PATTY: A Taxonomy of Relational Patterns with Semantic Types

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

Abstract

This paper presents PATTY: a large resource for textual patterns that denote binary relations between entities. The patterns are semantically typed and organized into a subsumption taxonomy. The PATTY system is based on efficient algorithms for frequent itemset mining and can process Web-scale corpora. It harnesses the rich type system and entity population of large knowledge bases. The PATTY taxonomy comprises 350,569 pattern synsets. Random-sampling-based evaluation shows a pattern accuracy of 84.7%. PATTY has 8,162 subsumptions, with a random-sampling-based precision of 75%. The PATTY resource is freely available for interactive access and download.

1 Introduction

Motivation. WordNet (Fellbaum 1998) is one of the most widely used lexical resources in computer science. It groups nouns, verbs, and adjectives into sets of synonyms, and arranges these synonyms in a taxonomy of hypernyms. WordNet is limited to single words. It does not contain entire *phrases* or *patterns*. For example, WordNet does not contain the pattern *X is romantically involved with Y.* Just like words, patterns can be synonymous, and they can subsume each other. The pattern *X is romantically involved with Y* is synonymous with the pattern *X is dating Y.* Both are subsumed by *X knows Y.* Patterns for relations are a vital ingredient for many applications, including information extraction and question answering. If a large-scale resource of relational patterns were available, this could boost progress in NLP and AI tasks.

Yet, existing large-scale knowledge bases are mostly limited to abstract binary relationships between entities, such as "bornIn" (Auer 2007; Bollacker 2008; Nastase 2010; Suchanek 2007). These do not correspond to real text phrases. Only the ReVerb system (Fader 2011) yields a larger number of relational textual patterns. However, no attempt is made to organize these patterns into synonymous patterns, let alone into a taxonomy. Thus, the patterns themselves do not exhibit semantics.

Goal. Our goal in this paper is to systematically compile relational patterns from a corpus, and to impose a semantically typed structure on them. The result we aim at is a WordNet-style taxonomy of binary relations. In particular, we aim at patterns that contain *semantic types*, such as ⟨*singer*⟩ *sings* ⟨*song*⟩. We also want to automatically generalize syntactic variations such as *sings her* ⟨*song*⟩ and *sings his* ⟨*song*⟩, into a more general pattern *sings [prp]* ⟨*song*⟩ with POS tag [prp]. Analogously but more demandingly, we want to automatically infer that the above patterns are semantically subsumed by the pattern ⟨*musician*⟩ *performs on* ⟨*musical composition*⟩ with more general types for the entity arguments in the pattern.

Compiling and organizing such patterns is challenging for the following reasons. 1) The number of possible patterns increases exponentially with the length of the patterns. For example, the string "Amy sings 'Rehab'" can give rise to the patterns ⟨*singer*⟩ *sings* ⟨*song*⟩, ⟨*person*⟩ *sings* ⟨*artifact*⟩, ⟨*person*⟩ *[vbz]* ⟨*entity*⟩, etc. If wildcards for multiple words are allowed (such as in ⟨*person*⟩ *sings* * ⟨*song*⟩), the number of possible patterns explodes. 2) A pattern can be semantically more general than another pattern (when one relation is implied by the other relation), and it can also be syntactically more general than another pattern (by the use of placeholders such as [vbz]). These two subsumption orders have a non-obvious interplay, and none can be analyzed without the other. 3) We have to handle pattern sparseness and coincidental matches. If the corpus is small, e.g., the patterns ⟨*singer*⟩ *later disliked her song* ⟨*song*⟩ and ⟨*singer*⟩ *sang* ⟨*song*⟩, may apply to the same set of entity pairs in the corpus. Still, the patterns are not synonymous. 4) Computing mutual subsumptions on a large set of patterns may be prohibitively slow. Moreover, due to noise and vague semantics, patterns may even not form a crisp taxonomy, but require a hierarchy in which subsumption relations have to be weighted by statistical confidence measures.

Contributions. In this paper, we present PATTY, a large resource of relational patterns that are arranged in a semantically meaningful taxonomy, along with entity-pair instances. More precisely, our contributions are as follows:

1) *SOL patterns:* We define an expressive family of relational patterns, which combines syntactic features (S), ontological type signatures (O), and lexical features (L). The crucial novelty is the addition of the ontological, semantic dimension to patterns. When compared to a state-of-the-art pattern language, we found that SOL patterns yield higher recall while achieving similar precision.

2) *Mining algorithms:* We present efficient and scalable algorithms that can infer SOL patterns and subsumptions at scale, based on instance-level overlaps and an ontological type hierarchy.

3) A large *Lexical resource:*. On the Wikipedia corpus, we obtained 350,569 pattern synsets with 84.7% precision. We make our pattern taxonomy available for further research at `www.mpi-inf.mpg.de/yago-naga/patty/`.

The paper is structured as follows. Section 2 discusses related work. Section 3 outlines the basic machinery for pattern extraction. Section 4 introduces our SOL pattern model. Sections 5 and 6 present the syntactic and semantic generalization of patterns. Section 7 explains how to arrange the patterns into a taxonomy. Section 8 reports our experimental findings.

2 Related Work

A wealth of taxonomic knowledge bases (KBs) about entities and their semantic classes have become available. These are very rich in terms of unary predicates (semantic classes) and their entity instances. However, the number of *binary relations* (i.e., relation types, not instances) in these KBs is usually small: Freebase (Bollacker 2008) has a few thousand hand-crafted relations. WikiNet (Nastase 2010) has automatically extracted ca. 500 relations from Wikipedia category names. DBpedia (Auer 2007) has automatically compiled ca. 8000 names of properties from Wikipedia infoboxes, but these include many involuntary semantic duplicates such as *surname* and *lastname*. In all of these projects, the resource contains the *relation names*, but not the *natural language patterns* for them. The same is true for other projects along these lines (Navigli 2010; Philpot 2008; Ponzetto 2007; Suchanek 2007).

In contrast, knowledge base projects that automatically populate relations from Web pages also learn *surface patterns* for the relations: examples are TextRunner/ReVerb (Banko 2007; Fader 2011), NELL (Carlson 2010; Mohamed11), Probase (Wu 2011), the dynamic lexicon approach by (Hoffmann 2010; Wu 2008), the LDA-style clustering approach by (Yao 2011), and projects on Web tables (Limaye 2010; Venetis 2011). Of these, only TextRunner/ReVerb and NELL have made large pattern collections publicly available.

ReVerb (Fader 2011) constrains patterns to verbs or verb phrases that end with prepositions, while PATTY can learn arbitrary patterns. More importantly, all methods in the TextRunner/ReVerb family are blind to the ontological dimension of the entities in the patterns. Therefore, there is no notion of semantic typing for relation phrases as in PATTY.

NELL (Carlson 2010) is based on a fixed set of prespecified relations with type signatures, (e.g., *personHasCitizenship:* ⟨*person*⟩ $\times$ ⟨*country*⟩), and learns to extract suitable noun-phrase pairs from a large Web corpus. In contrast, PATTY discovers patterns for relations that are a priori unknown.

In *OntExt* (Mohamed11), the NELL architecture was extended to automatically compute new relation types (beyond the prespecified ones) for a given type signature of arguments, based on a clustering technique. For example, the relation *musicianPlaysInstrument* is found by clustering pattern co-occurrences for the noun-phrase pairs that fall into the specific type signature $\langle$*musician*$\rangle$ $\times$ $\langle$*musicinstrument*$\rangle$. This technique works for one type signature at a time, and does not scale up to mining a large corpus. Also, the technique is not suitable for inferring semantic subsumptions. In contrast, PATTY efficiently acquires patterns from large-scale corpora and organizes them into a subsumption hierarchy.

Class-based *attribute discovery* is a special case of mining relational patterns (e.g., (Alfonseca 2010; Pasca 2007; Pasca 2008; Reisinger 2009)). Given a semantic class, such as *movies* or *musicians*, the task is to determine relevant attributes, such as *cast* and *budget* for movies, or *albums* and *biography* for musicians, along with their instances. Unlike PATTY's patterns, the attributes are not typed. They come with a prespecified type for the domain, but without any type for the range of the underlying relation.

There are further *relation-centric tasks in NLP and text mining* that have commonalities with our endeavor, but differ in fundamental ways. The SemEval-2010 task on classification of semantic relations between noun-phrase pairs (Hendrickx 2010) aimed at predicting the relation for a given sentence and pair of nominals, but used a fixed set of prespecified relations. Another task in this research avenue is to characterize and predict the argument types for a given relation or pattern (Kozareva 2010; Nakov 2008). This is closer to KB population and less related to our task of discovering relational patterns and systematically organizing them.

From a *linguistic perspective*, there is ample work on patterns for unary predicates of the form *class(entity)*. This includes work on entailment of classes, i.e., on *is-a* and *subclassOf* relationships. Entailment among binary predicates of the form *relation(entity1, entity2)* has received less attention (Lin 2001; Chklovski 2004; Hashimoto 2009; Berant 2011). These works focus solely on verbs, while PATTY learns arbitrary phrases for patterns.

Several lexical resources capture verb categories and entailment: WordNet 3.0 (Fellbaum 1998) contains about 13,000 verb senses, with troponymy and entailment relations; VerbNet (Kipper 2008) is a hierarchical lexicon with more than 5,000 verb senses in ca. 300 classes, including selectional preferences. Again, all of these resources focus solely on verbs.

ConceptNet 5.0 (Havasi 2007) is a thesaurus of commonsense knowledge built as a crowdsourcing endeavor. PATTY, in contrast, is constructed fully automatically from large corpora. Automatic learning of paraphrases and textual entailment has received much attention (see the survey of (Androutsopoulos 2010)), but does not consider fine-grained typing for binary relations, as PATTY does.

3 Pattern Extraction

This section explains how we obtain basic textual patterns from the input corpus. We first apply the Stanford Parser (Marneffe 2006) to the individual sentences of the corpus to obtain dependency paths. The dependency paths form a directed graph, with words being nodes and dependencies being edges. For example, the sentence *"Winehouse effortlessly performed her song Rehab."* yields the following dependency paths:

nsubj(performed-3, Winehouse-1)
advmod(performed-3, effortlessly-2)
poss(Rehab-6, her-4)
nn(Rehab-6, song-5)
dobj(performed-3, Rehab-6)

While our method also works with patterns obtained from shallow features such as POS tags, we found that dependency paths improve pattern extraction precision especially on long sentences.

We then detect mentions of named entities in the parsed corpus. For this purpose, we use a dictionary of entities. This can be any resource that contains named entities with their surface names and semantic types (Auer 2007; Suchanek 2007; Hoffart 2011; Bollacker 2008). In our experiments, we used the YAGO2 knowledge base (Hoffart 2011). We match noun phrases that contain at least one proper noun against the dictionary. For disamiguation, we use a simple context-similarity prior, as described in (Suchanek 2009). We empirically found that this technique has accuracy well above 80% (and higher for prominent and thus frequently occurring entities). In our example, the entity detection yields the entities *Amy Winehouse* and *Rehab (song)*.

Whenever two named entities appear in the same sentence, we extract a textual pattern. For this purpose, we traverse the dependency graph to get the shortest path that connects the two entities. In the example, the shortest path between "Winehouse" and "Rehab" is: Winehouse *nsubj* performed *dobj* Rehab. In order to capture only relations that refer to subject-relation-object triples, we only consider shortest paths that start with subject-like dependencies, such as *nsubj, rcmod and partmod*. To reflect the full meaning of the patterns, we expand the shortest path with adverbial and adjectival modifiers, for example the *advmod* dependency. The sequence of words on the expanded shortest path becomes our final textual pattern. In the example, the textual pattern is *Amy Winehouse effortlessly performed Rehab (song)*.

4 SOL Pattern Model

Textual patterns are tied to the particular surface form of the text. Therefore, we transform the textual patterns into a new type of patterns, called syntactic-ontologic-lexical patterns (SOL patterns). SOL patterns extend lexico-syntactic patterns by ontological type signatures for entities. The SOL pattern language is expressive enough to capture fine-grained relational patterns, yet simple enough to be dealt with by efficient mining algorithms at Web scale.

A SOL pattern is an abstraction of a textual pattern that connects two entities of interest. It is a sequence of words, POS-tags, wildcards, and ontological types. A POS-tag stands for a word of the part-of-speech class. We introduce the special POS-tag [word], which stands for any word of any POS class. A wildcard, denoted $*$, stands for any (possibly empty) sequence of words. Wildcards are essential to avoid overfitting of patterns to the corpus. An ontological type is a semantic class name (such as ⟨*singer*⟩) that stands for an instance of that class. Every pattern contains at least two types, and these are designated as *entity placeholders*.

A string and a pattern *match*, if there is an order-preserving bijection from sequences of words in the string to items in the pattern, so that each item can stand for the respective sequence of words. For example, the pattern ⟨*person*⟩*'s [adj] voice* * ⟨*song*⟩ matches the strings "Amy Winehouse's soft voice in 'Rehab"' and "Elvis Presley's solid voice in his song 'All shook up"'. The *type signature* of a pattern is the pair of the entity placeholders. In the example, the type signature is $person \times song$. The *support set* of a pattern is the set of pairs of entities that appear in the place of the entity placeholders in all strings in the corpus that match the pattern. In the example, the support set of the pattern could be $\{(Amy, Rehab), (Elvis, AllShookUp)\}$. Each pair is called a *support pair* of the pattern.

Pattern B is *syntactically more general* than pattern A if every string that matches A also matches B. Pattern B is *semantically more general* than A if the support set of B is a superset of the support set of A. If A is semantically more general than B and B is semantically more general than A, the patterns are called *synonymous*. A set of synonymous patterns is called a *pattern synset*. Two patterns, of which neither is semantically more general than the other, are called *semantically different*.

To generate SOL patterns from the textual patterns, we decompose the textual patterns into n-grams (n consecutive words). A SOL pattern contains only the n-grams that appear frequently in the corpus and the remaining word sequences are replaced by wildcards. For example, in the sentence "was the first female to run for the governor of" might give rise to the pattern * *the first female* * *governor of*, if "the first female" and "governor of" are frequent in the corpus.

To find the frequent n-grams efficiently, we apply the technique of frequent itemset mining (Agrawal 1993; Srikant 1996): each sentence is viewed as a "shopping transaction" with a "purchase" of several n-grams, and the mining algorithm computes the n-gram combinations with large co-occurrence support[1]. These n-grams allow us to break down a sen-

[1]Our implementation restricts n-grams to length 3 and uses up to 4 n-grams per sentence

tence into wildcard-separated subsequences, which yields an SOL pattern. We generate multiple patterns with different types, one for each combination of types that the detected entities have in the underlying ontology.

We quantify the *statistical strength* of a pattern by means of its support set. For a given pattern p with type signature *t1* $\times$ *t2*, the *support* of p is the size of its support set. For confidence, we compare the support-set sizes of p and an untyped variant p^u of p, in which the types $\langle$*t1*$\rangle$ and $\langle$*t2*$\rangle$ are replaced by the generic type $\langle$*entity*$\rangle$. We define the *confidence* of p as the ratio of the support-set sizes of p and p^u.

5 Syntactic Pattern Generalization

Almost every pattern can be generalized into a syntactically more general pattern in several ways: by replacing words by POS-tags, by introducing wildcards (combining more n-grams), or by generalizing the types in the pattern. It is not obvious which generalizations will be reasonable and useful. We observe, however, that generalizing a pattern may create a pattern that subsumes two semantically different patterns. For example, the generalization $\langle$*person*$\rangle$ *[vb]* $\langle$*person*$\rangle$ subsumes the two semantically different patterns $\langle$*person*$\rangle$ *loves* $\langle$*person*$\rangle$ and $\langle$*person*$\rangle$ *hates* $\langle$*person*$\rangle$. This means that the pattern is semantically meaningless.

Therefore, we proceed as follows. For every pattern, we generate all possible generalizations. If a generalization subsumes multiple patterns with disjoint support sets, we abandon the generalized pattern. Otherwise, we add it to our set of patterns.

6 Semantic Pattern Generalization

The main difficulty in generating semantic subsumptions is that the support sets may contain spurious pairs or be incomplete, thus destroying crisp set inclusions. To overcome this problem, we designed a notion of a *soft set inclusion*, in which one set S can be a subset of another set B to a certain degree. One possible measure for this degree is the confidence, i.e., the ratio of elements in S that are in B, $deg(S \subseteq B) = |S \cap B|/|S|$. However, if a support set S has only few elements due to sparsity, it may become a subset of another support set B, even if the two patterns are semantically different. Therefore, one has to take into account also the *support*, i.e., the size of the set S. Traditionally, this is done through a weighted trade-off between confidence and support.

To avoid the weight tuning, we instead devised a probabilistic model. We interpret S as a *random sample* from the "true" support set S' that the pattern would have on an infinitely large corpus. We want to estimate the ratio of elements of S' that are in B. This ratio is a Bernoulli parameter that can be estimated from the ratio of elements of the sample S that are in B. We compute the Wilson score interval $[c - d, c + d]$ (Brown 2001) for the sample. This interval guarantees that with a given probability (set a priori, usually to $\alpha = 95\%$), the true ratio falls into the interval $[c - d, c + d]$. If the sample is small, d is large and c is close to 0.5. If the sample is large, d decreases and c approaches the naive estimation $|S \cap B|/|S|$. Thereby, the Wilson interval center naturally balances the trade-off between confidence and the support. Hence we define $deg(S \subset B) = c$. This estimator may degrade when the sample size is too small We can alternatively use a conservative estimator $deg(S \subset B) = c - d$, i.e., the lower bound of the Wilson score interval. This gives a low score to the case where $S \subset B$ if we have few samples (S is small).

7 Taxonomy Construction

We now have to arrange the patterns in a semantic taxonomy. A baseline solution would compare every pattern support set to every other pattern support set in order to determine inclusion, mutual inclusion, or independence. This would be prohibitively slow. For this reason, we make use of a prefix-tree for frequent patterns (Han 2005). The prefix-tree stores support sets of patterns. We then developed an algorithm for obtaining set intersections from the prefix-tree.

7.1 Prefix-Tree Construction

Suppose we have pattern synsets and their support sets as shown in Table 1. An entity pair in a support set is denoted by a letter. For example, in the support set for the pattern $\langle Politican \rangle$ *was governor of* $\langle State \rangle$, the entry $\langle$*A,80*$\rangle$ may denote the entity

ID	Pattern Synset & Support Sets
P_1	$\langle Politician \rangle$ was governor of $\langle State \rangle$ A,80 B,75 C,70
P_2	$\langle Politician \rangle$ *politician from* $\langle State \rangle$ A,80 B,75 C,70 D,66 E,64
P_3	$\langle Person \rangle$ *daughter of* $\langle Person \rangle$ F,78 G,75 H,66
P_4	$\langle Person \rangle$ *child of* $\langle Person \rangle$ I,88 J,87 F,78 G,75 K,64

Table 1: Pattern Synsets and their Support Sets

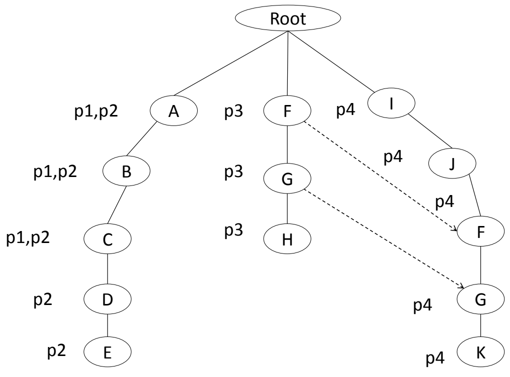

Figure 1: Prefix-Tree for the Synsets in Table 1.

pair *Arnold Schwarzenegger, California*, with an occurrence frequency *80*. The contents of the support sets are used to construct a prefix-tree, where nodes are entity pairs. If synsets have entity pairs in common, they share a common prefix; thus the shared parts can be represented by one prefix-path in the tree. This enables subsumptions to be directly "read off" from the tree, while representing the tree in a compact manner. To increase the chance of shared prefixes, entity pairs are inserted into the tree in decreasing order of occurrence frequency.

The prefix-tree of support sets is a prefix-tree augmented with synset information stored at the nodes. Each node (entity pair) stores the identifiers of the pattern sysnets whose support sets contain that entity pair. In addition, each node stores a link to the next node with the same entity pair.

Figure 1 shows the tree for the pattern synsets in Table 1. The left-most path contains synsets P_1 and P_2. The two patterns have a prefix in common, thus they share the same path. This is reflected by the synsets stored in the nodes in the path. Synsets P_2 and P_3 belong to two different paths due to dissimilar prefixes although they have common nodes. Instead, their common nodes are connected by the same-entity-pair links shown as dotted lines in Figure 1. These links are created whenever the entity pair already exists in the tree but with a prefix different from the prefix of the synset being added to the tree. The size of the tree is at most the total number of entity pairs making up the supports sets of the synsets. The height of the tree is at most the size of the the largest support set.

7.2 Mining Subsumptions from the Prefix-Tree

To efficiently mine subsumptions from the prefix-tree, we have to avoid comparing every path to every other path as this introduces the same inefficiencies that the baseline approach suffers from.

From the construction of the tree it follows that for any node N_i in the tree, all paths containing N_i can be found by following node N_i*'s* links including the same-entity-pair links. By traversing the entire path of a synset P_i, we can reach all the pattern synsets sharing common nodes with P_i. This leads to our main insight: if we start traversing the tree bottom up, starting at the last node in $P_i's$ support set, we can determine exactly which paths are subsumed by P_i. Traversing the tree this way for all patterns gives us the sizes of the support set intersection. The determined intersection sizes can then be used in the Wilson estimator to determine the degree of semantic subsumption and semantic equivalence of patterns.

7.3 DAG Construction

Once we have generated subsumptions between relational patterns, there might be cycles in the graph we generate. We ideally want to remove the minimal total number of subsumptions whose removal results in an a directed acyclic graph (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 a well known NP-hard problem (Kann 1992). We use a greedy algorithm for removing cycles and eliminating redundancy in the subsumptions, thus effectively constructing a DAG. Starting with a list of subsumption edges ordered by decreasing weights, we construct the DAG bottom-up by adding the highest-weight subsumption edge. This step is repeated for all subsumptions, where we add a subsumption to the DAG only if it does not introduce cycles or redundancy. Redundancy occurs when there already exists a path, by transitivity of subsumptions, between pattern synsets linked by the subsumption. This process finally yields a DAG of pattern synsets – the PATTY taxonony.

8 Experimental Evaluation

8.1 Setup

The PATTY extraction and mining algorithms were run on two different input corpora: the New York Times archive (*NYT*) which includes about 1.8 Million newspaper articles from the years 1987 to 2007, and the English edition of Wikipedia (*WKP*), which contains about 3.8 Million articles (as of June 21, 2011). Experiments were carried out, for each corpus, with two different type systems: a) the type system of YAGO2, which consists of about 350,000 semantic classes from WordNet and the Wikipedia category system, and b) the two-level domain/type hierarchy of Freebase which consists of 85 domains and a total of about 2000 types within these domains.

All relational patterns and their respective entity pairs are stored in a MongoDB database. We evaluated PATTY along four dimensions: quality of patterns, quality of subsumptions, coverage, and design alternatives. These dimensions are discussed in the following four subsections. We also performed an extrinsic study to demonstrate the usefulness of PATTY for paraphrasing the relations of DBpedia and YAGO2. In terms of runtimes, he most expensive part is the pattern extraction, where we identify pattern candidates through dependency parsing and perform entity recognition on the entire corpus. This phase runs about a day for Wikipedia a cluster. All other phases of the PATTY system take less than an hour. All experimental data is available on our Web site at `www.mpi-inf.mpg.de/yago-naga/patty/`.

8.2 Precision of Relational Patterns

To assess the precision of the automatically mined patterns (patterns in this section always mean pattern synsets), we sampled the PATTY taxonomy for each combination of input corpus and type system. We ranked the patterns by their statistical strength (Section 4), and evaluated the precision of the *top 100* pattern synsets. Several human judges were shown a sampled pattern synset, its type signature, and a few example instances, and then stated whether the pattern synset indicates a valid relation or not. Evaluators checked the correctness of the type signature, whether the majority of patterns in the synset is reasonable, and whether the instances seem plausible. If so, the synset was flagged as meaningful. The results of this evaluation are shown in column four of Table 2, with a 0.9-confidence Wilson score interval (Brown 2001). In addition, the same assessment procedure was applied to *randomly sampled* synsets, to evaluate the quality in the long tail of patterns. The results are shown in column five of Table 2. For the top 100 patterns, we achieve above 90% precision for Wikipedia, and above 80% for 100 random samples.

Corpus	Types	Patterns	Top 100	Random
NYT	YAGO2	86,982	0.89±0.06	0.72±0.09
	Freebase	809,091	0.87 ±0.06	0.71±0.09
WKP	YAGO2	350,569	0.95±0.04	**0.85**±0.07
	Freebase	1,631,531	0.93±0.05	0.80±0.08

Table 2: Precision of Relational Patterns

From the results we make two observations. First, Wikipedia patterns have higher precision than those from the New York Times corpus. This is because some the language in the news corpus does not express relational information; especially the news on stock markets produced noisy patterns picked up by PATTY. However, we still manage to have a precision of close to 90% for the top 100 patterns and around 72% for random sample on the NYT corpus. The second observation is that the YAGO2 type system generally led to higher precision than the Freebase type system. This is because YAGO2 has finer grained, ontologically clean types, whereas Freebase has broader categories with a more liberal assignment of entities to categories.

8.3 Precision of Subsumptions

We evaluated the quality of the subsumptions by assessing 100 top-ranked as well as 100 randomly selected subsumptions. As shown in Table 3, a large number of the subsumptions are correct. The Wikipedia-based PATTY taxonomy has a random-sampling-based precision of 75%.

Corpus	Types	# Edges	Top 100	Random
NYT	YAGO2	12,601	0.86±0.07	0.68±0.09
	Freebase	80,296	0.89±0.06	0.41±0.09
WKP	YAGO2	**8,162**	0.83±0.07	**0.75**±0.07
	Freebase	20,339	0.85±0.07	0.62±0.09

Table 3: Quality of Subsumptions

Example subsumptions from Wikipedia are:

- $\langle$*person*$\rangle$ *nominated for* $\langle$*award*$\rangle$ $\sqsupset$ $\langle$*person*$\rangle$ *winner of* $\langle$*award*$\rangle$
- $\langle$*person*$\rangle$ *' s wife* $\langle$*person*$\rangle$ $\sqsupset$ $\langle$*person*$\rangle$ *'s widow* $\langle$*person*$\rangle$

8.4 Coverage

To evaluate the coverage of PATTY, we would need a complete ground-truth resource that contains all possible binary relations between entities. Unfortunately, there is no such resource[2]. We tried to approximate such a resource by manually compiling all binary relations between entities that appear in Wikipedia articles of a certain domain. We chose the domain of popular music, because it offers a plethora of non-trivial relations (such as *addictedTo(person,drug)*, *coveredBy(musician,musician)*, *dedicatedSongTo(musician,entity)*)). We considered the Wikipedia articles of five musicians (Amy Winehouse, Bob Dylan, Neil Young, John Coltrane, Nina Simone). For each page, two annotators hand-extracted all relationship types that they would spot in the respective articles. The annotators limited themselves to relations where at least one argument type is $\langle$*musician*$\rangle$. Then we formed the intersection of the two annotators' outputs (i.e., their agreement) as a reasonable gold standard for relations identifiable by skilled humans. In total, the gold-standard set contains 163 relations.

We then compared our relational patterns to the relations included in four major knowledge bases, namely, YAGO2, DBpedia (DBP), Freebase (FB), and NELL, limited to the specific domain of music. Table 4 shows the absolute number of relations covered by each resource. For PATTY, the patterns were derived from the Wikipedia corpus with the YAGO2 type system.

gold standard	PATTY	YAGO2	DBP	FB	NELL
163	**126**	31	39	69	13

Table 4: Coverage of Music Relations

PATTY covered 126 of the 163 gold-standard relations. This is more than what can be found in large semi-curated knowledge bases such as Freebase, and twice as much as Wikipedia-infobox-based resources such as DBpedia or YAGO offer. Some PATTY examples that do not appear in the other resources at all are:

- $\langle$*musician*$\rangle$ *PRP idol* $\langle$*musician*$\rangle$ for the relation *hasMusicalIdol*
- $\langle$*person*$\rangle$ *criticized by* $\langle$*organization*$\rangle$ for *critizedByMedia*
- $\langle$*person*$\rangle$ *headliner* $\langle$*artifact*$\rangle$ for *headlinerAt*
- $\langle$*person*$\rangle$ *successfully sued* $\langle$*person*$\rangle$ for *suedBy*
- $\langle$*musician*$\rangle$ *wrote hits for* $\langle$*musician*$\rangle$ for *wroteHitsFor*,

This shows (albeit anecdotically) that PATTY's patterns contribute added value beyond today's knowledge bases.

8.5 Pattern Language Alternatives

We also investigated various design alternatives to the PATTY pattern language. We looked at three main alternatives: the first is verb-phrase-centric patterns advocated by ReVerb (Fader 2011), the second is the PATTY language without type signatures (just using sets of n-grams with syntactic generalizations), and the third one is the full PATTY language. The results for the Wikipedia corpus and the

[2]Lexical resources such as WordNet contain only verbs, but not binary relations such as *is the president of.* Other resources are likely incomplete.

	Reverb-style patterns	PATTY without types	PATTY full
# Patterns	5,996	184,629	**350,569**
Patterns Precision	0.96±0.03	0.74±0.08	**0.95**±0.04
# Subsumptions	74	15,347	**8,162**
Subsumptions Precision	0.79 ±0.09	0.58±0.09	**0.83**±0.07
# Facts	192,144	6,384,684	**3,890,075**
Facts Precision.	0.86 ±0.07	0.64±0.09	**0.88** ±0.06

Table 5: Results for Different Pattern Language Alternatives

Relation	Paraphrases	Precision	Sample Paraphrases
DBPedia/artist	83	0.96±0.03	[adj] studio album of, [det] song by …
DBPedia/associatedBand	386	0.74±0.11	joined band along, plays in …
DBPedia/doctoralAdvisor	36	0.558±0.15	[det] student of, under * supervision …
DBPedia/recordLabel	113	0.86±0.09	[adj] artist signed to, [adj] record label …
DBPedia/riverMouth	31	0.83±0.12	drains into, [adj] tributary of …
DBPedia/team	1,108	0.91±0.07	be * traded to, [prp] debut for …
YAGO/actedIn	330	0.88±0.08	starred in * film, [adj] role for …
YAGO/created	466	0.79±0.10	founded, 's book …
YAGO/isLeaderOf	40	0.53±0.14	elected by, governor of …
YAGO/holdsPoliticalPosition	72	0.73±0.10	[prp] tenure as, oath as …

Table 6: Sample Results for Relation Paraphrasing

YAGO2 type system are shown in Table 5; precision figures are based on the respective top 100 patterns or subsumption edges. We observe from these results that the type signatures are crucial for precision. Moreover, the number of patterns, subsumptions and facts found by verb-phrase-centric patterns (ReVerb (Fader 2011)), are limited in recall. General pattern synsets with type signatures, as newly pursued in this paper, substantially outperform the verb-phrase-centric alternative in terms of pattern and subsumption recall while yielding high precision.

8.6 Extrinsic Study: Relation Paraphrasing

To further evaluate the usefulness of PATTY, we performed a study on relation paraphrasing: given a relation from a knowledge base, identify patterns that can be used to express that relation. Paraphrasing relations with high-quality patterns is important for populating knowledge bases and counters the problem of semantic drifting caused by ambiguous and noisy patterns.

We considered relations from two knowledge bases, DBpedia and YAGO2, focusing on relations that hold between entities and do not include literals. PATTY paraphrased 225 DBpedia relations with a total of 127,811 patterns, and 25 YAGO2 relations with a total of 43,124 patterns. Among these we evaluated a random sample of 1,000 relation paraphrases. Table 6 shows precision figures for some selected relations, along anecdotic example patterns.

Some relations are hard to capture precisely. For *DBPedia/doctoralAdvisor*, e.g., PATTY picked up patterns like "worked with" as paraphrases. These are not entirely wrong, but we evaluated them as false because they are too general to indicate the more specific doctoral advisor relation.

Overall, however, the paraphrasing precision is high. Our evaluation showed an average precision of **0.76**±0.03 across all relations.

9 Conclusion and Future Directions

This paper presented PATTY, a large resource of text patterns. Different from existing resources, PATTY organizes patterns into synsets and a taxonomy, similar in spirit to WordNet. Our evaluation shows that PATTY's patterns are semantically meaningful, and that they cover large parts of the relations of other knowledge bases. The Wikipedia-based version of PATTY contains 350,569 pattern synsets at a precision of 84.7%, with 8,162 subsumptions, at a precision of 75%. The PATTY resource is freely available for interactive access and download at www.mpi-inf.mpg.de/yago-naga/patty/.

Our approach harnesses existing knowledge bases for entity-type information. However, PATTY is not tied to a particular choice for this purpose. In fact, it would be straightforward to adjust PATTY to using surface-form noun phrases rather than disambiguated entities, as long as we have means to infer at least coarse-grained types (e.g., person, organization, location). An interesting future direction is to study this generalized setting. We would also like to investigate the enhanced interplay of information extraction and pattern extraction, and possible applications for question answering.

References

Ion Androutsopoulos, Prodromos Malakasiotis: A Survey of Paraphrasing and Textual Entailment Methods. Journal of Artificial Intelligence Research 38: 135–187, 2010

Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference 1993

Enrique Alfonseca, Marius Pasca, Enrique Robledo-Arnuncio: Acquisition of instance attributes via labeled and related instances. SIGIR 2010

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary G. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC 2007, data at http://dbpedia.org

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni: Open Information Extraction from the Web. IJCAI 2007

Jonathan Berant, Ido Dagan, Jacob Goldberger: Global Learning of Typed Entailment Rules. ACL 2011

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor: Freebase: a collaboratively created graph database for structuring human knowledge. SIGMOD Conference 2008, data at http://freebase.com

Lawrence D. Brown, T.Tony Cai, Anirban Dasgupta: Interval Estimation for a Binomial Proportion. Statistical Science 16: 101–133, 2001

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. AAAI 2010, data at http://rtw.ml.cmu.edu/rtw/

Timothy Chklovski, Patrick Pantel: VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. EMNLP 2004; data available at http://demo.patrickpantel.com/demos/verbocean/

Anthony Fader, Stephen Soderland, Oren Etzioni: Identifying Relations for Open Information Extraction. EMNLP 2011, data at http://reverb.cs.washington.edu

Christiane Fellbaum (Editor): WordNet: An Electronic Lexical Database. MIT Press, 1998

Jiawei Han, Jian Pei , Yiwen Yin : Mining frequent patterns without candidate generation. SIGMOD 2000.

Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata, Jun'ichi Kazama: Large-Scale Verb Entailment Acquisition from the Web. EMNLP 2009

Catherine Havasi, Robert Speer, and Jason Alonso. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge, RANLP 2007; data available at http://conceptnet5.media.mit.edu/

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Seaghdha, Sebastian Pado, Marco Pennacchiotti, Lorenza Romano, Stan Szpakowicz: SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals, 5th ACL International Workshop on Semantic Evaluation, 2010; data available at http://www.isi.edu/~kozareva/downloads.html

Johannes Hoffart, Fabian Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, Gerhard Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. WWW 2011, data at http://yago-knowledge.org

Raphael Hoffmann, Congle Zhang, Daniel S. Weld: Learning 5000 Relational Extractors. ACL 2010

Vigo Kann: On the approximability of NP-complete optimization problems. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm. 1992.

Karin Kipper, Anna Korhonen, Neville Ryant, Martha Palmer, A Large-scale Classification of English Verbs, Language Resources and Evaluation Journal, 42(1): 21-40, 2008, data available at http://verbs.colorado.edu/~mpalmer/projects/verbnet/downloads.html

Zornitsa Kozareva, Eduard H. Hovy: Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns. ACL 2010

Girija Limaye, Sunita Sarawagi, Soumen Chakrabarti: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB 3(1): 1338-1347 (2010)

Dekang Lin, Patrick Pantel: DIRT: discovery of inference rules from text. KDD 2001

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. LREC 2006.

Thahir Mohamed, Estevam R. Hruschka Jr., Tom M. Mitchell: Discovering Relations between Noun Categories. EMNLP 2011

Ndapandula Nakashole, Martin Theobald, Gerhard Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. WSDM 2011

Preslav Nakov, Marti A. Hearst: Solving Relational Similarity Problems Using the Web as a Corpus. ACL 2008

Vivi Nastase, Michael Strube, Benjamin Boerschinger, Cäcilia Zirn, Anas Elghafari: WikiNet: A Very Large Scale Multi-Lingual Concept Network. LREC 2010, data at http://www.h-its.org/english/research/nlp/download/wikinet.php

Roberto Navigli, Simone Paolo Ponzetto: BabelNet: Building a Very Large Multilingual Semantic Network. ACL 2010 data at http://lcl.uniroma1.it/babelnet/

Marius Pasca, Benjamin Van Durme: What You Seek Is What You Get: Extraction of Class Attributes from Query Logs. IJCAI 2007

Marius Pasca, Benjamin Van Durme: Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. ACL 2008

Andrew Philpot, Eduard Hovy, Patrick Pantel: The Omega Ontology, in: Ontology and the Lexicon, Cambridge University Press, 2008, data at http://omega.isi.edu/

Simone Paolo Ponzetto, Michael Strube: Deriving a Large-Scale Taxonomy from Wikipedia. AAAI 2007, data at http://www.h-its.org/english/research/nlp/download/wikitaxonomy.php

Joseph Reisinger, Marius Pasca: Latent Variable Models of Concept-Attribute Attachment. ACL/AFNLP 2009

Ramakrishnan Srikant, Rakesh Agrawal: Mining Sequential Patterns: Generalizations and Performance Improvements. EDBT 1996

Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007

Fabian M. Suchanek, Mauro Sozio, Gerhard Weikum: SOFIE: a self-organizing framework for information extraction. WWW 2009

Lin Sun, Anna Korhonen: Hierarchical Verb Clustering Using Graph Factorization. EMNLP 2011

Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, Chung Wu: Recovering Semantics of Tables on the Web. PVLDB 4(9): 528-538, 2011

Tom White: Hadoop: The Definitive Guide, 2nd Edition. O'Reilly, 2010.

Fei Wu, Daniel S. Weld: Automatically refining the wikipedia infobox ontology. WWW 2008

Wentao Wu, Hongsong Li, Haixun Wang, Kenny Q. Zhu: Towards a Probabilistic Taxonomy of Many Concepts. Technical Report MSR-TR-2011-25, Microsoft Research, 2011

Limin Yao, Aria Haghighi, Sebastian Riedel, Andrew McCallum: Structured Relation Discovery using Generative Models. EMNLP 2011

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, Ji-Rong Wen: StatSnowball: a statistical approach to extracting entity relationships. WWW 2009