Unsupervised techniques for discovering ontology elements from Wikipedia article links

Zareen Syed
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250, USA
zarsyed1@umbc.edu

Tim Finin
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250, USA
finin@umbc.edu

Abstract

We present an unsupervised and unrestricted approach to discovering an infobox like ontology by exploiting the inter-article links within Wikipedia. It discovers new slots and fillers that may not be available in the Wikipedia infoboxes. Our results demonstrate that there are certain types of properties that are evident in the link structure of resources like Wikipedia that can be predicted with high accuracy using little or no linguistic analysis. The discovered properties can be further used to discover a class hierarchy. Our experiments have focused on analyzing people in Wikipedia, but the techniques can be directly applied to other types of entities in text resources that are rich with hyperlinks.

1 Introduction

One of the biggest challenges faced by the Semantic Web vision is the availability of structured data that can be published as RDF. One approach is to develop techniques to translate information in spreadsheets, databases, XML documents and other traditional data formats into RDF (Syed et al. 2010). Another is to refine the technology needed to extract structured information from unstructured free text (McNamee and Dang, 2009).

For both approaches, there is a second problem that must be addressed: do we start with an ontology or small catalog of ontologies that will be used to encode the data or is extracting the right ontology part of the problem. We describe exploratory work on a system that can discover ontological elements as well as data from a free text with embedded hyperlinks.

Wikipedia is a remarkable and rich online encyclopedia with a wealth of general knowledge about varied concepts, entities, events and facts in the world. Its size and coverage make it a valuable resource for extracting information about different entities and concepts. Wikipedia contains both free text and structured information related to concepts in the form of infoboxes, category hierarchy and inter-article links. Infoboxes are the most structured form and are composed of a set of subject-attribute-value triples that summarize or highlight the key features of the concept or subject of the article. Resources like DBpedia (Auer et al., 2007) and Freebase (Bollacker et al., 2007) have harvested this structured data and have made it available as triples for semantic querying.

While infoboxes are a readily available source of structured data, the free text of the article contains much more information about the entity. Barker et al. (2007) unified the state of the art approaches in natural language processing and knowledge representation in their prototype system for understanding free text. Text resources which are rich in hyperlinks especially to knowledge based resources (such as encyclopedias or dictionaries) have additional information encoded in the form of links, which can be used to complement the existing systems for text understanding and knowledge discovery. Furthermore, systems such as Wikify (Mihalcea and Csomai, 2007) can be employed to link words in free text to knowledge resources like Wikipedia and thus enrich the free text with hyperlinks.

We describe an approach for unsupervised ontology discovery from links in the free text of the Wikipedia articles, without specifying a relation or set of relations in advance. We first identify candidate slots and fillers for an entity, then classify en-

Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading, pages 78–86,
Los Angeles, California, June 2010. ©2010 Association for Computational Linguistics

tities and finally derive a class hierarchy. We have evaluated our approach for the *Person* class, but it can be easily generalized to other entity types such as organizations, places, and products.

The techniques we describe are not suggested as alternatives to natural language understanding or information extraction, but as a source for additional evidence that can be used to extract ontological elements and relations from the kind of text found in Wikipedia and other heavily-linked text collections. This approach might be particularly useful in "slot fillings" tasks like the one in the Knowledge Base Population track (McNamee and Dang, 2010) at the 2009 Text Analysis Conference. We see several contributions that this work has to offer:

- Unsupervised and unrestricted ontology discovery. We describe an automatic approach that does not require a predefined list of relations or training data. The analysis uses inter-article links in the text and does not depend on existing infoboxes, enabling it to suggest slots and fillers that do not exist in any extant infoboxes.
- Meaningful slot labels. We use WordNet (Miller et al., 1990) nodes to represent and label slots enabling us to exploit WordNet's hypernym and hyponym relations as a property hierarchy.
- Entity classification and class labeling. We introduce a new feature set for entity classification, i.e. the discovered ranked slots, which performs better than other feature sets extracted from Wikipedia. We also present an approach for assigning meaningful class label vectors using WordNet nodes.
- Deriving a class hierarchy. We have developed an approach for deriving a class hierarchy based on the ranked slot similarity between classes and the label vectors.

In the remainder of the paper we describe the details of the approach, mention closely related work, present and discuss preliminary results and provide some conclusions and possible next steps.

2 Approach

Figure 1 shows our ontology discovery framework and its major steps. We describe each step in the rest of this section.

2.1 Discovering Candidate Slots and Fillers

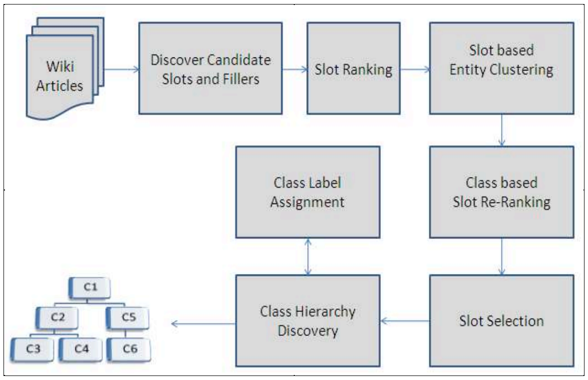
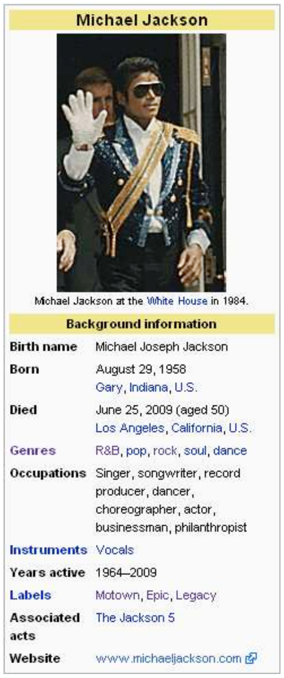

Wiki Articles
Discover Candidate Slots and Fillers
Slot Ranking
Slot based Entity Clustering
Class based Slot Re-Ranking
Class Label Assignment
Class Hierarchy Discovery
Slot Selection
C1
C2
C5
C3
C4
C6

Figure 1: The ontology discovery framework comprises a number of steps, including candidate slot and filler discovery followed by slot ranking, slot selection, entity classification, slot re-ranking, class labeling, and class hierarchy discovery.

Most Wikipedia articles represent a concept, i.e., a generic class of objects (e.g., Musician), an individual object (e.g., Michael_Jackson), or a generic relation or property (e.g., age). Inter-article links within Wikipedia represent relations between concepts. In our approach we consider the linked concepts as candidate fillers for slots related to the primary article/concept. There are several cases where the filler is subsumed by the slot label for example, the infobox present in the article on "Michael_Jackson" (Figure 2) mentions *pop, rock* and *soul* as fillers for the slot *Genre* and all three of these are a type of *Genre*. The *Labels* slot contains fillers such as *Motown, Epic* and *Legacy* which are all Record Label Companies. Based on this observation, we discover and exploit "isa" relations between fillers (linked concepts) and WordNet nodes to serve as candidate slot labels.

In order to find an "isa" relation between a concept and a WordNet synset we use manually created mappings by DBpedia, which links about 467,000 articles to synsets. However, Wikipedia has more than two million articles[1], therefore, to map any remaining concepts we use the automatically generated mappings available between WordNet synsets and Wikipedia categories (Ponzetto and Navigli, 2009). A single Wikipedia article might have multiple categories associated with it and therefore multiple WordNet synsets. Wikipedia's category system serves more as a way to tag articles and facilitate navigation rather than

[1] This estimate is for the English version and does not include redirects and administrative pages such as disambiguation pages.



Michael Jackson

Michael Jackson at the White House in 1984.

Background information	
Birth name	Michael Joseph Jackson
Born	August 29, 1958 Gary, Indiana, U.S.
Died	June 25, 2009 (aged 50) Los Angeles, California, U.S.
Genres	R&B, pop, rock, soul, dance
Occupations	Singer, songwriter, record producer, dancer, choreographer, actor, businessman, philanthropist
Instruments	Vocals
Years active	1964–2009
Labels	Motown, Epic, Legacy
Associated acts	The Jackson 5
Website	www.michaeljackson.com

Figure 2. The Wikipedia infobox for the Michael_Jackson article has a number of slots from appropriate infobox templates.

to categorize them. The article on Michael Jordan, for example, has 36 categories associated with it. In order to select an individual WordNet synset as a label for the concept's type, we use two heuristics:

- **Category label extraction.** Since the first sentence in Wikipedia articles usually defines the concept, we extract a category label from the first sentence using patterns based on POS tags similar to Kazama and Torisawa (2007).
- **Assign matching WordNet synset.** We consider all the WordNet synsets associated with the categories of the article using the category to WordNet mapping (Ponzetto and Navigli, 2009) and assign the WordNet synset if any of the words in the synset matches with the extracted category label. We repeat the process with hypernyms and hyponyms of the synset up to three levels.

2.2 Slot Ranking

All slots discovered using outgoing links might not be meaningful, therefore we have developed techniques for ranking and selecting slots. Our approach is based on the observation that entities of the same type have common slots. For example, there is a set of slots common for *musical artists* whereas, a different set is common for *basketball players*. The Wikipedia infobox templates based on classes also provide a set of properties or slots to use for particular types of entities or concepts.

In case of people, it is common to note that there is a set of slots that are *generalized*, i.e., they are common across all types of persons. Examples are *name, born,* and *spouse*. There are also sets of *specialized* slots, which are generally associated with a given profession. For example, the slots for basketball players have information for basketball related activities and musical artists have slots with music related activities. The slots for "Michael_Jordan" include *Professional Team(s), NBA Draft, Position(s)* and slots for "Michael_Jackson" include *Genres, Instruments* and *Labels*.

Another observation is that people engaged in a particular profession tend to be linked to others within the same profession. Hence the maxim "A man is known by the company he keeps." For example, basketball players are linked to other basketball players and politicians are linked to other politicians. We rank the slots based on the number of linked persons having the same slots. We generated a list of person articles in Wikipedia by getting all Wikipedia articles under the *Person* type in Freebase[2]. We randomly select up to 25 linked persons (which also link back) and extract their candidate slots and vote for a slot based on the number of times it appears as a slot in a linked person normalized by the number of linked persons to assign a slot score.

2.3 Entity Classification and Slot Re-Ranking

The ranked candidate slots are used to classify entities and then further ranked based on number of times they appear among the entities in the cluster. We use complete link clustering using a simple slot similarity function:

$$sim_{slot}(p_i, p_j) = \cos(slot(p_i), slot(p_j))$$

This similarity metric for slots is computed as the cosine similarity between tf.idf weighted slot vectors, where the slot score represents the term fre-

[2] We found that the Freebase classification for Person was more extensive that DBpedia's in the datasets available to us in early 2009.

quency component and the inverse document frequency is based on the number of times the slot appears in different individuals.

We also collapsed location expressing slots (*country, county, state, district, island* etc.) into the slot labeled *location* by generating a list of location words from WordNet as these slots were causing the persons related to same type of geographical location to cluster together.

After clustering, we re-score the slots based on number of times they appear among the individuals in the cluster normalized by the cluster size. The output of clustering is a vector of scored slots associated with each cluster.

2.4 Slot Selection

The slot selection process identifies and filters out slots judged to be irrelevant. Our intuition is that *specialized* slots or attributes for a particular entity type should be somehow related to each other. For example, we would expect attributes like *league, season* and *team* for basketball players and *genre, label, song* and *album* for musical artists. If an attribute like *album* appears for basketball players it should be discarded as it is not related to other attributes.

We adopted a clustering approach for finding attributes that are related to each other. For each pair of attributes in the slot vector, we compute a similarity score based on how many times the two attribute labels appear together in Wikipedia person articles within a distance of 100 words as compared to the number of times they appear in total and weigh it using weights of the individual attributes in the slot vector. This metric is captured in the following equation, where *Df* is the document frequency and *wt* is the attribute weight.

$$sim_{attr}(a_i, a_j) = wt(a_i) \times wt(a_j) \times \frac{Df(a_i, a_j, 100)}{Df(a_i) + Df(a_j)}$$

Our initial experiments using single and complete link clustering revealed that single link was more appropriate for slot selection. We got clusters at a partition distance of 0.9 and selected the largest cluster from the set of clusters. In addition, we also added any attributes exceeding a 0.4 score into the set of selected attributes. Selected ranked slots for Michael Jackson are given in Table 1.

Slot	Score	Fillers Example
Musician	1.00	ray_charles, sam_cooke ...
Album	0.99	bad_(album), ...
Location	0.97	gary, indiana, chicago, …
Music_genre	0.90	pop_music, soul_music, ...
Label	0.79	a&m_records, epic_records, ...
Phonograph_ record	0.67	give_in_to_me, this_place_hotel …
Act	0.59	singing
Movie	0.46	moonwalker …
Company	0.43	war_child_(charity), …
Actor	0.41	stan_winston, eddie_murphy,
Singer	0.40	britney_spears, …
Magazine	0.29	entertainment_weekly,…
Writing_style	0.27	hip_hop_music
Group	0.21	'n_sync, RIAA
Song	0.20	d.s._(song) …

Table 1: Fifteen slots were discovered for musician Michael Jackson along with scores and example fillers.

2.5 Class Labeling

Assigning class labels to clusters gives additional information about the type of entities in a cluster. We generate a cluster label vector for each cluster which represents the type of entities in the cluster. We compute a list of person types by taking all hyponyms under the corresponding person sense in WordNet. That list mostly contained the professions list for persons such as basketball player, president, bishop etc. To assign a WordNet type to a person in Wikipedia we matched the entries in the list to the words in the first sentence of the person article and assigned it the set of types that matched. For example, for Michael Jordan the matching types found were *basketball_player, businessman* and *player*.

We assigned the most frequent sense to the matching word as followed by Suchanek et al. (2008) and Wu and Weld (2008), which works for majority of the cases. We then also add all the hypernyms of the matching types under the Person node. The vector for Michael Jordan has entries *basketball_player, athlete, businessperson, person, contestant, businessman* and *player*. After getting matching types and their hypernyms for all the members of the cluster, we score each type based on the number of times it occurs in its members normalized by the cluster size. For example for one of the clusters with 146 basketball players we got the following label vector: {*player*:0.97, *contestant*:0.97, *athlete*:0.96, *basketball_player*:0.96}. To select an individual label for a class we can pick the label with the highest score (the most generalized label) or the most specialized label having a score above a given threshold.

2.6 Discovering Class Hierarchy

We employ two different feature sets to discover the class hierarchy, i.e., the selected slot vectors and the class label vectors and combine both functions using their weighted sum. The similarity functions are described below.

The common slot similarity function is the cosine similarity between the common slot tf.idf vectors, where the slot score represents the tf and the idf is based on the number of times a particular slot appears in different clusters at that iteration. We re-compute the idf term in each iteration. We define the *common slot* tf.idf vector for a cluster as one where we assign a non-zero weight to only the slots that have non-zero weight for all cluster members. The label similarity function is the cosine similarity between the label vectors for clusters. The hybrid similarity function is a weighted sum of the common slot and label similarity functions. Using these similarity functions we apply complete link hierarchical clustering algorithm to discover the class hierarchy.

$$
\begin{aligned}
&sim_{com\_slot}(c_i,c_j) = \cos(com\_slot(c_i),com\_slot(c_j))\\
&sim_{label}(c_i,c_j) = \cos(label(c_i),label(c_j))\\
&sim_{hyb}(c_i,c_j) = w_c \times sim_{com\_slot}(c_i,c_j) + w_l \times sim_{label}(c_i,c_j)\\
&w_c + w_l = 1
\end{aligned}
$$

3 Experiments and Evaluation

For our experiments and evaluation we used the Wikipedia dump from March 2008 and the DBpedia infobox ontology created from Wikipedia infoboxes using hand-generated mappings (Auer et al., 2007). The *Person* class is a direct subclass of the owl:Thing class and has 21 immediate subclasses and 36 subclasses at the second level. We used the persons in different classes in DBpedia ontology at level two to generate data sets for experiments.

There are several articles in Wikipedia that are very small and have very few out-links and in-links. Our approach is based on the out-links and availability of information about different related things on the article, therefore, in order to avoid data sparseness, we randomly select articles with greater than 100 in-links and out-links, at least 5KB page length and having at least five links to entities of the same type that link back (in our case persons).

We first compare our slot vector features with other features extracted from Wikipedia for entity classification task and then evaluate their accuracy. We then discover the class hierarchy and compare the different similarity functions.

3.1 Entity Classification

We did some initial experiments to compare our ranked slot features with other feature sets extracted from Wikipedia. We created a dataset composed of 25 different classes of Persons present at level 2 in the DBpedia ontology by randomly selecting 200 person articles from each class. For several classes we got less than 200 articles which fulfilled our selection criteria defined earlier. We generated twelve types of feature sets and evaluated them using ground truth from DBpedia ontology.

We compare tf.idf vectors constructed using twelve different feature sets: (1) Ranked slot features, where tf is the slot score; (2) Words in first sentence of an article; (3) Associated categories; (4) Assigned WordNet nodes (see section 2.2); (5) Associated categories tokenized into words; (6) Combined Feature Sets 1 to 5 (All); (7-11) Feature sets 7 to 11 are combinations excluding one feature set at a time; (12) Unranked slots where tf is 1 for all slots. We applied complete link clustering and evaluated the precision, recall and F-measure at different numbers of clusters ranging from one to 100. Table 2 gives the precision, recall and number of clusters where we got the maximum F-measure using different feature sets.

No.	Feature Set	k	P	R	F
1	Ranked Slots	40	0.74	0.72	0.73
2	First Sentence	89	0.07	0.53	0.12
3	Categories	1	0.05	1.00	0.10
4	WordNet Nodes	87	0.40	0.22	0.29
5	(3 tokenized)	93	0.85	0.47	0.60
6	All (1 to 5)	68	0.87	0.62	0.72
7	(All – 5)	82	0.79	0.46	0.58
8	(All – 4)	58	0.78	0.63	0.70
9	(All – 3)	53	0.76	0.65	0.70
10	(All – 2)	58	0.88	0.63	0.74
11	(All – 1)	57	0.77	0.60	0.68
12	(1 unranked)	34	0.57	0.65	0.61

Table 2: Comparison of the precision, recall and F-measure for different feature sets for entity classification. The k column shows the number of clusters that maximized the F score.

Feature set 10 (all features except feature 2) gave the best F-measure i.e. 0.74, whereas, feature set 1 (ranked slots only) gave the second best F-measure i.e. 0.73 which is very close to the best result. Feature set 12 (unranked slots) gave a lower F-measure i.e. 0.61 which shows that ranking or weighing slots based on linked entities of the same type performs better for classification.

3.2 Slot and Filler Evaluation

To evaluate our approach to finding slot fillers, we focused on DBpedia classes two levels below Person (e.g., Governor and FigureSkater). We randomly selected 200 articles from each of these classes using the criteria defined earlier to avoid data sparseness. Classes for which fewer than 20 articles were found were discarded. The resulting dataset comprised 28 classes and 3810 articles[3].

We used our ranked slots tf.idf feature set and ran a complete link clustering algorithm producing clusters at partition distance of 0.8. The slots were re-scored based on the number of times they appeared in the cluster members normalized by the cluster size. We applied slot selection over the re-scored slots for each cluster. In order to evaluate our slots and fillers we mapped each cluster to a DBpedia class based on the maximum number of members of a particular DBpedia class in our cluster. This process predicted 124 unique properties for the classes. Of these, we were able to manually align 46 to properties in either DBpedia or Freebase for the corresponding class. We initially tried to evaluate the discovered slots by comparing them with those found in the ontologies underlying DBpedia and Freebase, but were able to find an overlap in the subject and object pairs for very few properties.

No.	Property	Accuracy
1	automobile_race	1.00
2	championship	1.00
3	expressive_style	1.00
4	fictional_character	1.00
5	label	1.00
6	racetrack	1.00
7	team_sport	1.00
8	writing_style	1.00
9	academic_degree	0.95
10	album	0.95
11	book	0.95
12	contest	0.95
13	election	0.95
14	league	0.95
15	phonograph_record	0.95
16	race	0.95
17	tournament	0.94
18	award	0.90
19	movie	0.90
20	novel	0.90
21	school	0.90
22	season	0.90
23	serial	0.90
24	song	0.90
25	car	0.85
26	church	0.85
27	game	0.85
28	musical_instrument	0.85
29	show	0.85
30	sport	0.85
31	stadium	0.85
32	broadcast	0.80
33	telecast	0.80
34	hockey_league	0.75
35	music_genre	0.70
36	trophy	0.70
37	university	0.65
38	character	0.60
39	disease	0.60
40	magazine	0.55
41	team	0.50
42	baseball_club	0.45
43	club	0.45
44	party	0.45
45	captain	0.30
46	coach	0.25
	Avg. Accuracy:	0.81

Table 3: Manual evaluation of discovered properties

We randomly selected 20 subject object pairs for each of the 46 properties from the corresponding classes and manually judged whether or not the relation was correct by consulting the correspond-

[3] For some of the classes, fewer than the full complement of 200 articles were found.

Similarity Function	k (L=2)	F (L=2)	k (L=1)	F (L=1)
sim_{slot}	56	0.61	13	0.55
sim_{com_slot}	74	0.61	15	0.65
sim_{label}	50	0.63	10	0.76
sim_{hyb} w_c=w_l=0.5	59	0.63	10	0.76
sim_{hyb} w_c=0.2, w_l=0.8	61	0.63	8	0.79

Table 4: Evaluation results for class hierarchy prediction using different similarity functions.

ing Wikipedia articles (Table 3). The average accuracy for the 46 relations was 81%.

3.3 Discovering Class Hierarchy

In order to discover the class hierarchy, we took all of the clusters obtained earlier at partition distance of 0.8 and their corresponding slot vectors after slot selection. We experimented with different similarity functions and evaluated their accuracy by comparing the results with the DBpedia ontology. A complete link clustering algorithm was applied using different settings of the similarity functions and the resulting hierarchy compared to DBpedia's Person class hierarchy. Table 4 shows the highest F measure obtained for Person's immediate sub-classes (L1), "sub-sub-classes" (L2) and the number of clusters (k) for which we got the highest F-measure using a particular similarity function.

The highest F-measure both at level 2 (0.63) and level 1 (0.79) was obtained by sim_{hyb} with w_c=0.2, w_l=0.8 and also at lowest number of clusters at L1 (k=8). The sim_{hyb} (w_c=w_l=0.5) and sim_{label} functions gave almost the same F-measure at both levels. The sim_{com_slot} function gave better performance at L1 (F=0.65) than the base line sim_{slot} (F=0.55) which was originally used for entity clustering. However, both these functions gave the same F-measure at L2 (F=0.61).

4 Discussion

In case of property evaluation, properties for which the accuracy was 60% or below include *coach, captain, baseball_club, club, party, team* and *magazine*. For the *magazine* property (corresponding to *Writer* and *ComicsCreator* class) we observed that many times a magazine name was mentioned in an article because it published some news about a person rather than that person contributing any article in that magazine. For all the remaining properties we observed that these were related to some sort of competition. For example, a person played against a *team, club, coach* or *captain*. The political *party* relation is a similar case, where articles frequently mention a politician's party affiliation as well as significant opposition parties. For such properties, we need to exploit additional contextual information to judge whether the person competed "for" or "against" a particular *team, club, coach* or *party*. Even if the accuracy for fillers for such slots is low, it can still be useful to discover the kind of slots associated with an entity.

We also observed that there were some cases where the property was related to a family member of the primary person such as for *disease, school* and *university*. Certain other properties such as *spouse, predecessor, successor,* etc. require more contextual information and are not directly evident in the link structure. However, our experiments show that there are certain properties that can be predicted with high accuracy using the article links only and can be used to enrich the existing infobox ontology or for other purposes.

While our work has mostly experimented with person entities, the approach can be applied to other types as well. For example, we were able to discover *software* as a candidate slot for companies like Microsoft, Google and Yahoo!, which appeared among the top three ranked slots using our slot ranking scheme and corresponds to the *products* slot in the infoboxes of these companies.

For class hierarchy discovery, we have exploited the *specialized* slots after slot selection. One way to incorporate *generalized* slots in the hierarchy is to consider all slots for class members (without slot selection) and recursively propagate the common slots present at any level to the level above it. For example, if we find the slot *team* to be common for different types of Athletes such as basketball players, soccer players etc. we can propagate it to the Athlete class, which is one level higher in the hierarchy.

5 Related Work

Unsupervised relation discovery was initially introduced by Hasegawa et al. (2004). They developed an approach to discover relations by clustering pairs of entities based on intervening words represented as context vectors. Shinyama and Sekine (2006) generated basic patterns using parts of text syntactically connected to the entity and then generated a basic cluster composed of a set of events having the same relation.

Several approaches have used linguistic analysis to generate features for supervised or unsupervised relation extraction (Nguyen et al., 2007; Etzioni et al., 2008; Yan et al., 2009). Our approach mainly exploits the heavily linked structure of Wikipedia and demonstrates that there are several relations that can be discovered with high accuracy without the need of features generated from a linguistic analysis of the Wikipedia article text.

Suchanek et al. (2008) used Wikipedia categories and infoboxes to extract 92 relations by applying specialized heuristics for each relation and incorporated the relations in their YAGO ontology, whereas our techniques do not use specialized heuristics based on the type of relation. Kylin (Weld et al., 2008) generated infoboxes for articles by learning from existing infoboxes, whereas we can discover new fillers for several existing slots and also discover new slots for infoboxes. KOG (Wu and Weld, 2008) automatically refined the Wikipedia infobox ontology and integrated Wikipedia's infobox-class schemata with WordNet. Since we already use the WordNet nodes for representing slots, it eliminates the need for several of KOG's infobox refinement steps.

While YAGO, Kylin and KOG all rely on relations present in the infoboxes, our approach can complement these by discovering new relations evident in inter-article links in Wikipedia. For example, we could add slots like *songs* and *albums* to the infobox schema for *Musical Artists*, *movies* for the *Actors* infobox schema, and *party* for the *Politicians* schema.

6 Conclusions and Future Work

People have been learning by reading for thousands of years. The past decade, however, has seen a significant change in the way people read. The developed world now does much of its reading online and this change will soon be nearly universal. Most online content is read as hypertext via a Web browser or custom reading device. Unlike text, hypertext is semi-structured information, especially when links are drawn from global namespace, making it easy for many documents to link unambiguously to a common referent.

The structured component of hypertext augments the information in its plain text and provides an additional source of information from which both people and machines can learn. The work described in this paper is aimed at learning useful information, both about the implicit ontology and facts, from the links embedded in collection of hypertext documents.

Our approach is fully unsupervised and does not require having a pre-defined catalogue of relations. We have discovered several new slots and fillers that are not present in existing Wikipedia infoboxes and also a scheme to rank the slots based on linked entities of the same type. We compared our results with ground truth from the DBpedia infobox ontology and Freebase for the set of properties that were common and manually evaluated the accuracy of the common properties. Our results show that there are several properties that can be discovered with high accuracy from the link structure in Wikipedia and can also be used to discover a class hierarchy.

We plan to explore the discovery of slots from non-Wikipedia articles by linking them to Wikipedia concepts using existing systems like Wikify (Mihalcea and Csomai, 2007). Wikipedia articles are encyclopedic in nature with the whole article revolving around a single topic or concept. Consequently, linked articles are a good source of properties and relations. This might not be the case in other genres, such as news articles, that discuss a number of different entities and events. One way to extend this work to other genres is by first detecting the entities in the article and then only processing links in sentences that mention an entity to discover its properties.

Acknowledgements

The research described in this paper was supported in part by a Fulbright fellowship, a gift from Microsoft Research, NSF award IIS-0326460 and the Johns Hopkins University Human Language Technology Center of Excellence.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In Proceedings of the 6th International Semantic Web Conference: 11–15.

Ken Barker et al. 2007. Learning by reading: A prototype system, performance baseline and lessons learned, Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI Press.

K. Bollacker, R. Cook, and P. Tufts. 2007. Freebase: A Shared Database of Structured General Human Knowledge. Proceedings of the National Conference on Artificial Intelligence (Volume 2): 1962-1963.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. Communications of the ACM 51, 12 (December): 68-74.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics: 415-422.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: 698–707.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In Proceedings of the 2009 Text Analysis Conference. National Institute of Standards and Technology, November.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In Proceedings of the 16th ACM Conference on Information and Knowledge Management: 233–242.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. WordNet: An on-line lexical database. International Journal of Lexicography, 3:235–244.

Dat P. T. Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Subtree mining for relation extraction from Wikipedia. In Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics:125–128.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from Wikipedia and WordNet. Web Semantics, 6(3):203–217.

Zareen Syed, Tim Finin, Varish Mulwad and Anupam Joshi. 2010. Exploiting a Web of Semantic Data for Interpreting Tables, Proceedings of the Second Web Science Conference.

Simone P. Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence: 2083–2088.

Yusuke Shinyama and Satoshi Sekine. 2006. Pre-emptive information extraction using unrestricted relation discovery. In Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics:.

Daniel S. Weld, Raphael Hoffmann, and Fei Wu. 2008. Using Wikipedia to bootstrap open information extrac-tion. SIGMOD Record, 37(4): 62–68.

Fei Wu and Daniel S. Weld. 2008. Automatically refining the Wikipedia infobox ontology. In Proceedings of the 17th International World Wide Web Conference, pages 635–644.

Wikipedia. 2008. Wikipedia, the free encyclopedia.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining Wikipedia texts using information from the web. In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics: Volume 2: 1021–1029.