is not allowed to happen. Figure 2(b) demonstrates a constraint where the combination of edges is (*sibling*, *sibling*, *ancestor*) following clockwise direction with respect to $x$ and $y$.

Following the approaches to inject the prior knowledge directly in (CITATIONS XXX YYY), we incorporate the relational constraints into our model to choose the best network $t^*$ from a set of concept networks constructed from $\langle x, y, \mathcal{Z}\rangle$. The scoring function is a linear combination of the edge weights $w(e)$ and the penalties $\rho_k$ of concept networks violating constraint $C_k \in \mathcal{C}$.

$$t^* = \text{argmax}_t \sum_{e\in t} w(e) - \sum_{k=1}^{|\mathcal{C}|} \rho_k d_{C_k}(t) \quad (5)$$

In Eq. 5, function $d_{C_k}(t)$ measures the degree that concept network $t$ violates constraint $C_k$. In our work, relational constraints are mined from the training data (see Sec. 4.2). The selected constraints are considered to have high confident about the knowledge. We, thus, use them as hard constraints, and set their penalty $\rho_k$ to $\infty$ (CITATIONS XXX YYY).

Objective function 5 allows us to pick the best setting of all edges connecting concepts in the networks with respect to relational constraints.

After having the model to pick the topmost concept network for a particular subset of additional concepts $\mathcal{Z} \in \mathcal{Z}^*$, we want to get the final decision on the relation between $x$ and $y$ by going over other subsets in $\mathcal{Z}^*$. For each $\mathcal{Z}$, the topmost concept network presents the most likely relation between two input concept $x$ and $y$. We repeat the process of finding the topmost concept networks for other subsets $\mathcal{Z} \in \mathcal{Z}^*$. After going through all subsets in $\mathcal{Z}^*$ and pick the topmost concept network in each case, a concept relation $\ell$ has a pool of topmost concept networks $\mathcal{T}_\ell$ in which each concept network contains the edge with relation $\ell$ between $x$ and $y$. Specifically, to make the final decision on the relation of two input concepts $(x, y)$, we solve the objective function defined in Eq. 6.

$$\begin{aligned} \ell^* &= \text{argmax}_\ell \frac{1}{|\mathcal{T}_\ell|} \sum_{t\in\mathcal{T}_\ell} \lambda_t score(t) \quad (6)\\ &= \text{argmax}_\ell \frac{1}{|\mathcal{T}_\ell|} \sum_{t\in\mathcal{T}_\ell} \lambda_t \left( \sum_{e\in t} w(e) - \sum_{k=1}^{|\mathcal{C}|} \rho_k d_{C_k}(t) \right) \end{aligned}$$

where

$$\lambda_t = \frac{\#\text{ of occurrence of } t}{\#\text{ of predicted concept networks in training data}} \quad (7)$$

is the occurrence probability of concept network $t$ in the training data with additional concepts. Note that, to measure $\lambda_t$, a local relation classifier is used to predict the best relation between concepts in the concept networks in the training data.

## 4.2 Constraint Selection

In Eqn. (6), a list of relational constraints $\mathcal{C}$ is used. Recall that a relational constraint is an invalid concept network that is not allow to happen. In our work, relational constraints are mined from the training data by applying *forward feature selection* technique (CITATIONS XXXX YYY). The algorithm starts with an empty set of constraints $\mathcal{C}$, then grows the constraint set gradually by adding to the set the *best* concept network in each iteration. The *best* concept network is defined as the constraint used in Eqn. (6) that helps improve the system's performance most. Our *forward constraint selection* algorithm is described in Fig. 3.

```
Algorithm FORWARD CONSTRAINT SELECTION
    INPUT: Data set with related concepts D = ⟨(x, y, ℓ, Z*)⟩
        A trained local relation classifier L.
        A list of concept networks T = {t_1, t_2, ..., t_m};
    OUTPUT: Constraint set C.

    C = ∅;
    bestAcc = evaluate(D, L, C);
    isIncreasing = true;

    While (isIncreasing) do
        isIncreasing = false;
        For each t ∈ T do
            C = C ∪ {t};
            acc = evaluates(D, L, C);
            If (acc > bestAcc) then
                t* = t;
                bestAcc = acc;
                isIncreasing = true;
            End if
            C = C\{t};
        End for
        If (isIncreasing) then
            C = C ∪ {t*};
            T = T\{t*};
        End if
    End while

    RETURN: C;
```

**Figure 3: Forward constraint selection algorithm. The *evaluate* function evaluates all examples from the input data set with related concepts using a trained local classifier $\mathcal{L}$ and constraint set $\mathcal{C}$ by solving the objective function in Eq. (6).**

One of the inputs of the algorithm, $\mathcal{T}$, is a set of $m$ concept networks. At each iteration, all concept networks $t \in \mathcal{T}$ are tried; and at the end of the iteration, the *best* network is added into $\mathcal{C}$, and also removed from $\mathcal{T}$. In this paper, for simplicity, we only used 3-vertex concept networks, thus, $m = 64$. However, the general case of $n$-vertex concept networks can also be used with the our model.

## 4.3 Related Concepts Extraction

The input of our problem is a pair of two concepts. To apply our proposed constraint-based inference model, we need to acquire additional concepts to the input pair. It can be argued that, with strong relational constraints, one can use random concepts as additional concepts to do the inference. However, with a high chance, a random additional concept simply get *no relation* from two input concepts. This will not help much in the inference model because there are many relational constraints with relations other than *no relation*. To address this issue, we investigate different approaches to obtain additional concepts which are related to input con-

cepts. From now on, we refer to additional concepts as *related concepts*.

In the first direction, as the first step to verify the correctness of our proposed constraint-based inference model, related concepts are added manually for input concepts. By using *gold related concepts*, a significant improvement on the system's performance is necessary to prove the correctness of the inference model. Our experiments (see Sec. 5.2) indeed show that the proposed inference model is correct and efficient.

However, in real world applications, *gold related concepts* are not available. We, therefore, propose an approach that makes use of YAGO ontology [23] to provide related concepts. It is worth to note 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. YAGO builds on entities and relations and currently contains millions of entities and facts. All entities and relations in YAGO are extracted from Wikipedia category system and WordNet. In YAGO model, all objects (e.g. *cities*, *people*, etc.) are represented as *entities* [2]. Follow YAGO model, our input concepts are considered as 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 ancestor of an *entity* by using TYPE relation which gives us the entity's *classes*. By using two relations MEANS and TYPE in YAGO model, we can obtain direct ancestors, siblings, and also children of an input concept using the following query patterns.

YAGO QUERY PATTERNS
INPUT: concept "$X$"
OUTPUT: lists of ancestors, siblings, and children of "$X$"

PATTERN 1:
"$X$" MEANS `?A`
`?A` TYPE `?B`
`?C` TYPE `?B`

PATTERN 2:
"$X$" MEANS `?D`
`?E` TYPE `?D`

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

**Figure 4: Query patterns used in YAGO to obtain lists of direct ancestors, siblings, and children of input concept "$X$".**

In Fig. 4, PATTERN 1 is used to get lists of ancestors and siblings of a given concept. For example, with concept "$X$"= "honda civic", PATTERN 1 returns lists of ancestors {wikicategory_1970s_automobiles, wordnet_car_102958343, wikicategory_Honda_vehicles, ...}, and siblings {Ford_Capri, Mercury_Comet, Honda_Jazz, Honda_Accord, Volkswagen_Karmann_Ghia, Pontiac_Vibe, ...}. The prefix and suffix annotation of the ancestors are dropped. In the case that ancestors are noun phrases, we only use the head of the noun phrases. We use the Noun Group parser from (CITATIONS XXX YYY) to extract the *head* of a noun phrase. In the other hand, PATTERN 2 returns an empty list of children of "honda civic", because entity `Honda_Civic` is at the deepest level in YAGO ontology. The list of children will be non-empty if "$X$" is some generic concept such as "*actor*", "*president*", etc.

[2]For more details about YAGO, please see [23]

# 5. EXPERIMENTAL STUDY

In this section, we first describe the data sets used in our experimental study. After that, we present the experiments to evaluate our approach at different levels: our local classifier only, local classifier pluses going beyond Wikipedia, and pluses constrant-based inference model with related concepts from gold data and YAGO ontology. We then compare our results with other related work, and also do several experimental analyses.

## 5.1 Datasets

Our approach to identifying relational knowledge is evaluated by using a dataset of 40 semantic classes of almost 11,000 instances, which was used in [14]. This dataset was manually constructed[3] and was used to evaluate many information extraction tasks [14, 15]. Each semantic class is an incomplete set of representative instances, and has about 272 instances in average. The smallest class is *search engine* with 25 instances, and the largest class is *actor* with 1500 instances. Table 2 shows a snippet of the dataset. Interestingly, 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*, *Sharon Stone*), and also classes with common nouns (e.g. *basic food* with *rice*, *milk*, *eggs*). This fact helps evaluate the ability of systems that can deal with not only concepts like name entities, but also common instances.

| Semantic class (Size) | Examples of Instances |
|---|---|
| basic food (155) | rice, milk, eggs, beans, fish |
| chemical element (118) | lead, copper, aluminum, calcium |
| city (589) | San Francisco, Dubai, Chicago |
| disease (209) | arthritis, hypertension, influenza |
| actor (1500) | Kate Hudson, Mel Gibson |

**Table 2: A snippet of 40 semantic classes with instances. The class names in the original dataset (*basicfood*, *chemicalelem*) were presented in a meaningful form as shown in the left column.**

In the original dataset, the semantic class names are not often written in a meaningful form, such as *chemicalelem*, *proglanguage*, and *worldwarbattle*. These names are not usable for our system to find corresponding concepts in Wikipedia. In our experiments, each semantic class name in the original dataset is expanded to meaningful forms. For example, *terroristgroup* is expanded to *terrorist group*, *terrorrist*, *torrorism*. The expansion is kept to be minimum. Moreover, we also use these expansions for other systems to compare to our approach in the experiments.

An example in our learning problem is a pair of two concepts ($X$, $Y$) such as (*city*, *Dubai*), (*lead*, *aluminum*). Note that in this paper, we refer to both the name of the semantic classes and their instances as concepts, such as *city* and *Dubai*. We pair the semantic classes and instances in the original data set to create training and testing examples.

[3]Private communication with Marius Pasca, 2009.

The examples cover all types of relational knowledge of interest: $X \leftarrow Y$, $X \rightarrow Y$, $X \leftrightarrow Y$, and $X \nleftrightarrow Y$. Specifically, examples are created with the following guidelines.

- $X \leftarrow Y$ examples: For each semantic class, we pair the name of the class with its instances. These examples have the genearl form (*semantic class X*, *child of X*).
- $X \rightarrow Y$ examples: These examples have the general form (*concept X*, *semantic class of X*).
- $X \leftrightarrow Y$ examples: The general form of these examples is (*concept X*, *concept Y*), where $X$ and $Y$ are two instances of a semantic class, and $X \neq Y$.
- $X \nleftrightarrow Y$ examples: To make examples with two concepts having no relation, we pair either a semantic class name and an instance of another semantic class (and vice versa), or an instance in a semantic class and another instance in other classes.

Tab. 3 shows some examples created from the original data set.

| 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 3: Some examples in our data set.**

We randomly create $20,000$ examples following the guidelines above. From these examples, we use $8,000$ examples for the training set, and $12,000$ examples for the test set. We refer to the 12,000-example test set as the *TestAll* test set.

From the training set, we discard examples with one or both concepts not in Wikipedia (*non-Wikipedia examples*). By using our concept disambiguation algorithm (see Fig. 1), we can discover *non-Wikipedia* examples by seeing if the algorithm retrieves no relevant articles for either one or both concepts in the examples. This results in a training set of $6,959$ examples. It is important to note that, we do not discard *non-Wikipedia* examples in the *TestAll* test set. Table 4 shows the statistic of the training and testing data with the number of examples in each relation class.

| Data | $X \leftarrow Y$ | $X \rightarrow Y$ | $X \leftrightarrow Y$ | $X \nleftrightarrow Y$ | Total |
|---|---|---|---|---|---|
| *Training* | 1,739 | 1,754 | 1,664 | 1,802 | 6,959 |
| *TestAll* | 3,045 | 3,025 | 2,965 | 2,965 | 12,000 |

**Table 4: Details of the training and test sets with the number of examples in each relation class. The *Training* set containts on examples in Wikipedia. *TestAll* includes *non-Wikipedia* examples.**

To evaluate our system, we use a snapshot of Wikipedia in July, 2008. We first clean up articles in Wikipedia and remove articles that are not of interest. The removed articles include articles without a category, except the redirect pages, or articles with useless categories such as *Protected redirects*. We also remove administrative articles including *Wikipedia* pages, *Template* pages, *Image* pages, and *Portal* pages. Furthermore, we do not use articles without titles. After the pre-processing step, we have 5,503,763 articles left. We index the articles using the Apache Lucene Information Retrieval library[4]. Lucene is a high-performance text search library that is written in Java and is a widely used off-the-shelf IR system.

## 5.2 Experimental Results

We are going to evaluate our system at three different levels. The first level evaluate our local relation classifier described in Sec. 3. We refer to the first level as *Local Classifier*. In the second level, we apply our method presented in Sec. 3.3 to finds replacements for concepts that are not in Wikipedia. The results for this level is presented under the name *+BeyondWiki*. After that, for the third level of evaluation, we use the approach described in Sec. 4 to extract related concepts for input examples from YAGO ontology to evaluate our constraint-based inference model. We use the name *+Inference (YAGO)* to refer to the third level experiment. Note that, *+Inference (YAGO)* is evaluated on top of the results of *Local Classifier* and *+BeyondWiki*. *+Inference (YAGO)* shows our final results on the identification problem of relational knowledge.

We also report the performance of our system in the third level of evaluation using the related concepts added manually. This experiment, as mentioned in Sec. 4.3, shows the correctness of our inference model. To have gold related concepts, we take advantage of our original data set described in Sec. 5.1. The original data set provide us ancestor and sibling concepts for basic concepts such as *Chicago*, *hypertension*, etc., and children for generic concepts such as *chemical element*, *actor*, etc. We refer to this experiment as *+Inference (Gold)* which is also evaluated on top of *Local Classifier* and *+BeyondWiki*. Furthermore, in this experiment, for each input concept, we use 1 ancestors, 5 siblings, and 5 children as its related concepts (i.e. 22 related concepts for an example) for both *+Inference (YAGO)* and *+Inference (Gold)*.

In our system, we vary the value of $K$ as the number of levels that one goes up on the Wikipedia category system to extract features (see Sec. 3) for input concept pairs. In our experiments, $K$ is set from 0 to 4. We train our classifier on the *Train* training set and evaluate on the *TestAll* test set described in table 4. We evaluate performance of the systems by calculating the accuracy of identification of relation between concepts in pairs. The accuracy is calculated by the percentage of the number of correct prediction over the total number of examples used in testing. All of the results report in our experiments are measure by accuracy which is the percentage of the number of correct prediction over the total number of examples in the test set. Table 5 summarizes the performance of our system at different levels of evaluation.

First of all, from Tab. 5, we see that without using the Wikipedia category system ($K = 0$), we get very poor performance. The situation changes significantly when the Wikipedia category system is used. The best performance

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

| $K$ | *Local Classifier* Accuracy | *+BeyondWiki* Accuracy | *+Inference (YAGO)* | | *+Inference (Gold)* | |
|---|---|---|---|---|---|---|
| | | | Accuracy | Error Reduction | Accuracy | Error Reduction |
| 0 | 37.69 | 37.99 | 37.99 | - | 37.99 | - |
| 1 | 79.37 | 80.62 | 82.13 | 13.38 | 84.57 | 25.21 |
| 2 | 81.89 | 83.58 | 84.79 | 16.01 | 86.52 | 25.57 |
| 3 | 81.38 | 83.14 | 85.07 | 19.82 | 85.44 | 21.80 |
| 4 | 80.3 | 82.0 | 84.08 | 19.19 | 84.42 | 20.91 |

**Table 5: Performances in accuracy of our system, evaluated on *TestAll* test set. $K$ is the number of levels one goes up on the Wikipedia category system to extract features for input concepts. With $K = 0$, no constraints are mined, the inference model is not used. '-' means the error reduction is too small.**

of the local relation classifier is 81.89% in accuracy. By using our approach to find replacements for *non-Wikipedia* concepts (going beyond Wikipedia), we get remarkable improvement with 83.58% of accuracy as the best performance. However, with $K = 0$, the performance almost stay the same. One of the reasons for this phenomenon is that although *+BeyondWik* works well, we cannot expect to get good result with a poor local classifier at $K = 0$.

Significant improvement of the system's performance compare to *Local Classifier* is achieved when we use our constraint-base inference model to predict concept relations. Except when $K = 0$, all other values of $K$ gain significant error reduction. The best performance of our system is 85.07% with $K = 3$ when applying our inference model on top of *Local Classifier +BeyondWiki*, gaining 19.82% of error reduction. All inference processes in this experiment use 22 related concepts for each input concept pair. With $K = 0$, the *forward constraint selection* algorithm returns no constraint, therefore, the inference process does not help improving the performance of the system.

On the other hand, the performance of *+Inference (Gold)* shows that our constraint-based inference model using relational constraints is correct and effective. The system using *+Inference (Gold)* achieves the best performance – 86.52% of accuracy, gains 25.57% of error reduction. However, in real world problem, the gold related concepts are not available. Therefore, we propose the approach using YAGO ontology to extract related concepts for the inference process to make prediction for real world input concept pairs. Especially, the result of the *+Inference (Gold)* experiment shows that the more relevant the related concepts are, the better performance our system achieves.

Table 6 shows some output examples of our system from the best setting (*Local Classifier +BeyondWiki +Inference(YAGO)*). Example #1 shows the case where all levels of prediction are correct. In #2, the local classifier makes wrong prediction, but then is corrected by *+BeyondWiki* because *+BeyondWiki* finds a good replacement for non-Wikipedia concept *bartlomiej strobel*. In examples #3 and #4, both *Local Classifier* and *+BeyondWiki* make wrong prediction, but *+Inference (YAGO)* gets it right. Example #5 shows an interesting case where the local classifier itself, by some chance, makes correct prediction, but *+BeyondWiki*, after finding a replacement for *harald slott-moller*, makes a wrong decision. However, *+Inference (YAGO)*, with its power of internal reasoning, gets back to the right answer. In exmaple #6, both *Local Classifier* and *+BeyondWiki* make correct prediction, but *+Inference (YAGO)* gets it wrong. This is because of bad related concepts extracted for the two input concepts.

## 5.3 Comparison to Prior Work

To our best knowledge, there is no prior work targets directly the problem of identifying relational knowledge defined in this paper. Most of prior work which relate to our problem focus on building lexical taxonomy or knowledge ontology [23, 16]. With some effort of searching the knowledge base, one can get the answer for relationship between input concepts. We compare our system with three other systems which are built on different existing resources.

1. Strube 07 uses a large scale taxonomy which was derived from Wikipedia [16], 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. As a result, the taxonomy contains a large amount of subsumption, i.e. *isa*, relations [16]. Given an input with two concepts $X$ and $Y$, using the taxonomy, $X$ is an ancestor of $Y$ if one of the articles about $Y$ is subsumed by an article about $Y$, using *isa* links of articles, up to $K$ levels in the taxonomy. Similarly, for the case that $Y$ is an ancestor of $X$. If $X$ and $Y$ share a common ancestor within $K$ levels in the taxonomy, they are considered as siblings. We first apply our concept disambiguation algorithm on given concepts and then mount them onto the taxonomy to infer the relations. The taxonomy used in this experiment is in the latest version of March, 2008[5].

2. Snow 06 uses the *extended WordNet* [21, 22] as the background knowledge. The authors of the extended WordNet [21] first identified lexico-syntactic patterns indicative of hypernymy from corpora. These patterns were used to extract candidate noun pairs that may hold the hypernym relation. A trained classifier is applied on these noun pairs to recognize the pairs holding hypernym relation. Starting from WordNet-2.1 [6], the latest version of the extended WordNet has augmented 400,000 synsets. Words that are added into the extended WordNet can be common nouns or proper nouns. The extended WordNet can serve as the very good background knowledge to identify relational knowledge of interest by looking for the input concepts in the extended WordNet tree for all possible senses of the concepts and then infer their relationship. We also vary the value of $K$ as the number of levels one goes

[5]Private communication with Michael Strube and Simone Paolo Ponzetto, 2009.

| No. | $X$ | $Y$ | True Label | *Local Classifier* Prediction | *+BeyondWiki* | | | *+Inference (YAGO)* Prediction |
|---|---|---|---|---|---|---|---|---|
| | | | | | Replacement | | Prediction | |
| | | | | | $X'$ | $Y'$ | | |
| 1 | city | lisbon | $X \leftarrow Y$ | $X \leftarrow Y$ | - | - | $X \leftarrow Y$ | $X \leftarrow Y$ |
| 2 | *bartlomiej strobel* | painter | $X \rightarrow Y$ | $X \leftrightarrow Y$ | drew struzan | - | $X \rightarrow Y$ | $X \rightarrow Y$ |
| 3 | taiwan | singapore | $X \leftrightarrow Y$ | $X \leftarrow Y$ | - | - | $X \leftarrow Y$ | $X \leftrightarrow Y$ |
| 4 | *southern herald* | newspaper | $X \rightarrow Y$ | $X \leftrightarrow Y$ | liberty | - | $X \leftrightarrow Y$ | $X \rightarrow Y$ |
| 5 | *harald slott-moller* | jean ingres | $X \leftrightarrow Y$ | $X \leftrightarrow Y$ | george inness | - | $X \nleftrightarrow Y$ | $X \leftrightarrow Y$ |
| 6 | somalia | hurricane | $X \nleftrightarrow Y$ | $X \nleftrightarrow Y$ | - | - | $X \nleftrightarrow Y$ | $X \leftrightarrow Y$ |

**Table 6: Relationship prediction examples from *TestAll* test set. Concepts in *italic font* are not in Wikipedia, and replaced by corresponding concepts in *+BeyondWiki*.**

up on the WordNet tree to find common subsumption concepts.

3. YAGO 07 uses YAGO ontology [23] as the main source of background knowledge. Because YAGO ontology is a combination of Wikipedia and WordNet (see our brief description in Sec. 4.3), this system is expected to be very powerful in recognizing concept relationship. To access a concept's ancestors and siblings, we combine PATTERN 1 in Fig. 4 and the SUBCLASSOF relation in YAGO model to go up on the ontology. The SUBCLASSOF relation can be cascaded to allow one go up to $K$ levels on the hierarchical structure of the ontology.

Our system, OURS, used to compare with other systems is the best system using constrant-based inference model with related concepts extracted from YAGO ontology (*Local Classifier +BeyondWiki +Inference (YAGO)*). Table 7 shows the comparison of the systems in details.

| $K$ | STRUBE 07 | SNOW 06 | YAGO 07 | OURS |
|---|---|---|---|---|
| 0 | 23.78 | 24.71 | **50.08** | 37.99 |
| 1 | 23.83 | 40.24 | 64.43 | **82.13** |
| 2 | 23.84 | 42.63 | 63.94 | **84.79** |
| 3 | 23.88 | 40.96 | 62.02 | **85.07** |
| 4 | 24.32 | 40.65 | 60.57 | **84.08** |

**Table 7: Our system's performance compared to other systems. Performances are measure by accuracy. The *TestAll* test set is used in this experiment.**

It is showed clearly in Tab. 7 that our system significantly outperforms other systems implemented by using existing sophisticated resources. Our approach also utilizes Wikipedia as the main background knowledge. However, we are superior to other systems because we employ advance machine learning techniques along with a powerful and effective contraint-based inference model.

We, furthermore, evaluate and compare the systems on different derived data sets from the *TestAll* test set. The first derived data set is *TestWiki* containing only concepts in Wikipedia. The second derived data set is *TestWn* containing only concepts in the *extended WordNet* [21, 22]. This experiment provides a fair comparison for systems that use main resources built on either Wikipedia or the *extended WordNet*, or both Wikipedia and WordNet. By removing all examples with either concept in Wikipedia from *TestAll*, there are 10,456 examples left in *TestWiki*. Similarly, *TestWn* has 8,625 examples after dropping from *TestAll* 3,375 examples having at least one concept not in the *extended WordNet*. We only report the best result that each system achieves on corresponding test set. We use our best model (*Local Classifier +BeyondWiki +Inference (YAGO)* in this experiment. When evaluated on the *TestWiki* test set, our model becomes (*Local Classifier +Inference*) because we do not need to use *+BeyondWiki*. The results of this experiment are showed in Tab. 8.

| | STRUBE 07 | SNOW 06 | YAGO 07 | OURS |
|---|---|---|---|---|
| *TestWiki* | 24.59 | 44.34 | 70.29 | **90.83** |
| *TestWn* | 24.13 | 47.79 | 70.81 | **90.76** |

**Table 8: Comparing systems' performance with different test sets derived from the *TestAll* test set. The best performance of each system (by varying $K$) is reported.**

## 5.4 Experimental Analysis

In this section, we give several experimental analyses on our system. These analyses provide a deeper understanding of different aspects of the algorithms and models in the system.

### 5.4.1 Concept Disambiguation

To present the performance of our concept disambiguation algorithm, we pick 3 semantic classes which represent different levels of ambiguity.

These three classes are *England Football Clubs*, *Superheroes*, and *Rivers in England*. We choose these three classes because they have many ambiguous concepts. For example, the concept *Chelsea* may refer to places such as a railway station in London (*Chelsea tube station*), a city in Massachusetts in the United States (*Chelsea, Massachusetts*), and, of course, a sport organization as in *Chelsea Football Club*. In general, *football clubs* are often named after the name of the city where it resides, *superheroes* are named with random names which can match with many other things such as *Thunderbird*, *Tiny*, and *Speedy*; similarly, rivers may be named with ambiguous names such as *Burn* and *Hun*.

For *England Football Clubs*, we create 40 concept pairs in the form $\langle football, X\rangle$, and also 40 concept pairs in the form $\langle X, Y\rangle$, where $X$, $Y$ are names of football clubs, and $X \neq Y$. Similarly, with the *Superheroes* class, we generate 40 pairs of $\langle superhero, X\rangle$, and 40 pairs of $\langle X, Y\rangle$), where

$X, Y$ are names of superheroes, and $X \neq Y$. For the *Rivers in England*, we have only 34 instances (used in [25]), we use all of them to make pairs of $\langle river, X\rangle$, and $\langle X, Y\rangle$. In total, there are 80, 80, and 68 pairs for *Football Clubs*, *Superheroes*, and *Rivers*, respectively.

We compare our algorithm (OURS) with a search performed on two input concepts separately (SEPSCH). The SEPSCH method only searches titles of articles in Wikipedia to get as relevant articles as possible.

| | *Football Clubs* | | *Superheroes* | | *Rivers* | |
|---|---|---|---|---|---|---|
| | T5 | T10 | T5 | T10 | T5 | T10 |
| SEPSCH | 95.0 | 96.25 | 60.0 | 65.0 | 75.0 | 79.41 |
| OURS | 100 | 100 | 81.25 | 81.25 | 77.94 | 77.94 |

**Table 9: Performance of concept disambiguation methods. T5 and T10 corresponds to examining and evaluating top 5 and 10 articles retrieved. SEPSCH refers to the search performed on the two concepts separately. OURS is the algorithm in Fig. 1.**

Tab. 9 presents the experimental results. We evaluate the performances by manually examining and evaluating top 5 (T5) and top 10 (T10) Wikipedia articles retrieved by the two compared algorithms. As showed in Tab. 9, our algorithm outperforms the SEPSCH method about 10% with top 5 and 6% with top 10 in average accuracy. It worth noting that our algorithm produces no different result for T5 and T10 lists. This shows that our algorithm is effective in retrieving relevant article to the input concepts by putting relevant articles to very top in the relevant list if the input concepts can be found in Wikipedia. In the case of *River*, the fact that two river names are not usually appear in the same article possibly makes our algorithm return unexpected articles. This explains why we do not get very reasonable results for *River*.

### 5.4.2 Performance on Individual Semantic Class

We study our system by analyzing its performance on individual semantic class. This experiment shows the performance of our system on our 4 concept relation classes including $X \leftarrow Y$, $X \rightarrow Y$, $X \leftrightarrow Y$, and $X \nleftrightarrow Y$. There are 40 semantic classes in the datasets. Recall that examples in our datasets are pairs of concepts. An example is counted to a semantic class if at least one concept of the example belongs to that class. We use the *TestAll* test set in this experiment. Our best system (*Local Classifier +BeyondWiki +Inference (YAGO)*) is used. The performance of the classifier is measured in accuracy which is the portion of correct prediction over the total number of examples. Table 10 presents the performance of our classifier on individual semantic class, and also the average accuracy on each relation class.

From the last row in the table, the average accuracies of the classifier on relations $X \leftarrow Y$ and $X \rightarrow Y$ are roughly the same. This essentially occurs because these two classes are symmetric. The performance of the system on sibling relation is lower than other relations. This can be explained by the fact that disambiguating two sibling concepts is not a trivial problem. As we discussed above, two sibling concepts such as two *rivers* are not usually mentioned in the same article. This fact may cause the disambiguation algorithm to retrieve unexpected articles associating with two input concepts, and eventually screw up the final prediction.

| Semantic class | $X \leftarrow Y$ | $X \rightarrow Y$ | $X \leftrightarrow Y$ | $X \nleftrightarrow Y$ | Average |
|---|---|---|---|---|---|
| searchengine | 100 | 100 | 100 | 94.12 | 98.53 |
| nbateam | 100 | 100 | 100 | 90.37 | 97.59 |
| stadium | 100 | 98.39 | 98.77 | 92.95 | 97.53 |
| nationalpark | 100 | 100 | 100 | 88.32 | 97.08 |
| videogame | 97.73 | 98.0 | 98.91 | 89.8 | 96.11 |
| soccerclub | 100 | 100 | 100 | 82.39 | 95.60 |
| chemicalelem | 100 | 100 | 90.48 | 88.28 | 94.69 |
| carmodel | 95.08 | 96.12 | 96.15 | 90.84 | 94.55 |
| actor | 96.53 | 95.98 | 95.79 | 88.72 | 94.25 |
| hurricane | 100 | 100 | 82.61 | 90.41 | 93.25 |
| cartooncharacter | 92.86 | 100 | 96.15 | 83.92 | 93.23 |
| cellphonemodel | 94.64 | 86.44 | 95.4 | 92.03 | 92.13 |
| movie | 95.43 | 98.25 | 95.74 | 78.57 | 92.00 |
| worldwarbattle | 92.31 | 100 | 84.29 | 90.84 | 91.86 |
| university | 97.69 | 97.83 | 86.96 | 83.45 | 91.48 |
| holiday | 91.3 | 100 | 88.68 | 85.0 | 91.25 |
| skyscraper | 93.48 | 96.08 | 82.76 | 91.6 | 90.98 |
| mountain | 96.67 | 94.37 | 84.93 | 86.58 | 90.64 |
| painter | 91.58 | 90.16 | 83.91 | 92.81 | 89.61 |
| disease | 85.25 | 93.24 | 80.28 | 95.89 | 88.67 |
| wine | 100 | 89.47 | 78.12 | 86.76 | 88.59 |
| terroristgroup | 100 | 100 | 70.0 | 84.0 | 88.50 |
| proglanguage | 93.55 | 87.5 | 79.01 | 92.72 | 88.19 |
| sportevent | 100 | 100 | 64.89 | 87.01 | 87.97 |
| skybody | 90.91 | 93.1 | 75.0 | 92.62 | 87.91 |
| treaty | 88.06 | 86.89 | 88.1 | 88.37 | 87.86 |
| city | 96.49 | 98.77 | 77.78 | 69.66 | 85.67 |
| newspaper | 77.36 | 80.0 | 89.58 | 93.89 | 85.21 |
| country | 92.98 | 96.72 | 85.54 | 62.07 | 84.33 |
| award | 92.45 | 90.74 | 63.95 | 90.0 | 84.28 |
| religion | 92.0 | 86.11 | 68.6 | 83.33 | 82.51 |
| river | 82.76 | 97.5 | 57. | 86.11 | 80.84 |
| company | 90.5 | 92.73 | 61.25 | 76.92 | 80.35 |
| drug | 77.78 | 78.85 | 67.37 | 94.27 | 79.57 |
| flower | 85.0 | 85.0 | 58.62 | 84.5 | 78.28 |
| currency | 90.91 | 80.0 | 46.43 | 91.04 | 77.09 |
| aircraftmodel | 66.67 | 70.49 | 81.03 | 88.6 | 76.70 |
| basicfood | 77.27 | 86.67 | 50.0 | 87.92 | 75.47 |
| empire | 80.0 | 69.57 | 56.6 | 81.82 | 72.00 |
| digitalcamera | 41.6 | 47.1 | 70.45 | 85.11 | 61.06 |
| Average | 90.92 | 91.55 | 80.78 | 87.09 | |

**Table 10: Performance of our best system on individual semantic class. An example is counted to a semantic class if at least one concept of the example belongs to that class. The semantic classes are presented in a decreasing order of its average accuracy of all concept relation classes. The average accuracy of each relation class evaluated on all 40 semantic classes is showed in the last row. This experiment is done with $K = 3$.**

### 5.4.3 Effect of the Number of Related Concept

In this experiment, we study the performance of our system with different numbers of related concepts used in our constraint-based inference model. The related concepts includes the ancestors, siblings, and children of an input concept. For simplicity, in this experiment, we only report the total number of related concepts extracted for two concept of an input example. This experiment is done with the best model of *+Inference (Gold)* on the *TestAll* test set. Fig. 5 illustrates the improvement of the system's performance when the number of related concepts used in the inference model increases.

From the graph we see that the improvement keeps increasing when more related concepts are used. Due to the time limitation, we stop at 22 related concepts. However, one can expected to get better performance if more related concepts are used.

## 6. CONCLUSION

Textual inference problems that often arise in data and knowledge management problems necessitate some level of "common sense" inference that relies of identifying relations such as *ancestor* and *sibling* among concepts.

We have developed a robust and accurate machine learn-

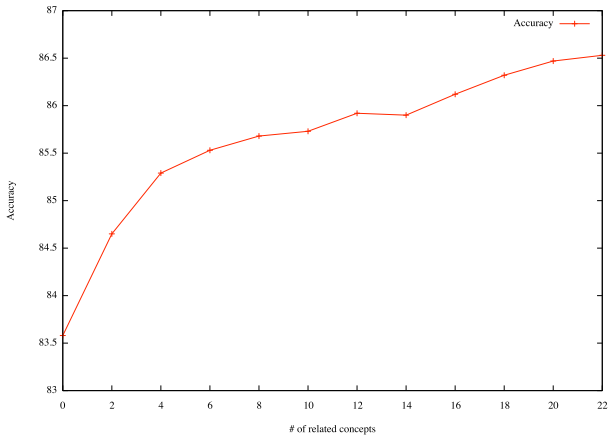


**Figure 5: Relation between system's performance and number of related concepts used in the constraint-based inference model. This experiment is done with the best model of *+Inference (Gold)* on the *TestAll* test set.**

ing approach to this problem, that is based using a very small number of annotated examples. The results presented show that our approach generalizes well across semantic classes and, even though we use Wikipedia as the key knowledge source, it can handle well also concepts that are not mentioned in Wikipedia. We showed that our approach has significantly better coverage and accuracy than existing knowledge acquisition and relation identification approaches. While this study focused on developing and analyzing our approach, our immediate future step will be the evaluation of it in the context of supporting textual inference applications.

## 7. REFERENCES


[1] M. Banko, M. Cafarella, M. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, 2007.

[2] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[3] R. Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1678–1679, 2005.

[4] I. Dagan, O. Glickman, and B. Magnini, editors. *The PASCAL Recognising Textual Entailment Challenge.*, volume 3944. Springer-Verlag, Berlin, 2006.

[5] D. Davidov and A. Rappoport. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proceedings of ACL-08: HLT*, pages 692–700, Columbus, Ohio, June 2008. Association for Computational Linguistics.

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

[7] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[8] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.

[9] A. Haghighi, A. Ng, and C. Manning. Robust textual inference via graph matching. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 387–394, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[10] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[11] Z. Kozareva, E. Riloff, and E. Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[12] B. Maccartney and C. D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

[13] M. Mohler and R. Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 567–575, Athens, Greece, March 2009. Association for Computational Linguistics.

[14] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 101–110, New York, NY, USA, 2007. ACM Press.

[15] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08: HLT*, pages 19–27, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[16] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. *AAAI-07*, 2007.

[17] W. W. C. Richard C. Wang. Automatic set instance extraction using the web. In *Proceedings of ACL-09: IJCNLP*, 2009.

[18] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In H. T. Ng and E. Riloff, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics, 2004.

[19] L. Sarmento, V. Jijkuon, M. de Rijke, and E. Oliveira. "more like these": growing entity classes from seeds. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 959–962, New York, NY, USA, 2007. ACM.

[20] S. Sekine. On-demand information extraction. In *Proc.*

*of the Annual Meeting of the ACL*, pages 731–738, 2006.

[21] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2005)*, November 2005. This is a draft version from the NIPS preproceedings; the final version will be published by April 2005.

[22] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[23] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[24] P. P. Talukdar, T. Brants, M. Liberman, and F. Pereira. A context pattern induction method for named entity extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 141–148, New York City, June 2006. Association for Computational Linguistics.

[25] P. P. Vishnu Vyas. Semi-automatic entity set refinement. In *Proceedings of NAACL-09*, 2009.

[26] R. Wang and W. Cohen. Language-independent set expansion of named entities using the web. pages 342–350, Oct. 2007.

[27] R. Wang and W. Cohen. Iterative set expansion of named entities using the web. pages 1091–1096, Dec. 2008.

[28] Z. Zhang. Weakly-supervised relation classification for information extraction. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588, New York, NY, USA, 2004. ACM Press.