Automatically Refining the Wikipedia Infobox Ontology

Fei Wu
Computer Science & Engineering Department,
University of Washington, Seattle, WA, USA
wufei@cs.washington.edu

Daniel S. Weld
Computer Science & Engineering Department,
University of Washington, Seattle, WA, USA
weld@cs.washington.edu

ABSTRACT

The combined efforts of human volunteers have recently extracted numerous facts from Wikipedia, storing them as machine-harvestable object-attribute-value triples in Wikipedia infoboxes. Machine learning systems, such as Kylin, use these infoboxes as training data, accurately extracting even more semantic knowledge from natural language text. But in order to realize the full power of this information, it must be situated in a cleanly-structured ontology. This paper introduces KOG, an autonomous system for refining Wikipedia's infobox-class ontology towards this end. We cast the problem of ontology refinement as a machine learning problem and solve it using both SVMs and a more powerful joint-inference approach expressed in Markov Logic Networks. We present experiments demonstrating the superiority of the joint-inference approach and evaluating other aspects of our system. Using these techniques, we build a rich ontology, integrating Wikipedia's infobox-class schemata with WordNet. We demonstrate how the resulting ontology may be used to enhance Wikipedia with improved query processing and other features.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous

Keywords

Semantic Web, Ontology, Wikipedia, Markov Logic Networks

1. INTRODUCTION

The vision of a Semantic Web will only be realized when there is a much greater volume of structured data available to power advanced applications. Given the recent progress in information extraction, it may be feasible to automatically gather this information from the Web, using machine learning trained extractors. Wikipedia, one of the world's most popular Websites[1], is a logical source for extraction, since it is both comprehensive and high-quality. Indeed, collaborative editing by myriad users has already resulted in the creation of *infoboxes*, a set of subject-attribute-value triples summarizing the key aspects of the article's subject, for numerous articles. DBpedia [5] has aggregated this infobox data, yielding over 15 million pieces of information.

Furthermore, one may use this infobox data to bootstrap a process for generating additional structured data from Wikipedia. For example, our autonomous Kylin system [35] trained machine-learning algorithms on the infobox data, yielding extractors which can accurately[2] generate infoboxes for articles which don't yet have them. We estimate that this approach can add over 10 million additional facts to those already incorporated into DBpedia. By running the learned extractors on a wider range of Web text and validating with statistical tests (as pioneered in the KnowItAll system [16]), one could gather even more structured data.

In order to effectively exploit extracted data, however, the triples must be organized using a clean and consistent ontology. Unfortunately, while Wikipedia has a category system for articles, the facility is noisy, redundant, incomplete, inconsistent and of very limited value for our purposes. Better taxonomies exist, of course, such as WordNet [1], but these don't have the rich attribute structure found in Wikipedia.

1.1 KOG: Refining the Wikipedia Ontology

This paper presents the Kylin Ontology Generator (KOG), an autonomous system that builds a rich ontology by combining Wikipedia infoboxes with WordNet using statistical-relational learning. Each infobox template is treated as a class, and the slots of the template are considered as attributes/slots. Applying a Markov Logic Networks (MLNs) model [28], KOG uses joint inference to predict subsumption relationships between infobox classes while simultaneously mapping the classes to WordNet nodes. KOG also maps attributes between related classes, allowing property inheritance.

1.2 Why a Refined Ontology is Important

Situating extracted facts in an ontology has several benefits.

Advanced Query Capability: One of the main advantages of extracting structured data from Wikipedia's raw text is the ability to go beyond keyword queries and ask SQL-like questions such as "What scientists born before 1920 won the Nobel prize?" An ontology can greatly increase the recall of such queries by supporting transitivity and other types of inference. For example, without recognizing that particle physicist is a subclass of physicist which is itself a subclass of scientists, a Wikipedia question-answering system would fail to return "Arthur Compton" in response to the question above. In many cases the attributes of different Wikipedia infobox classes are mismatched, for example one infobox class might have a "birth place" attribute while another has "cityofbirth" — matching corresponding attributes for subclasses is clearly essential for high recall.

Faceted Browsing: When referring to Wikipedia, readers use a mixture of search and browsing. A clear taxonomy and aligned

[1]Ranked 8th in January 2008 according to comScore World Metrix.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

[2]Kylin's precision ranges from mid-70s to high-90s percent, depending on the attribute type and infobox class.

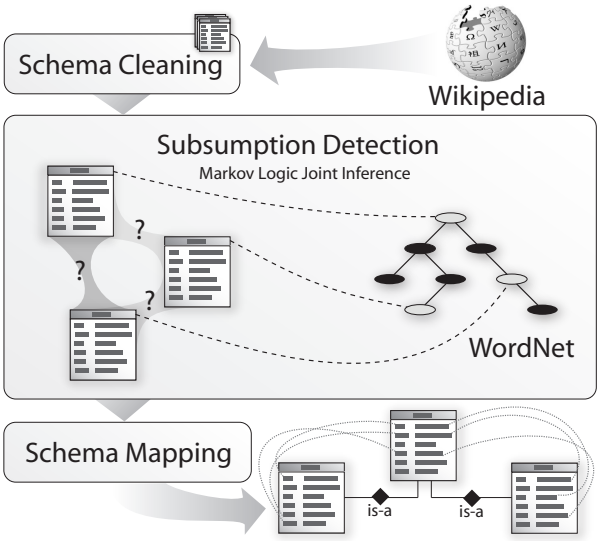


Figure 1: Architecture of Kylin Ontology Generator.

attributes enable faceted browsing, a powerful and popular way to investigate sets of articles [36].

Improving Extractors with Shrinkage: As long as an infobox class has many instances (articles), Kylin has sufficient training data to learn an accurate extractor. Unfortunately, long-tail distributions mean that most infobox classes *don't* have many instances. When learning an extractor for such a sparsely-populated class, C, one may use instances of the parent and children of C, appropriately weighted, as additional training examples [17, 34].

Semiautomatic Generation of New Templates: Today, Wikipedia infobox templates are designed manually with an error-prone "copy and edit" process. By displaying infobox classes in the context of a clean taxonomy, duplication and schema drift could be minimized. Base templates could be automatically suggested by inheriting attributes from the class' parent. Furthermore, by applying the extractors which Kylin learned for the parent class' attributes, one could automatically populate instances of the new infobox with candidate attribute values for human validation.

Infobox Migration: As Wikipedia evolves, authors are constantly reclassifying articles, which entails an error-prone conversion of articles from one infobox class to another. For example, our analysis of five Wikipedia dump "snapshots" between 9/25/06 and 7/16/07 shows an average of 3200 conversions per month; this number will only grow as Wikipedia continues to grow. An editing tool that exploited KOG's automatically-derived schema mappings might greatly speed this process, while reducing errors.

1.3 Contributions

KOG embodies several contributions:

- We address the problem of ontology refinement and identify the aspects of the Wikipedia data source which facilitate (as well as those which hinder) the refinement process. We codify a set of heuristics which allow these properties to be converted into features for input to machine learning algorithms.
- We cast the problem of subsumption detection as a machine learning problem and solve it using both support-vector machines and Markov Logic Networks (MLNs). The MLNs model is especially novel, simultaneously constructing a subsumption lattice and a mapping to WordNet using joint inference. Our experiments demonstrate the superiority of the joint inference approach and evaluate other aspects of our system.
- Using these techniques, we build a rich ontology which integrates and extends the information provided by both Wikipedia and WordNet; it incorporates both subsumption information, an integrated set of attributes, and type information for attribute values.
- We demonstrate how the resulting ontology may be used to enhance Wikipedia in many ways, such as advanced query processing for Wikipedia facts, facetted browsing, automated infobox edits and template generation. Furthermore, we believe that the ontology can benefit many other applications, such as information extraction, schema mapping, and information integration.

2. DESIDERATA & ARCHITECTURE

In order to support the applications described in the previous section, an ontology (and the process used to create it) must satisfy several criteria. First, we seek *automatic* ontology construction. While researchers have manually created ontologies, such as [12], this is laborious and requires continual maintenance. Automatic techniques, likely augmented with human review, have the potential to better scale as Wikipedia and other document stores evolve over time.

Second, the ontology should contain a well-defined ISA hierarchy, where individual classes are semantically distinct and natural classes are well represented.

Third, each class should be defined with a rich schema, listing a comprehensive list of informative attributes. Classes should be populated with numerous instances. We note that, while Wikipedia infobox classes have rich schemata, many duplicate classes and attributes exist. Furthermore, many natural classes have no corresponding Wikipdia infobox.

Fourth, classes (and attributes) should have meaningful names. Randomly-generated names, e.g. G0037, are unacceptable and overly terse names, e.g. "ABL" (the name of a Wikipedia infobox class), are less favored than alternatives such as "Australian Baseball League."

Finally, the ontology should have broad coverage — in our case, across the complete spectrum of Wikipedia articles. While these desiderata are subjective, they drove the design of KOG.

2.1 Architecture

As shown in Figure 1, KOG is comprised of three modules: the schema cleaner, subsumption detector, and schema mapper. The *schema cleaner* performs several functions. First, it merges duplicate classes and attributes. Second, it renames uncommon class and attribute names, such as "ABL," mentioned above. Third, it prunes rarely-used classes and attributes. Finally, it infers the type signature of each attribute.

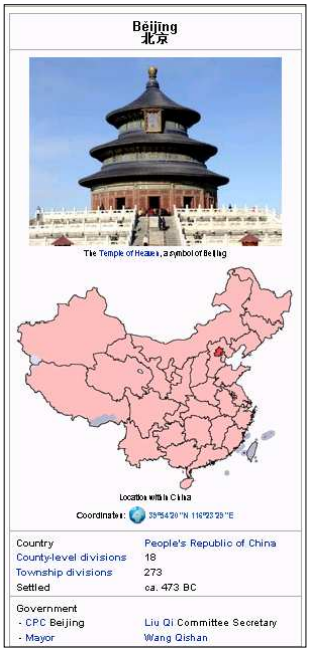
The *subsumption detector* identifies subsumption relations between infobox classes, using wide range of different features: TF/IDF-style similarity, the WordNet mapping, Wikipedia category tags, Web query statistics, and the edit history of the individual articles.

The *schema mapper* builds attribute mappings between related classes (especially between parent-child pairs in the subsumption hierarchy). Wikipedia's edit history is essential to this process.

Section 6 reports on several alternative designs for these modules. Our experiments show that a joint inference approach, which simultaneously constructs the ISA-tree, while mapping classes to WordNet, achieves the best performance. The next three sections provide details for each module.

3. SELECTING & CLEANING SCHEMATA

Schemata obtained from the Web are frequently noisy, requiring deduplication, attribute alignment and other cleaning before they




```
{{Infobox Settlement
|official_name  = Běijīng
|other_name  =
|native_name  = 北京
|settlement_type = [[Municipality of
                  China|Municipality
|image_skyline = SA Temple of Heaven.jpg
|image_caption  = The [[Temple of
                Heaven]], a symbol of Beijing
|citylogo_size  =
|image_map          = China-Beijing.png
|mapsize      = 275px
|map_caption          = Location within China
|subdivision_type     = Country
|subdivision_name     = [[People's Republic of
                  China]]
|subdivision_type1 = [[Political divisions of
                  China#County level|County-
                  level divisions]]
|subdivision_name1     = 18
|subdivision_type2  = [[Political divisions of
                  China#Township
            level|Township divisions]]
|subdivision_name2     = 273
|leader_title         =[[Communist Party of
                  China|CPC]] Beijing
|leader_name  =[[Liu Qi (Communist)|Liu Qi]]
                  Committee Secretary
|leader_title1        =  [[Mayor]]
|leader_name1         =[[Wang Qishan]]
|established_title     = Settled
|established_date      = ca. 473 BC
…
}}
```

Figure 2: Sample Wikipedia infobox and the attribute / value data used to generate it.

may be organized into an ontology. Consider the Wikipedia infobox class schemata as a case study. Widely used to display a concise, tabular summary of an article, an infobox defines subject-attribute-value triples with the intent that infoboxes of the same class will share most attributes. For example, Figure 2 shows the "Beijing" infobox for the class *settlement*; this was dynamically generated from the data shown in Figure 2. The set of attributes used in a class' infoboxes, and the types of the associated values, implicitly define the schema of individual classes. But even the explicitly defined Wikipedia schemata are noisy, with duplicate schemata, sparse instantiation, obscure class names, and untyped attributes. We now explain how to cope with these problems.

3.1 Recognizing Duplicate Schemata

One challenge stems from the need to group distinct but nearly-identical schemata. In the case of Wikipedia, users are free to modify infobox templates allowing schema evolution during the course of authoring and causing the problem of class/attribute duplication. For example, four different templates: "U.S. County" (1428), "US County" (574), "Counties" (50), and "County" (19), were used to describe the same class in the 2/06/07 snapshot of Wikipedia. Similarly, multiple tags are used to denote the same semantic attributes (e.g., "Census Yr", "Census Est." and "Census Year").

Schema matching has been extensively studied in the data-management community, and the resulting techniques apply directly to the task of duplicate detection [14, 23]. In the case of Wikipedia, however, the task is facilitated by additional features from the collaborative authoring process: redirection pages and the edit history.

Wikipedia uses redirection pages to map synonyms to a single article. For example, "Peking" is redirected to "Beijing". By checking all redirection pages, KOG notes when one infobox template redirects to another.

Next, KOG converts class names to a canonical form: parentheses are replaced with "of" (e.g., "highschool (american)" to "highschool of american"). Underscores, "_", are replaced with a space

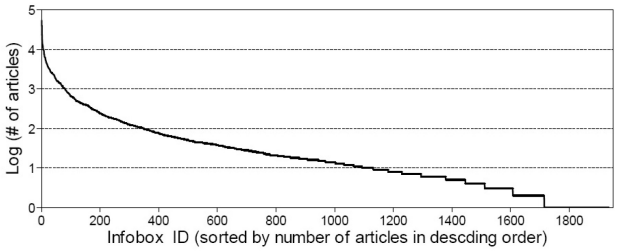


Figure 3: The number of article instances per infobox class is similar to a Zipf distribution.

and digits are discarded (e.g., "musical artist_2" to "musical artist"[3]). Finally, all tokens are converted to lowercase. Classes mapping to the same canonical form are considered to be duplicates.

Conveniently, Wikipedia records a full edit history of changes to the site; KOG exploits this information to locate duplicate attributes within each class. For example, if authors have renamed attributes from one name to another more times than a threshold (e.g., 5) then this suggests that they likely denote the same semantics and could be treated as duplicates. Similarly, if two attributes of class c never have values filled simultaneously, and they have been both transferred to the same attribute of class d, this edit pattern also indicates duplication. KOG combines these edit-history features with the evidence from canonical forms to render its final match.

3.2 Ignoring Rare Classes and Attributes

Another consequence of free authoring is schema sparseness — many classes and attributes are used rarely, leading to a long-tailed distribution. For example, Figure 3 shows the number of Wikipedia article instances (log scale) per infobox class. Of the 1935 classes, 25% have fewer than 5 instances and 11% have only one. The case is even worse for infobox-class attributes — only 46% of attributes are used by at least 15% of the instances in their class.

We observed that rare infobox classes and attributes often indicate non-representative uses (e.g., the "mayor" attribute for "U.S. County"), or result from noisy data — As a first step, KOG eliminates them from processing. Currently, KOG uses simple statistics for pruning — infobox classes with fewer than 5 articles are ignored. For each class, we keep only those attributes used by more than 15% of instance articles. In the future, we plan more sophisticated methods for dealing with rare items.

3.3 Assigning Meaningful Names

In Wikipedia a number of infobox classes have obscure or terse names, such as "ABL," mentioned earlier, and "pref gr," which denotes "prefectures of Greece." Even humans may find it difficult to discern the underlying meaning, yet without an intuitive name a class has limited value.

KOG detects which names may need renaming by identifying those which are missing (even after stemming) from a dictionary, WordNet in our case. If any token from a class name fails to match a WordNet node, KOG passes the name to the four-step procedure described below. If this procedure succeeds at any step, it terminates and returns the recovered full name.

Step 1: Split the name using case boundary information. For example, "horseRacers" would be split into "horse Racers."

Step 2: Use spell checking (i.e., the statistically-based *GoogleSpellSuggestion* function) to find transmutations, e.g. correcting

[3]Sometimes authors add digits to names to indicate a minor difference. Inspection suggests that the variants should be merged when creating a general purpose ontology.

"hungerstriker" to "hunger striker."

Step 3: Collect the category tags of all instance articles within the class, and pick the most frequent k (2 in our case) tags. If the abbreviated form of one tag matches the class name, the tag is treated as the recovered name. Otherwise, the most frequent tag is returned. With the "ABL" class, for example, "Australian Baseball Team" and "Australian Baseball League" are the two most frequent tags, and "Australian Baseball League" would be returned.

Step 4: Query Wikipedia to see if there is an article corresponding to the given class. If it is a redirected page and the title has a good form (as measured heuristically), such as "Video Game" redirected from "cvg", KOG uses the title for the new class name. Otherwise, it uses the definition phrase in the first sentence of the article as the final result. For example, for the "amphoe" class, there is an "Amphoe" article whose first sentence reads "An amphoe is the second level administrative subdivision of Thailand," and so KOG uses *"second level administrative subdivision of Thailand"* as the class name.

These four heuristics may also be used to rename obscurely named *attributes*, such as "yrcom," (year of commission). In addition, KOG uses Wikipedia's edit history to see if people have manually renamed the attribute in some instances of the class. For example, "stat_pop" can be renamed "population estimation," because users have made some transfers between these two attributes.

3.4 Inferring Attribute Types

Even though infobox classes have associated schemata, there is no type system for attribute values. Indeed, since infoboxes are intended solely to provide convenient visual summaries for human readers, there is no *guarantee* that users are consistent with datatypes. Yet inspection shows that most attributes *do* have an implicit type (e.g. "spouse" has type "person"), and if types could be inferred, they would greatly facilitate extraction, fact checking, integration, etc.

KOG infers attribute types from the corresponding set of values, using the following five-step procedure:

Step 1: Let c be an infobox class with attribute a. For example, a might be the "spouse" attribute of the "person" class. KOG generates the set of possible values of a, and finds the corresponding Wikipedia objects, $V_{c,a}$; note not every value will have corresponding Wikipedia article.

Step 2: Create a partial function $\omega : V_{c,a} \rightarrow N_w$ from value objects to WordNet nodes by combining two preexisting partial mappings. The first source, "DBpediaMap," is DBpedia's [5] manually created mapping from 287,676 articles to corresponding WordNet nodes. If DBpediaMap does not have an image for $v \in V_{c,a}$, then KOG uses "YagoMap," an automatically-created mapping, which links a greater number, 1,227,023, of Wikipedia articles to WordNet nodes [32].

Step 3: Consider the set of WordNet nodes $\{n$: there exist at least t distinct $v \in V_{c,a}$ such that $\omega(v) = n\}$ for some threshold, t (we use $t = 10$). If there are at least 2 nodes in this set, KOG considers the two which are mapped by the most values in $V_{c,a}$ and finds their relationship in WordNet. If the relationship is *alternative, sibling, or parent/child*, KOG returns their least common parent synset as the final type for the given attribute. For example, if the two most frequent nodes are "physicist" and "mathematician", then KOG would choose type "scientist," because it is the direct parent of those two siblings. If no relationship is found, KOG sets the type equal to the synset of the most frequent node.

Step 4: If no WordNet node is mapped by at least t values in $V_{c,a}$, KOG creates a larger set of values, V, by adding the values of a similar class, c' which also has attribute a. For example, Wikipedia entities from "Person.Spouse" and "Actor.Spouse" would be put together to compute the accumulated frequency. The most frequent WordNet node would be returned as the type of the target attribute.

Step 5: If Step 4 also fails, KOG analyzes the edit history to find the most related attribute, which has the highest number of transfers with the target attribute. The type of this most-related attribute is then returned as the type of a.

KOG can also generate a type signature for a complete infobox class. Indeed, this is easy after the class has been mapped to a WordNet node, which is described in the next section.

4. DETECTING SUBSUMPTION

Detecting subsumption relations, i.e. that one class *ISA* subset of another, is the most important challenge for KOG. We model this task as a binary classification problem and use machine learning to solve it. Thus, the two key questions are: 1) which features to use, and 2) which machine learning algorithm to apply. In fact, we implemented two very different learning frameworks: SVMs and a joint inference approach based on Markov logic. The next subsection defines the features: a mixture of similarity metrics and Boolean functions. Section 6 shows that our joint inference approach performs substantially better.

4.1 Features for Classification

KOG uses six kinds of features, some metric and some Boolean.

Similarity Measures: Class similarity is an indication of subsumption, though not a sufficient condition. KOG uses four different similarity metrics.

Attribute Set Similarity: KOG models a class as the set of its attributes, compresses each set into a bag of words, and computes the TF/IDF similarity score between the bags.

First Sentence Set Similarity: For each class, KOG creates a bag of words by taking the first sentence of each instance of the class. Again the TF/IDF score between the bags defines the similarity.

Category Set Similarity: The bags are created from the category tags attached to the class instances.

Transfer Frequency: This similarity score is computed from Wikipedia's edit history. If c and c' denote two classes, define their *transfer-frequency* similarity as the number of articles whose class membership switched from c to c' or vice versa. We normalize this frequency to $[0, 1.0]$.

Class-Name String Inclusion: Inspired by [33], we say that the feature *isaFT(c,d,Contain)* holds iff: 1) the name of d is a substring of c's name, and 2) the two names have the same head (as determined by the Stanford parser [2]). For example, the feature holds for c = "English public school" and d = "public school," since both have "school" as head.

Category Tags: Many infobox classes have their own Wikipedia pages, and sometimes a special type of category, "XXX infobox templates," is used to tag those pages. We say that the feature *isaFT(c,d,HasCategory)* holds if class c has a special category tag called "⟨name of d⟩ infobox templates."

For example, the page for the "volleyball player" class has a category tag called "athlete infobox templates," and there exists another class named "athlete," so *isaFT("volleyball player", "athletes", HasCategory)*. This feature is strongly linked to subsumption (e.g., "volleyball player" *ISA* "athlete," but nothing is guaranteed. For example, both "athlete" and "Olympic" classes have the category tag "Sports infobox templates", but neither of them *ISA* sports.

Edit History: The edit patterns from Wikipedia's evolution contain useful information — intuitively, when changing the type of an instance, an author is more likely to specialize than generalize. We

define the *degree* of class c as the number of classes transferring with c. Given a pair of classes c and d, KOG checks: 1) whether the *transfer-frequency* between c and d is high enough (e.g, bigger than 5 in our case); 2) Whether the degree of d is much bigger than that of c (e.g. more than twice as big). If both conditions are true, we say the feature *isaFT(c,d,EditH)* holds — weak evidence for *"A ISA B"*.

Hearst Patterns: Following [19, 16], KOG queries Google to collect type information about class names using patterns such as *"NP0, like NP1"* and *"NP0 such as NP1"*, which often match phrases such as "...scientists such as phsyicists, chemists, and geologists." We say *isaFT(c,d,HPattern)* holds if the Google hit number for HPattern(c,d) is big enough(e.g. 200 in our case) while very small for HPattern(d,c)(e.g. less than 10 in our case).

WordNet Mapping: By computing a mapping from infobox classes to WordNet concept nodes, KOG gains useful features for predicting subsumption. For example, if both c and d have perfectly corresponding nodes in WordNet and one WordNet node subsumes the other (say *isaFT(c,d,isaWN)*), then this is likely to be highly predictive for a learner. Since computing the mapping to WordNet is involved, we describe it in the next subsection.

4.2 Computing the WordNet Mapping

In this section we explain how KOG generates two mappings between infobox classes and WordNet nodes: $\omega(c)$ returns a node whose name closely matches the name of c, while $\varphi(c)$ denotes the node which most frequently characterizes the *instances* of c according to Yago [32]. Based on these two mappings, KOG seeks the closest semantic match $\varpi(c)$ for each class c in WordNet (e.g., "scientist" class should map to the "scientist" node instead of the "person" node), and outputs one of the seven mapping types characterizing different degrees of match; in descending order of strength we have: *LongName*, *LongHead*, *ShortNameAfterYago*, *ShortHeadAfterYago*, *HeadYago*, *ShortName*, and *ShortHead*. The following steps are tried in order until one succeeds.

Step 1: If class c's name (after cleaning) has more than one token, and has an exact match in WordNet, $\omega(c)$, then $\omega(c)$ is output as the closest semantic match $\varpi(c)$ with mapping type *LongName*. This kind of mapping is very reliable — a random sample of 50 cases showed perfect precision.

Step 2: KOG uses the Stanford parser to locate the head of c's name and returns the WordNet node which matches the longest substring of that head, $\omega(c)$. For example, "beach volleyball player," is matched to "volleyball player" in WordNet, instead of "player." If the matched head has more than one token, then $\omega(c)$ is returned with type *LongHead*; a sample shows that it is also very reliable.

Step 3: If neither of the previous techniques work, KOG looks for a consensus mapping amongst articles which instantiate the class, much as it did when determining an attribute's type in Section 3.4. However, instead of using both the DBpediaMap and YagoMap to define the mapping, as done previously, KOG just uses YagoMap, saving the higher-quality, manually-generated DBpediaMap to use as training data for the learner. Let I_c denote the instances of infobox class c; for all $o \in I_c$ let $\varphi(o)$ be the WordNet node defined by Yago. Let $\varphi(c)$ be the most common node in $\varphi(I_c)$. If c's head is a single token, and has a matched node $\omega(c)$ in WordNet, KOG checks the relationship between $\omega(c)$ and $\varphi(c)$ in WordNet. If $\omega(c)$ is a child or alternative of $\varphi(c)$ and the head is the class name itself (i.e., c's name is a single token), KOG returns $\omega(c)$ with type *ShortNameAfterYago*; otherwise, KOG returns $\omega(c)$ with type *ShortHeadAfterYago*. If no relationship is found between $\omega(c)$ and $\varphi(c)$, KOG returns $\varphi(c)$ with type *HeadYago*. If no $\varphi(c)$ is found, KOG returns $\omega(c)$ with type of either *ShortName* or *ShortHead*, depending on whether c is single token. As in Yago [32], we select the most frequent sense of the mapped node in WordNet, which turns out to work well in most cases.

To finally determine whether $\varpi(c)$ is returning a good mapping, KOG encodes its mapping type as a Boolean feature: *mapType(c,t)*, where t denotes one of the seven types (e.g., *LongName*). A support vector machine (SVM) is learned using DBpediaMap as a training set (We used the LIBSVM implementation [3]). In this way, the SVM learns relative confidences for each mapping type and outputs a score for the WordNet mappings. This score can be used to easily control the precision / recall tradeoff. Furthermore, the score could also identify potentially incorrect mappings for further verification (whether manual or automatic).

Now, when given two classes c and d, KOG can check whether $\varpi(c)$ subsumes $\varpi(d)$ in WordNet. If so, KOG constructs the feature, *isaFT(c,d,isaWN)*, which is likely to be highly predictive for the subsumption classifier described next.

We close by noting that, in addition to being a useful feature for the subsumption classifier, the WordNet mapping has other important benefits. For example, each node in WordNet has an associated set of synonyms which can be used for query expansion during query processing over the infobox knowledge base. For example, consider a query about ballplayers born in a given year. Even though there is no "ballplayer" class in Wikipedia, WordNet knows that "ballplayer" and "baseball player" are synonyms and so a query processing system can operate on records of the "baseball player" class. Additionally, associating the attributes from a Wikipedia schema (as well as a long list of class instances) with a WordNet node may also provides substantial benefit to WordNet users as well.

4.3 Max-Margin Classification

One might think that there would be no need to learn a second classifier for subsumption, once KOG has learned the mapping from infobox classes to WordNet, but in practice the WordNet mapping is not 100% correct, so the other features improve both precision and recall. But even if the first SVM classifier could learn a correct mapping to WordNet, it would be insufficient. For example, "television actor" and "actor" are both correctly mapped to the WordNet node "person," but this mapping is incapable of predicting that "actor" subsumes "television actor." Instead, KOG treats the mapping as just another feature and learns the subsumption relation using all available information.

KOG uses the "DBpediaMap" to constructs the training dataset (details in section 6.2) to train an SVM classifier for subsumption. By automatically weighing the relative importance of all features, KOG finds an optimal hyperplane for classifying subsumption. With a confidence threshold of 0.5, the SVM achieves an average precision of 97.2% at a recall of 88.6%, which is quite good.

4.4 Classification via Joint Inference

While the SVM's performance is quite good, there is still space to improve. First, the SVM classfier predicts *ISA* between each pair of classes sequentially and separately. This local search ignores evidence which is potentially crucial. For example, if *"c ISA d"* and *"d ISA e"*, then it is likely that *"c ISA e"*, even if no strong features observed for the pair of c and e.

Secondly, the SVM classifiers separate the WordNet mapping and *ISA* classification as input and output, so that the crosstalk between these two parts is blocked. In reality, however, these two problems are strongly intermixed and evidence from either side can help to resolve the uncertainty of the other side. For example, given that class c and d have correct mappings to WordNet and *"$\varpi(c)$ ISA*

$\varpi(d)$", it is likely that *"c ISA d"*; on the other hand, if *"c ISA d"* but the retrieved mappings $\varpi(c)$ and $\varpi(d)$ have no *ISA* relationship in WordNet, then it is clear that something is wrong — but the SVM won't recognize the problem.

In contrast, a relational-learning model capable of joint inference *can* exploit this global information. To see if this would lead to greater performance, we applied the Markov Logic Networks(MLNs) model. By addressing *ISA* classification and WordNet mapping jointly, our MLNs model achieves substantially better performance for both tasks. Section 6 provides detailed experiments, but we note that with a confidence threshold of 0.5, our MLNs model eliminated 43% of the residual error while simultaneously increasing recall from 88.6% to 92.5%.

Before describing our MLNs classifier in more detail, we provide background on Markov Logic Networks.

Markov Logic Networks

A first-order knowledge base can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea of MLNs is to soften these constraints: when a world violates one formula in the KB it is deemed less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

(From Richardson & Domingos [28]) A *Markov Logic Network* L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.

2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.

Thus, there is an edge between two nodes of $M_{L,C}$ if and only if the corresponding ground predicates appear together in at least one grounding of one formula in L. An MLNs can be viewed as a *template* for constructing Markov networks. The probability distribution over possible worlds x specified by the ground Markov network $M_{L,C}$ is given by

$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}}) \tag{1}$$

where Z is the normalization factor, $\phi_i(x_{\{i\}})$ is the potential function defined on the ith clique which is related to a grounding of formula F_i, and $x_{\{i\}}$ is the discrete variable vector in the clique. Usually, it is represented as follows,

$$\phi_i(x_{\{i\}}) = \begin{cases} e^{w_i} & F_i(x_{\{i\}}) = True \\ 1 & F_i(x_{\{i\}}) = False \end{cases} \tag{2}$$

In this way, we can represent the probability as follows,

$$P(X = x) = \frac{1}{Z} \exp\left\{ \sum_i w_i n_i(x) \right\} \tag{3}$$

where $n_i(x)$ is the number of true groundings of F_i in x.

Using MLNs to Classify Subsumption

KOG uses the open source Alchemy system [22] to implement its MLNs classifier. As described in Section 4.1, two predicates are used to represent features: *mapType(c,t)* and *isaFT(c_1,c_2,f)*. Two query predicates *isa(c_1, c_2)* and *map(c)* are used to express the uncertainty of *ISA* classification and WordNet mapping, respectively. After learning, Alchemy computes the probabilities of these predicates.

We use three kinds of logical formulas to guide KOG's learning. The first represents the loose connection between WordNet mappings and the corresponding types. For example,

$$mapType(c, LongName) \Leftrightarrow map(c)$$

which means "Class c has a long class name and exactly matches a node in WordNet if and only if this mapping is correct."[4] Remember that Alchemy will learn the best probabilistic weight for this and the other rules. By using a metavariable, "$+t$," instead of the constant *LongName*, we direct Alchemy to learn weights for all possible indications: $mapType(c, +t) \Leftrightarrow map(c)$.

The second class of formulas encode the intuition that 1) *ISA* is transitive, and 2) features such as *isaFT(c_1,c_2,Contain)* are likely correlated with subsumption:

$$isa(c_1, c_2) \wedge isa(c_2, c_3) \Rightarrow isa(c_1, c_3)$$
$$isaFT(c_1, c_2, +f) \Leftrightarrow isa(c_1, c_2)$$

The final formulas encode the connection between the WordNet mapping and *ISA* classification:

$$isaFT(c_1, c_2, isaWN) \wedge map(c_1) \wedge map(c_2) \Rightarrow isa(c_1, c_2)$$

which means "if c_1 and c_2 both have correct WordNet mappings and the mapped nodes are *ISA* in WordNet, then c_1 *ISA* c_2."

Two other formulas complete the intuition:

$$isaFT(c_1, c_2, isaWN) \wedge isa(c_1, c_2) \Rightarrow map(c_1) \wedge map(c_2)$$
$$map(c_1) \wedge map(c_2) \wedge isa(c_1, c_2) \Rightarrow isaFT(c_1, c_2, isaWN)$$

Discriminative learning is used to determine the weights of formulas [30], and MC-SAT is used for inference [27]. Experimental results show that this joint inference approach improves the precision of both *ISA* classification and WordNet mapping.

5. MAPPING ATTRIBUTES ACROSS SCHEMATA

Schema mapping is a well-studied database problem, which seeks to identify corresponding attributes among different relational schemata [14, 23]. With KOG, we take a simple approach which exploits the structure of Wikipedia, relying on the edit history (e.g., Wikipedia's sequential record of every edit to every page) and string similarity comparison to find attribute mappings.

Let c and d denote two classes, typically a parent/child pair from the subsumption lattice constructed in the previous section. KOG considers different pairs of attributes, looking for a match by checking the following conditions in turn:

Step 1: If the *transfer frequency* between two attributes $c.a$ and $d.b$ is high enough (≥ 5 in our case), KOG matches them.

Step 2: If data is sparse, KOG considers attribute names independent of class, looking at the edit history of all classes with attributes named a and b. For example, it treats "actor.spouse" and "person.spouse" both as "spouse," and "person.wife" and "musician artist.wife" both as "wife," and computes the sum of the transfer frequencies between all possible pairs of attributes (a, b). If an

[4] In fact, Alchemy converts the bidirectional implication into two separate clauses, one for each direction; this allows it to learn different weights for each direction.

attribute $c.a$ wasn't mapped in Step 1 and the *transfer frequency* between attribute a and b is over threshold in this aggregate fashion, then KOG maps $c.a$ to $d.b$.

Step 3: If the previous steps fail, KOG uses the lexical, string comparison method, like that of Section 3.1.

Once mapping is complete, KOG iterates over all attributes, collecting every corresponding attribute into a bag of alternative names. For example, the "birth place" attribute of "person" is given the following alternative names: birthplace, place of birth, place birth, location, origin, cityofbirth, born." This naming information is helpful for query expansion and for other tasks (e.g., query suggestion, information integration, etc.)

Since KOG has already estimated a type signature for each attribute, it uses this to double-check whether the attribute mapping is consistent. Those which fail to match are tagged for subsequent verification and correction, which could be manual or automatic. In the future, we intend to add the attribute mapping phase, with type consistency, into our joint inference approach.

6. EXPERIMENTS

To investigate KOG's performance, we downloaded the English version of Wikipedia for five dates between 09/25/2006 and 07/16/2007. We evaluated ontology refinement on the 07/16/2007 snapshot; previous versions were used to compute edit-history information. There are many measurements for taxonomy creation. We chose the most general criteria of precision and recall.

6.1 Selecting & Cleaning Schemata

This section addresses three questions: How does KOG recognize duplicate schemata? How does it assign meaningful names? And what is the precision for attribute type inference?

Recognizing Duplicate Schemata

Our data set contained 1934 infobox templates. By following redirected pages, KOG found 205 duplicates; checking canonical forms identified another 57. A manual check yielded an estimate of 100% precision. To estimate recall, we randomly selected 50 classes and found 9 duplicates by manual checking. KOG successfully identified 7 of them, which leads to an estimated recall of 78%. Since KOG also prunes classes containing less than 5 instance articles, 1269 infobox classes are selected.

For attributes, KOG found 5365 duplicates — about 4 per class. We randomly selected 10 classes and manually identified 25 true duplications. On this set KOG predicted 23 duplicates of which 20 of them were correct. This leads to an estimated precision of 87%, and estimated recall of 79%. Since KOG ignores attributes which are used by less than 15% instance articles, 18406 attributes (out of 40161) survived cleaning.

Assigning Meaningful Names

By referring to WordNet, KOG selected 318 out of 1269 infoboxes for name recovery. KOG found names for 302 of these candidates and manual checking rated 267 of them to be correct. This is quite encouraging, because many class names are extremely hard to interpret — even for human beings. For example, KOG correctly renamed "wfys" to be "youth festivals" and renamed "nycs" to "New York City Subway." Table 1 shows the detailed contribution of each heuristic, where "All" means the combination of every heuristic, as described in section 3.1.

For attributes, we randomly selected 50 classes which contain a total of 654 attributes. By referring to WordNet, KOG identified 153 candidates, and it reassigned names to 122 of them; 102 of the new names were correct. This leads to an estimated precision of 84% and recall of 67%. We note that KOG didn't perform as well here as it did when renaming class names. One explanation may be that less attention is paid by humans to attribute names, and this provides a weaker signal for KOG to exploit.

Heuristic	Precision(%)	Recall(%)	F-Measure(%)
CaseCheck	97.0	10.1	18.2
GoogleSpell	91.7	6.9	12.9
Category	86.7	61.3	71.8
WikiQuery	90.0	5.7	10.7
All	88.4	84.0	86.1

Table 1: Performance of assigning meaning full class names.

Inferring Attribute Types

In order to check the performance of type inference, we randomly picked 20 infobox classes, which had a total of 329 attributes. KOG predicted a type for 282 of these and 186 predictions were correct. This leads to an estimated precision of 66% with a recall of 57%. These results are acceptable given the problem difficulty and lack of labeled training data, but we anticipate that by incorporating the techniques introduced by the REALM model [15], KOG could do substantially better.

6.2 Subsumption Detection

We now focus on three additional questions: What are the precision and recall of subsumption detection? How does KOG identify incorrect WordNet mappings? And what is the benefit (if any) of joint inference? First, however, we describe how KOG automatically derive a training dataset based on open Web resources.

Training dataset construction

Recall that "DBpediaMap" contains manually-created mappings from 287,676 articles to their corresponding WordNet nodes; the articles come from 266 of the infobox classes. KOG uses this data to construct the pseudo-training dataset for subsumption detection[5]. We call it "pseudo" because it is constructed indirectly, as follows. For each class covered by "DBpediaMap", we first select the most frequent aggregated label over its instance articles. Then we decide whether this is a *NameMap* or *HeadMap*: if the label exactly matches the class name or one of its alternative terms in WordNet, we call it a *NameMap*; otherwise call it a *HeadMap*. Besides "DBpediaMap", two other mapping types are also added to the pseudo dataset due to their high precision: one is *LongName* and another *LongHead*. In this way, we get a dataset of 401 classes with pseudo-labeled WordNet mappings. Then KOG produces positive and negative *ISA* pairs by following the hyponym tree in WordNet:

- Suppose both class c and d have *NameMap* $\varpi(c)$ and $\varpi(d)$. If $\varpi(c)$ *ISA* $\varpi(d)$, KOG labels *"c ISA d"*. Otherwise, *"c NOT ISA d"*;
- Suppose class c has *HeadMap* $\varpi(c)$ and class d has *NameMap* $\varpi(d)$. If $\varpi(c)$ *ISA* $\varpi(d)$*"*, or $\varpi(c)$ is an alternative term of $\varpi(d)$, we label *"c ISA d"*.

In this way, KOG gets 205 positive and 358 negative ISA pairs for training.

Results

To better understand the source of performance at subsumption classification, we also implemented a simplified MLNs classifier; it uses exactly the same features as the SVM classifier (without the formulas for crosstalk between WordNet mapping and *ISA* classification). For clarity, we call the simplified model "MLN," and the fully-functional one "MLN+." We test each model's performance with 5-fold cross validation on the pseudo-labeled dataset.

[5]Other datasets, like the automatically compiled "YagoMap" or the "Category Ontology" from [26] can also serve for this purpose

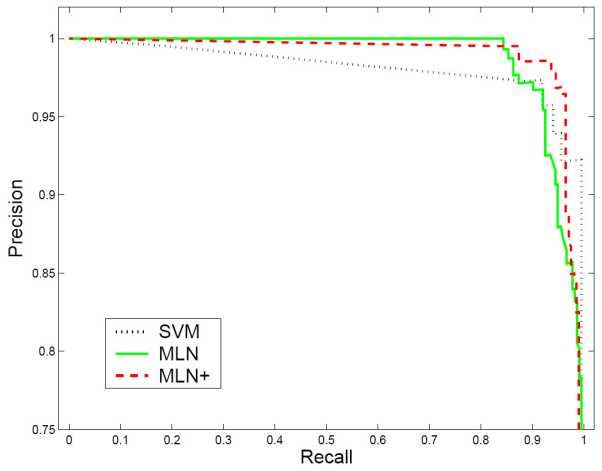


Figure 4: Precision vs. recall curve of ISA-tree construction.

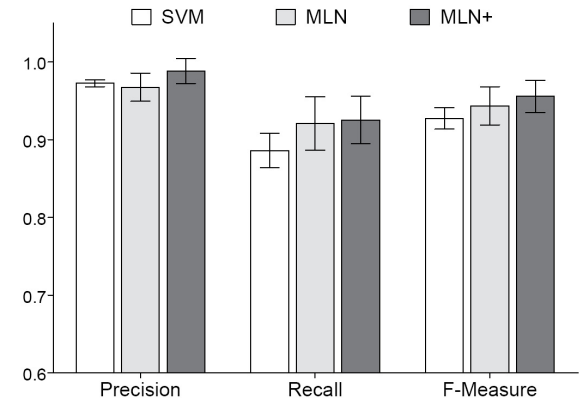


Figure 5: ISA classification with confidence threshold set as 0.5.

Figure 4 shows the precision/recall curves for subsumption classification. All three models perform well. We suspect most people are willing to trade away recall for higher precision. In this sense, both MLNs models perform better than the SVM classifier, and MLN+ is the best by further extending the recall. To have a close look, we set the confidence threshold at 0.5, and compared three models' performance in Figure 5. The SVM classifier achieves an excellent precision of 97.2% and recall of 88.6%. The MLN model drops precision to 96.8% but has better recall at 92.1%. And MLN+ wins on both counts, extending precision to 98.8% (eliminating residual error by 43%) and recall to 92.5%. Since the only difference between the two MLNs are the formulas inducing joint inference, it is clear that this is responsible.

As we mentioned before, the WordNet mapping is useful in its own right. To check how joint inference affects this task, we again implemented a simplified MLNs and compared the performance of three models. Figure 6 shows that both MLNs models achieve a big improvement over the SVM classifier.The MLN+ model has overall better performance than MLN, especially at high recall. This improvement stems from MLN+'s ability to identifying incorrect WordNet mappings, as shown in Figure 7. This ability may translate into an effective *mixed-initiative interface*, since the MLN+ model will be able to drive active learning, asking humans to correct examples which it knows are incorrect.

6.3 Mapping Attributes Across Schemata

This section evaluates the precision and recall of KOG's schema mapping capability. In particular, we are interested in the ability to

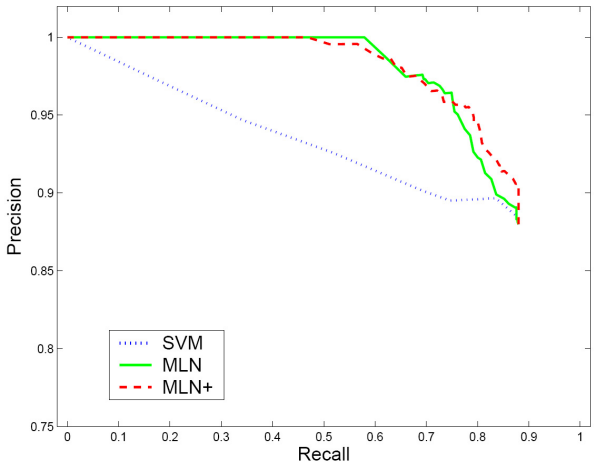


Figure 6: Precision vs. recall curve of WordNet mapping.

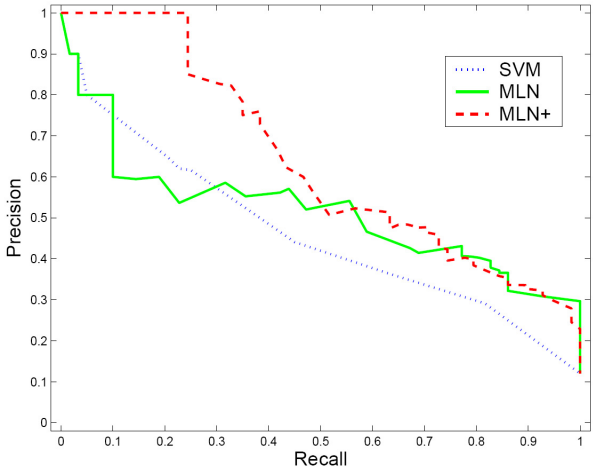


Figure 7: Relative performance detecting incorrect WordNet mappings.

accurately map corresponding attributes between parent and child schemata. To perform the evaluation, we took a random sample of 10 *ISA* class pairs (comprising 20 classes) from the constructed subsumption lattice. Manual checking revealed a total of 91 true mappings. KOG made 84 predictions and 79 of them were correct. This leads to an estimated precision of 94% and recall of 87%. There are two main causes for incorrect mappings: first, some ambiguous attributes are invented for flexible visualization purpose. For example, "free" and "free_label" are widely used by users to define attributes. This ambiguity misleads KOG to link several attributes to these free attributes. Secondly, the string-similarity heuristic also produces errors occasionally. For example, "road.direction_a" is mapped to "route.direction_b" since only one character is different. In the future we hope to incorporate recently-developed schema-mapping techniques from the database community in order to boost precision and recall.

6.4 Enabling Advanced Queries

The main motivation for KOG was the hope that the resulting ontology would support advanced queries over data extracted from Wikipedia and the Web. As preliminary confirmation of this, we did a case study to check its support for query over Wikipedia infoboxes. It turns out KOG helps to extend the recall in many cases. For example, given a query like:

- *"Which performing artists were born in Chicago?"*

Without the refined ontology, one would likely return zero results, because there is no "performing artist" infobox in the current Wikipedia. However, with KOG we know "performer" is an alternative of "performing artist" and its "location" attribute has "born" as an alias. As a result, the answer "Michael Ian Black" would be successfully retrieved from an infobox. Furthermore, by following the ISA tree, we know "actor" and "comedian" are children of "performer", and their attributes "birthplace", "birth place", "cityofbirth", "place of birth", "origin" are duplicates, all mapping to the "location" attribute of "performer." This expansion allows the return of 162 additional answers from "actor" and one additional answer from the "comedian" class.

7. RELATED WORK

Ontology Construction Based on Wikipedia Suchanek et al. built the Yago system by unifying WordNet and Wikipedia, where leaf category tags are mapped to WordNet nodes with rule-based and heuristic methods [32]. Strube et al. derived a large scale taxonomy based on the Wikipedia category system by applying several heuristics to identify the ISA relationships among category tags [26]. In contrast with this work, we focus on combining Wikipedia infoboxes with WordNet, and trained a sophisticated MLNs model to jointly infer the WordNet mapping and ISA classification. In some sense, their work and ours are complementary to each other: they achieve greater coverage with category tags, and we provide detailed schemata together with attribute mappings.

Herbelot et al. extracted ontological relationships from Wikipedia's biological texts, based on a semantic representation derived from the RMRS parser [21]. In contrast, KOG constructs a broad, general-purpose ontology. Hepp et al. propose to use standard Wiki technology as an ontology-engineering workbench and show an application of treating Wikipedia entries as ontology elements [20]. We are also motivated by the special structures (e.g., infoboxes) in Wikipedia, and try to address more advanced problems, such as subsumption extraction and schema mapping.

Learning Relations from Heterogenous Evidence Cimiano et al. trained an SVM classifier to predict taxonomic relations between terms by considering features from multiple and heterogeneous sources of evidence [10]. For KOG, we also used SVM classifiers to handle diverse features from heterogenous evidences. Furthermore, we also applied an MLNs model, showing the benefit of joint inference: by using a single MLNs classifier, KOG creates the WordNet mapping and ISA classification simultaneously — getting better performance on both tasks.

Snow et al.'s work [31] is closer to ours in the sense of handling uncertainty from semantic relationships and WordNet mappings all together over heterogenous evidence. However there are several important differences. First, they use local search to incrementally add one new relation in each step, greedily maximizing the one-step increase in likelihood. This hill-climbing model risks slipping into a local maximum, with no ability to jump to a globally better solution. In contrast, we use a MLNs model to jointly infer the value of all relations, more likely finding the optimal solution. Second, Snow et al. assume that each item of evidence is independent of all others given the taxonomy, and depends on the taxonomy only by way of the corresponding relation. In contrast, our MLNs model doesn't make any independence assumption during inference.

Schema Matching Several of the problems addressed by KOG may be seen as instances of the schema-matching problem: recognizing duplicate schemata, finding duplicate attributes, and matching the attributes of one schema with those of a subsuming class. Many researchers have investigated this general problem, especially those in the database and IR communities. For example, Doan et al. developed a solution combining several types of machine learning [14]. Madhavan et al. proposed a corpus-based matching approach which leverages a large set of schemata and mappings in a particular domain to improve robustness of matching algorithms [23]. He and Chang proposed a statistical schema mapping framework across Web query interfaces by integrating large numbers of data sources on the Internet [18]. Bilke and Naumann exploit the existence of duplicates within data sets to perform automatic attribute mappings [6]. We would like to incorporate these approaches into KOG, but to date have implemented a simpler, heuristic approach which exploits Wikipedia-specific structure to yield acceptable performance.

General Ontology Construction The most widely used method for automatic ontology extraction is by lexico-syntactic pattern analysis. This is first proposed by Marti Hearst to acquire hyponyms from large text corpora [19], and later followed by many successful systems, such as KnowItAll [16] and PANKOW [8, 9]. Cafarella et al. proposed the TGen system to discover schemas from the unstructured assertions harvested from the Web [7]. Another general way to learn ontology is clustering concept hierarchies as in [11]. Linguistic approaches are also applied, such as OntoLearn [33] and TextToOnto [29]. All these methods mainly focus on unstructured texts, while we fully exploited the rich (semi)structured information available on the Web, such as infoboxes in Wikipedia, to help ontology construction. These two methods can benefit each other by either improving precision or extending coverage.

Other Wikipedia-Related Systems Milne et al. implemented a new search engine interface called Koru, which harnesses Wikipedia to provide domain-independent, knowledge-based information retrieval [24]. Adler and Alfaro proposed a reputation system for Wikipedia which checks whether users' edits are preserved by subsequent authors [4]. Nguyan et al. try to gather assertions from Wikipedia articles by locating entity pairs in sentences and using an SVM to classify them into 13 predefined relationships [25]. DeRose et al. proposed a Cwiki approach to combine both machine and human's contributions to build community portals such as Wikipedia [13]. One of their core tasks is to address the inconsistency between machine and human contributors; here, our automatically-refined Wikipedia ontology could be helpful. On the flip side, Cwiki provides a good platform to implement applications, such as faceted browsing, or structured querying, which were based on our refined ontology.

8. CONCLUSION

Wikipedia is developing as the authoritative store of human knowledge. Recently, the combined efforts of human volunteers have extracted numerous facts from Wikipedia, storing them as machine-readable object-attribute-value triples in Wikipedia infoboxes. Furthermore, machine-learning systems, such as Kylin [35], can use these infoboxes as training data, and then accurately extract even more triples from Wikipedia's natural-language text. This huge repository of structured data could enable next-generation question answering systems which allow SQL-like queries over Wikipedia data, faceted browsing, and other capabilities. However, in order to realize the full power of this information, it must be situated in a cleanly-structured ontology.

This paper makes a step in this direction, presenting KOG, an autonomous system for refining Wikipedia's ontology. We cast the problem of ontology refinement as a machine learning problem and present a novel solution based on joint inference implemented using Markov Logic Networks. Our experiments show that

joint-inference dominates other methods, achieving an impressive 96.8% precision at 92.1% recall. The resulting ontology contains subsumption relations and schema mappings between Wikipedia's infobox classes; additionally, it maps these classes to WordNet.

In the future we intend to use the ontology to develop an improved query interface for Wikipedia and the Web. Combining Kylin with KOG is an obvious first step. We also anticipate an inference scheme which combines multiple facts to answer a broader range of questions. There are also several ways to improve KOG itself, including improved word sense disambiguation and extending our joint-inference approach to include schema mapping.

Acknowledgments We thank Eytan Adar, AnHai Doan, Oren Etzioni, Raphael Hoffmann, Stephen Soderland and the anonymous reviewers for valuable suggestions. This work was supported by NSF grant IIS-0307906, ONR grant N00014-06-1-0147, SRI CALO grant 03-000225 and the WRF / TJ Cable Professorship.

REFERENCES

[1] http://wordnet.princeton.edu/.

[2] http://nlp.stanford.edu/downloads/lex-parser.shtml.

[3] http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[4] B. T. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 261–270, New York, NY, USA, 2007. ACM.

[5] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *Proceedings of ISWC07*, 2007.

[6] A. Bilke and F. Naumann. Schema matching using duplicates. *Proceedings of ICDE05*, 2005.

[7] M. J. Cafarella, D. Suciu, and O. Etzioni. Navigating extracted data with schema discovery. *Proceedings of WebDB07*, 2007.

[8] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. *Proceedings of WWW04*, 2004.

[9] P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: Context-driven automatic semantic annotation with c-pankow. *Proceedings of WWW05*, 2005.

[10] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. *Ontology Learning from Text: Methods, Evaluation and Applications*, 2005.

[11] P. Cimiano and S. Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. *Proceedings of Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods, ICML05*, 2005.

[12] P. Clark and B. Porter. Building concept representations from reusable components. *Proceedings of AAAI97*, 1997.

[13] P. DeRose, X. Chai, B. Gao, W. Shen, A. Doan, P. Bohannon, and J. Zhu. Building community wikipedias: A human-machine approach. *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE-08)*, Cancun, Mexico, 2008.

[14] A. Doan, P. Domingos, and A. Y. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301, 2003.

[15] D. Downey, S. Schoenmackers, and O. Etzioni. Sparse information extraction: Unsupervised language models to the rescue. *Proceedings of ACL07*, 2007.

[16] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[17] D. Freitag and A. McCallum. Information Extraction with HMMs and Shrinkage. *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, Florida, 1999.

[18] B. He and K. Chang. Statistical schema matching across web query interfaces. *Proceedings of SIGMOD03*, 2003.

[19] M. Hearst. Automatic acquisition of hyponyms from large text corpora. *Proceedings of COLING92*, 1992.

[20] M. Hepp, K. Siorpaes, and D. Bachlechner. Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. *IEEE Internet Computing*, 11(5):54–65, 2007.

[21] A. Herbelot and A. Copestake. Acquiring ontological relationships from wikipedia using rmrs. *Proceedings of Workshop on Web content Mining with Human Language Technologies, ISWC06*, 2006.

[22] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, and P. Domingos. The alchemy system for statistical relational AI, technical report, university of washington, 2006.

[23] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. *Proceedings of ICDE05*, 2005.

[24] D. Milne, I. H. Witten, and D. Nichols. A knowledge-based search engine powered by wikipedia. *Proceedings of CIKM07*, 2007.

[25] D. P. T. Nguyen, Y. Matsuo, and M. Ishizuka. Relation extraction from wikipedia using subtree mining. *Proceedings of AAAI07*, 2007.

[26] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. *Proceedings of AAAI07*, 2007.

[27] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. *Proceedings of AAAI06*, 2006.

[28] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 2006.

[29] D. Sanchez and A. Moreno. Web-scale taxonomy learning. *Proceedings of Workshop on Extending and Learning Lexical Ontologies using Machine Learning, ICML05*, 2005.

[30] P. Singla and P. Domingos. Discriminative training of markov logic networks. *Proceedings AAAI05*, 2005.

[31] R. Snow, D. Jurafsky, and A. Ng. Semantic taxonomy induction from heterogenous evidence. *Proceedings of ACL06*, 2006.

[32] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. *Proceedings of WWW07*, 2007.

[33] P. Velardi, R. Navigli, A. Cucchiarelli, and F. Neri. Evaluation of ontolearn, a methodology for automatic learning of domain ontologies. *Ontology Learning and Population*, 2005.

[34] F. Wu, R. Hoffmann, and D. Weld. Information extraction from Wikipedia: Moving down the long tail. *In Submission*, 2008.

[35] F. Wu and D. Weld. Autonomouslly semantifying wikipedia. *Proceedings of CIKM07*, Lisbon, Porgugal, 2007.

[36] K. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *Proceedings of SIGCHI03*, 2003.