Discovering and Exploring Relations on the Web

Ndapandula Nakashole, Gerhard Weikum, Fabian Suchanek
Max Planck Institute for Informatics, Saarbruecken, Germany
{nnakasho,weikum,suchanek}@mpi-inf.mpg.de

1. INTRODUCTION AND OVERVIEW

We propose a demonstration of PATTY, a system for learning semantic relationships from the Web.

Motivation. There is increasing interest in imposing entity-relationship-oriented (ER) views on Web contents. State-of-the-art approaches can detect and disambiguate named entities in text or tables, and extract binary relations between entities based on patterns in textual or semistructured contents. These advances have enabled the automatic construction of large knowledge bases such as *dbpedia.org*, *freebase.com*, *trueknowledge.com*, or *yago-knowledge.org*, and they have greatly enhanced the "semantic awareness" of Web and enterprise search as well as question answering (e.g., [2]).

Problem Statement. A fundamental obstacle in these endeavors is that relationships between entities are expressed merely in latent form on the Web, using different paraphrases and patterns. For example, sources may use the verbal phrases "received" or "was honored with" to say that a person won an award. Understanding the semantic equivalence of phrases and mapping these diverse phrases into canonicalized representations is the core challenge in relational information extraction (IE). This problem arises in seed-based distantly supervised IE, with explicitly specified target relations, as well as in Open IE, where the relations themselves are a priori unknown and need to be discovered in an unsupervised manner. Comprehensively gathering and systematically organizing patterns for an *open set of relations* is the problem addressed by our system PATTY, presented in this demo paper.

Example. The fact that Natalie Portman has won the Oscar should be recognized as an instance of the *hasWonPrize* relation (with type signature *person* × *award*), but it occurs in very diverse forms: "Portman was honored with an Oscar", "... received ...", "... thanked the academy for ...", "Awards received: Academy Award 2011" (in a Web table or list), "Prizes: Oscar for Best Actress", etc. In addition, a paraphrase like "received" could also denote different relations such as *almaMater* ("received her Ph.D. degree from"), *critizedBy* ("not well received by"), or *meets* ("received by the queen"). On the other hand, a sentence such as "Bonham Carter was disappointed that her nomination for the Oscar was not successful" may easily be incorrectly interpreted as denoting a pattern for the *hasWonPrize* relation, whereas it actually indicates a different relation *nominatedForPrize*. Ideally, the analysis of patterns should reveal that *hasWonPrize* implies *nominatedForPrize*.

Prior Work. Pattern-based IE has a long history in natural language processing (NLP) and Web mining. Recently, Web-scale systems like NELL [1], Probase [9], Prospera [6], and ReVerb [3] have developed sophisticated machineries for distilling so-called *lexico-syntactic patterns* such as on verbal phrases or noun phrases with prepositions. Such patterns generalize surface strings into combinations of words and word-category tags, so-called part-of-speech (POS) tags or coarse-grained lexical categories such as *location* or *date*. For example, "disappointed * PRP$ nomination" is a pattern with a wildcard symbol * and a POS tag PRP$ standing for possessive personal pronouns such as "his" or "her". In addition, there is considerable work on identifying relations and attributes from Web tables [5] and [10] have addressed the mining of equivalent patterns, in order to discover new relations, based on clustering. These methods are not suitable for Web scale, as they involve building large matrices or expensive inference on latent models. Moreover, the issue of identifying subsumptions between patterns has been disregarded. Among these prior works, only ReVerb and NELL have made their patterns publicly available. However, the ReVerb patterns for Open IE are fairly noisy and connect noun phrases rather than entities. NELL is limited to a few hundred pre-specified relations. None of the prior approaches knows the ontological types of patterns, to reveal, e.g., that *hasWonPrize* holds between a person and an award.

Our Approach and Contribution. Our goal is to systematically harvest textual patterns from text corpora, to group synonymous patterns into pattern synsets, and to order these synsets into a subsumption hierarchy. For this purpose, we make use of a generalized notion of *ontologically typed patterns*. These patterns have a type signature for the entities that they connect, as in "received"[*person* × *award*]. We derive the type signatures through the use of a dictionary of entity-class pairs, as, e.g., provided by knowledge bases like YAGO, Freebase, or DBpedia. This approach gives us three advantages over prior work: First, we can better group equivalent patterns using the compatibility of their type signatures as an additional cue. For example, *actor* × *award* and *musician* × *award* generalize into *person* × *award*. Second, we can discriminate patterns that can denote different relations based on their type signatures. For example, "covered" can refer to *musician* × *song* or to *journalist* × *event* on a television program or in an article. Third, we can identify subsumptions between relational patterns, based on the overlap between their associated entity pairs. For example, "covered" is subsumed by "performed" for the same type signature *musician* × *song*. Our methods to this end are based on frequent sequence mining, suitably extended to our setting and to Map-Reduce-based distributed computation. We have run our methods on large data like the full

Proceedings of the VLDB Endowment, Vol. 5, No. 1

text of Wikipedia and the 500-million-pages Web corpus ClueWeb. The resulting pattern collections contain about 350,000 relational patterns with an average accuracy around 85% (estimated by extensive sampling). The PATTY system and all this data is available at `http://www.mpi-inf.mpg.de/yago-naga/patty/`.

Applications. PATTY is to relationships what WordNet is to entity classes. It organizes a huge number of relational patterns into sets of synonymous patterns, and into a subsumption hierarchy. This structure has benefits in a variety of applications: First, it can boost IE and knowledge-base population tasks by its rich and clean repository of paraphrases for the relations. Second, it can improve Open IE by associating type signatures with patterns. Third, it can help discovering "Web witnesses" when assessing the truthfulness of search results or statements in social media [4]. Last, it provides paraphrases for detecting relationships in keyword queries, thus lifting keyword search to the ER level. This can help with understanding questions and text snippets in natural-language QA.

Our demonstration allows the user to explore these possibilities in three ways: (1) Using PATTY as a thesaurus to find paraphrases for relations (2) using PATTY as a simple kind of QA system to query the database without having to know the schema and (3) exploring the relationships between entities, as expressed in the textual sources.

2. THE PATTY SYSTEM

The PATTY system processes large text corpora (e.g., the full text of Wikipedia, news archives, or Web crawls) to build a taxonomy of textual patterns. It is backed by a knowledge base of semantically typed entities. For the latter, we use either YAGO [8] or Freebase[1]: YAGO has about 350,000 classes derived from Wikipedia categories and integrated with WordNet classes; Freebase has a handcrafted type system with 85 topical domains as top tier and about 2000 entity classes as a second tier. PATTY works in four stages: 1) pattern extraction 2) pattern typing, 3) synset generation, and 4) subsumption mining. We will now explain these stages. For the details on stages 1-3, see [7], we focus here on the stage 4.

Pattern Extraction. A pattern is a surface string that occurs between a pair of entities in a sentence. For example, the string *"was governer of"* is a pattern that can appear between the entity *Bill Clinton* and the entity *Arkansas*. We find such patterns by identifying sentences noun phrases with a part-of-speech tagger. We transform nounphrases to entities by means of a dictionary of surface names that the input knowledge base provides. We represent patterns as sequences of frequent n-grams by using frequent itemset mining. We define the *support set*, $SS(p_k)$ of a pattern as the set of entity pairs that occur with the pattern. For example, the pattern $\langle$*was governor of*$\rangle$, has a support set containing entity pairs such as *(Bill Clinton, Arkansas)*, *(Janet Napolitano, Arizona)*, etc.

Pattern Typing. From the type-agnostic patterns we generate typed patterns by attaching a *type signature* to each pattern. A type signature is a type pair of the form *Type1* $\times$ *Type2*, where *Type1* and *Type2* are the semantic types for the first and second entities, respectively. We obtain the semantic types of the entities from our knowledge base. For example, the entity *Bill Clinton* might have types *politician, president, person*. Suppose the pattern $\langle$*was governor of*$\rangle$ frequently occurs with entities of type *Politician* for the first entity and entities of type *State* for the second entity, then we produce a typed pattern $\langle$*was governor of*$\rangle$[*Politician* $\times$ *State*].

Synset Generation. We determine synonyms among typed patterns in two ways. First, we identify syntactic synonyms by grouping patterns which are syntactically similar, e.g., differing only in specific choices of pronouns or adjectives (e.g., $\langle$*was the first female governor of*$\rangle$ vs. $\langle$*was the best governor of*$\rangle$). Second, we determine semantic synonyms, as follows: for two patterns with nearly identical support sets, we infer that the patterns are synonyms. We group synonymous typed patterns into synonym sets, *pattern synsets*.

Subsumption Mining. At this stage, we mine subsumption (hypernymy/hyponymy) relations between pattern synsets. Each synset P_k has a set of entity pairs that support it, $SS(P_k)$. We say that a synset P_k is *subsumed by* another synset P_l , if $SS(P_k) \subseteq SS(P_l)$. However, requiring complete containment of P_k in P_l is too constraining as there is always noise in the data and the data is not necessarily complete. Thus, we relax subsumptions to *approximate subsumptions* by allowing that P_k contains in its support-set a few entity-pairs that are not in P_l. We use a probabilistic variant of Jaccard coefficients to quantify the degree of containment between two support sets.

Mining taxonomic subsumptions from our synsets is not a trivial endeavor, because a quadratic comparison of each and every synset to every other synset would be prohibitively slow. Therefore, we developed a Map-Reduce algorithm for this purpose. As input, our algorithm requires a set of pattern synsets, each with their support set. As output, we compute a DAG of pattern synset subsumptions. We first invert the synset support data. Instead of providing, for a synset, all entity-pairs that occur with it, we provide for an entity pair all the synsets that it occurs with. This can be achieved by a Map-Reduce algorithm that is similar to a standard text indexing Map-Reduce algorithm.

From these data, we have to compute co-occurrence counts of pattern synsets, i.e., the number of entity-pairs that the supports of two patterns have in common. Our Map-Reduce algorithm for this purpose is as follows: The mappers emit pairs of synsets that co-occur for every synset they occur with. The reducers aggregate co-occurrence information, to effectively output the sizes of the set intersection of the possible subsumptions. From these, we can compute approximate degrees of subsumption as described above. This produces a weighted graph of subsumption relations between the synsets.

The graph of subsumption relations might contain cycles, which have to be eliminated. We ideally want to remove the minimal total number of subsumptions whose removal results in a DAG. This task is related to the minimum feedback-arc-set problem: given a directed graph, we want to remove the smallest set of edges whose removal makes the remaining graph acyclic. This is an NP-hard problem. Therefore, we approximate its solution by a greedy algorithm, starting with an empty DAG and adding edges as long as they do not introduce cycles or redundancy. This finally yields a DAG of pattern synsets – the PATTY taxonomy.

3. PATTY DEMONSTRATION

Implementation. PATTY is implemented in Java and makes use of the Stanford NLP tool suite for linguistic processing, Hadoop as the platform for large-scale text and data analysis, and MongoDB for storing all resulting data in a key-value representation.

PATTY has generated relation taxonomies from three different corpora: (i) the New York Times archive (*NYT*) which includes about 1.8 Million newspaper articles from the years 1987 to 2007, (ii) the English edition of Wikipedia (*WKP*) which contains about 3.8 Million articles (version of June 21, 2011), and (iii) the ClueWeb09 crawl from `lemurproject.org/clueweb09.php/` (*Web*) which comprises about 500 Million English Web pages. The variant based on Wikipedia is the richest and cleanest one. It consists of about 350,000 typed-pattern synsets organized in a hierarchy with 8,162 subsumptions. Sampling-based assessment indicates that about 85%

[1]http://freebase.com

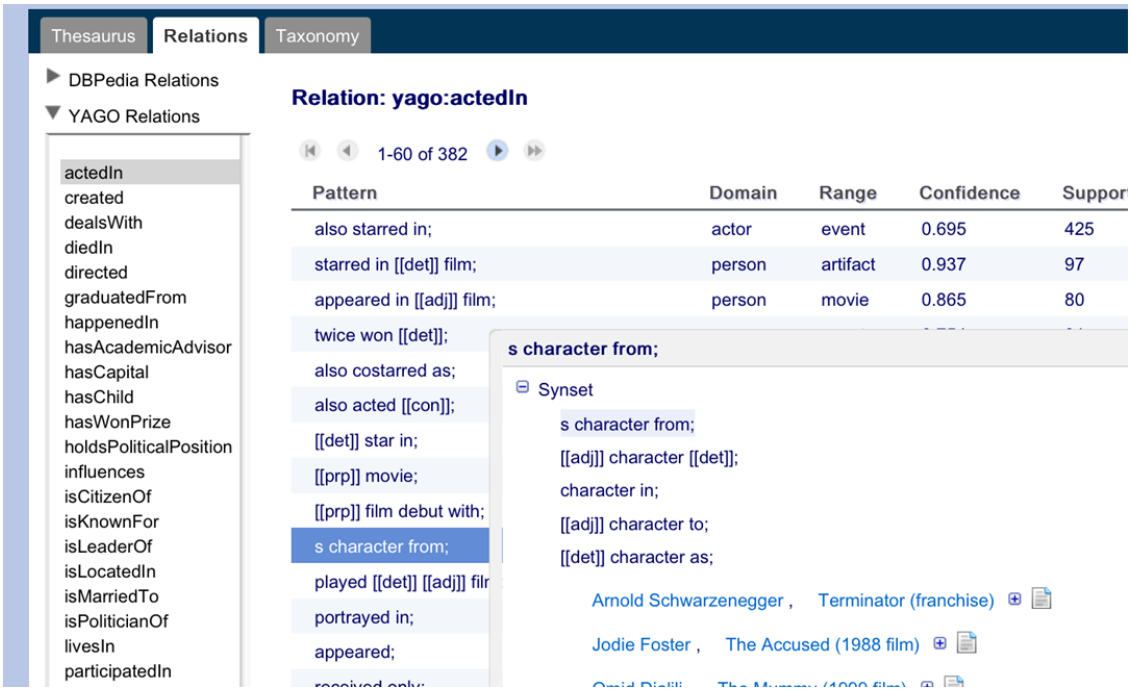

Figure 1: PATTY paraphrases for the YAGO relation actedIn

of the patterns are correct in the sense that they denote meaningful relations with a proper type signature. Futhermore, the subsumptions have a sampling-based accuracy of 83% and 75% for top-ranked and random subsumptions respectively. The Web-based frontend is running AJAX for asynchronous communication with the server.

Using PATTY as a Thesaurus. PATTY connects the world of textual surface patterns to the world of predefined RDF relationships. Users who are aware of RDF-based knowledge bases can explore how RDF relations map to their textual representations. In the demo, we provide paraphrases for DBpedia relations as well as YAGO relations. For example, as shown in Figure 1, PATTY knows over 300 ways in which the YAGO relation *actedIn* can be expressed textually. We hope that this wealth of data can inspire new applications in information extraction, QA, and text understanding.

But users do not need to be familiar with RDF in order to use PATTY. PATTY contains a large number of relations and their different surface forms, and this space can also be visually explored. For example, users can find different ways to express the *hasAcademicAdvisor* relation, simply by typing "worked under" into the search box. PATTY also provides the text snippets where the mention was found as a proof of provenance . These text snippets can be explored to understand the context in which a pattern can have a certain meaning. In addition, users can browse the different meanings of patterns, as they occur with different types of entities.

PATTY provides not only relations and patterns, but also a subsumption hierarchy of patterns, where more general patterns subsume more specific pattterns. The PATTY subsumptions can be explored by clicking nodes that are linked to the root of a pattern. When a node is clicked, the server retrieves all patterns that imply the activated pattern. Figure 3 is a screenshot of a small part of the subsumptions.

Schema-Agnostic Search. Internally, PATTY stores all extracted patterns with their support sets. This allows users to search for facts in the database. For this purpose, our demo provides a search interface where the user can enter Subject-Predicate-Object triples. Different from existing systems, the user does not have to know the schema of the database (i.e., the relations of the fact triples). It is fully sufficient to enter natural language keywords. For example, to find the costars of Brad Pitt, the user can type "costarred with" in place of the relation. PATTY will then search not only for the exact words "costarred with" but also automatically use the paraphrases "appeared with", "cast opposite", and "starred alongside", see Figure 2 . This way the query only needs to be issued once and the user does not need to do multiple paraphrases as is the case for keyword search engines. For each result, PATTY can show the textual sources from which it was derived.

The type signatures of the patterns can be used to narrow down the search results according to different semantic types. For example, when searching for a popular subject like Barack Obama or Albert Einstein, the result may span multiple pages. If the user is interested in only one particular aspect of the entity, then the domain of the subject can be semantically restricted. For example, to see what PATTY knows about Albert Einstein in his role as a scientist,the user can restrict the domain of the relation to *scientist*. Such a query returns Einstein's teaching positions, his co-authors, information about his theories, etc.; but it does not return information about his wives or political activities.

These schema-agnostic queries can be combined to simple join queries. This works by filling out multiple triples and linking them with variables, similar to the way SPARQL operates. Different from SPARQL, our system does not require the user to know the relation name or the entity names. For example, to find visionaries affiliated with MIT, it is sufficient to type *?x vision ?y, ?x ?z MIT*. This will search for people $?x$ who have a vision $?y$ and who stand in some relationship $?z$ with an entity with name *MIT*.

Explaining Relatedness. PATTY can also be used to discover relationships between entities [4]. For example, if the user wishes to know how Tom Cruise and Nicole Kidman are related, it is sufficient to type "Nicole Kidman" into the subject box and "Tom Cruise" into the object box. PATTY will then retrieve all semantic relationships

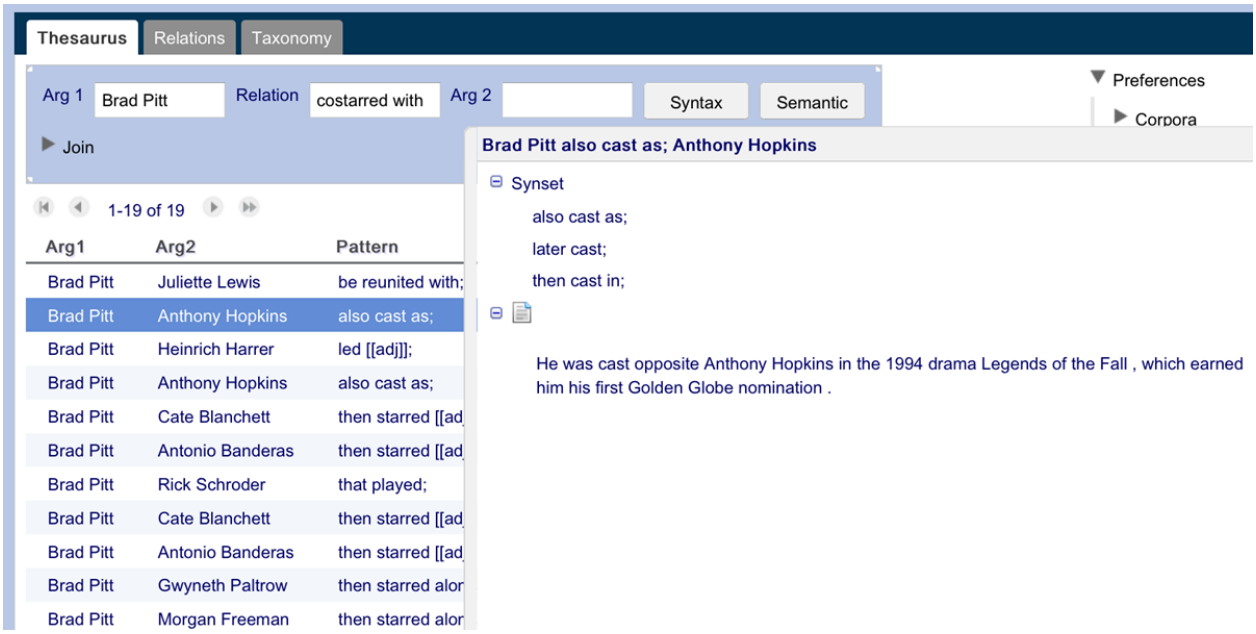

Figure 2: Searching for costars of Brad Pitt

between the two, together with the patterns in which this relationship is expressed. For each result, users can click on the source button discover provenance.

This principle can be extended to full conjunctive queries. For example, to find the entity that links Natalie Portman and Mila Kunis, the user can type *Natalie Portman ?r ?x, Mila Kunis ?s ?x*. This will find all entities $?x$ that link the two actresses, as well as an explanation of how this entity establishes the link. In the example, PATTY finds a ballet movie for $?x$, and says that both actresses appeared in this movie. As this example shows, PATTY has created an internal, semantic representation of the input text documents, which allow her to answer semi-structured queries. In addition to generating semantic patterns, in a sense, PATTY has summarized the input text documents. Users can exploit and query these summaries.

Figure 3: A part of the PATTY taxonomy

4. CONCLUSIONS

PATTY is a collection of relations learned automatically from text. It aims to be to patterns what WordNet is to words. The semantic types of PATTY relations enable advanced search over subject-predicate-object data. With the ongoing trends of enriching Web data (both text and tables) with entity-relationship-oriented semantic annotations, we believe a demo of the PATTY system will be of interest to the database community.

5. REFERENCES

[1] A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka, T.M. Mitchell: Coupled semi-supervised learning for information extraction, WSDM 2010

[2] S. Ceri, M. Brambilla (Eds.): Search Computing: Challenges and Directions, Springer, 2009

[3] A. Fader, S. Soderland, O. Etzioni: Identifying Relations for Open Information Extraction, EMNLP 2011

[4] L. Fang, A. Das Sarma, C. Yu, P. Bohannon: REX: Explaining Relationships between Entity Pairs. PVLDB 5(3), 2011

[5] T. Mohamed, E.R. Hruschka, T.M. Mitchell: Discovering Relations between Noun Categories, EMNLP 2011

[6] N. Nakashole, M. Theobald, G. Weikum: Scalable knowledge harvesting with high precision and high recall, WSDM 2011

[7] N. Nakashole, G. Weikum, F. Suchanek: PATTY: A Taxonomy of Relational Patterns with Semantic Types, EMNLP12

[8] F.M. Suchanek, G. Kasneci, G. Weikum: Yago: a core of semantic knowledge, WWW 2007

[9] W. Wu, H. Li, H. Wang, K. Zhu: Probase: A Probabilistic Taxonomy for Text Understanding, SIGMOD 2012

[10] L. Yao, A. Haghighi, S. Riedel, A. McCallum: Structured Relation Discovery using Generative Models. EMNLP 2011