DOI: 10.1016/j.websem.2016.03.004

Sar-graphs: A Language Resource Connecting Linguistic Knowledge with Semantic Relations from Knowledge Graphs

Sebastian Krause[a,*], Leonhard Hennig[a], Andrea Moro[b], Dirk Weissenborn[a], Feiyu Xu[a], Hans Uszkoreit[a], Roberto Navigli[b]

[a] *DFKI Language Technology Lab, Alt-Moabit 91c, 10559 Berlin, Germany*
[b] *Dipartimento di Informatica, Sapienza Università di Roma, Viale Regina Elena 295, 00161 Roma, Italy*

Abstract

Recent years have seen a significant growth and increased usage of large-scale knowledge resources in both academic research and industry. We can distinguish two main types of knowledge resources: those that store factual information about entities in the form of semantic relations (e.g., Freebase), namely so-called knowledge graphs, and those that represent general linguistic knowledge (e.g., WordNet or UWN). In this article, we present a third type of knowledge resource which completes the picture by connecting the two first types. Instances of this resource are *graphs of semantically-associated relations (sar-graphs)*, whose purpose is to link semantic relations from factual knowledge graphs with their linguistic representations in human language.

We present a general method for constructing sar-graphs using a language- and relation-independent, distantly supervised approach which, apart from generic language processing tools, relies solely on the availability of a lexical semantic resource, providing sense information for words, as well as a knowledge base containing seed relation instances. Using these seeds, our method extracts, validates and merges relation-specific linguistic patterns from text to create sar-graphs. To cope with the noisily labeled data arising in a distantly supervised setting, we propose several automatic pattern confidence estimation strategies, and also show how manual supervision can be used to improve the quality of sar-graph instances. We demonstrate the applicability of our method by constructing sar-graphs for 25 semantic relations, of which we make a subset publicly available at *http://sargraph.dfki.de*.

We believe sar-graphs will prove to be useful linguistic resources for a wide variety of natural language processing tasks, and in particular for information extraction and knowledge base population. We illustrate their usefulness with experiments in relation extraction and in computer assisted language learning.

Keywords: Knowledge graphs, language resources, linguistic patterns, relation extraction

1. Introduction

Knowledge graphs are vast networks which store entities and their semantic types, properties and relations. In recent years considerable effort has been invested into constructing these large knowledge bases in academic research, community-driven projects and industrial development. Prominent examples include Freebase [1], Yago [2, 3], DBpedia [4], NELL [5, 6], WikiData [7], PROSPERA [8], Google's Knowledge Graph [9] and also the Google Knowledge Vault [10]. A parallel and in part independent development is the emergence of several large-scale knowledge resources with a more language-centered focus, such as UWN [11], BabelNet [12], ConceptNet [13], and UBY [14]. These resources are important contributions to the linked data movement, where repositories of world-knowledge and linguistic knowledge complement each other. In this article, we present a method that aims to bridge these two types of resources by automatically building an intermediate resource.

In comparison to (world-)knowledge graphs, the underlying representation and semantic models of linguistic knowledge resources exhibit a greater de-

*Corresponding author
Email address: skrause@dfki.de (Sebastian Krause)

gree of diversity. ConceptNet makes use of natural-language representations for modeling commonsense information. BabelNet integrates entity information from Wikipedia with word senses from WordNet, as well as with many other resources such as Wikidata and Wiktionary [15]. UWN automatically builds a multilingual WordNet from various resources, similar to UBY, which integrates multiple resources via linking on the word-sense level. Few to none of the existing linguistic resources, however, provide a feasible approach to explicitly linking semantic relations from knowledge graphs with their linguistic representations. We aim to fill this gap with the resource whose structure we define in Section 2 and whose construction method we detail in Section 3. Instances of this resource are *graphs of semantically-associated relations*, which we refer to by the name *sar-graphs*. Our definition is a formalization of the idea sketched in [16]. We believe that sar-graphs are examples for a new type of knowledge repository, *language graphs*, as they represent the linguistic patterns for relations in a knowledge graph. A language graph can be thought of as a bridge between the language and knowledge encoded in a knowledge graph, a bridge that characterizes the ways in which a language can express instances of one or several relations, and thus a mapping between strings and things.

The construction strategies of the described (world-)knowledge resources range from 1) integrating existing structured or semi-structured knowledge (e.g., Wikipedia infoboxes) via 2) crowdsourcing to 3) automatic extraction from semi- and unstructured resources, where often 4) combinations of these are implemented. At the same time the existence of knowledge graphs enabled the development of new technologies for knowledge engineering, e.g., distantly supervised machine-learning methods [8, 17, 18, 19, 20]. Relation extraction is one of the central technologies contributing to the automatic creation of fact databases [10], on the other hand it benefits from the growing number of available factual resources by using them for automatic training and improvement of extraction systems. In Section 3, we describe how our own existing methods [18], which exploit factual knowledge bases for the automatic gathering of linguistic constructions, can be employed for the purpose of sar-graphs. Then in turn, one of many potential applications of sar-graphs is relation extraction, which we illustrate in Section 7.

An important aspect of the construction of sar-

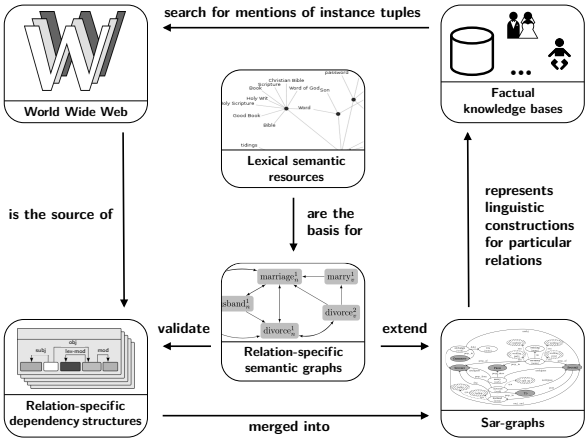


Figure 1: Relation of sar-graphs to other knowledge resources.

graphs is the disambiguation of their content words with respect to lexical semantics knowledge repositories, thereby generalizing content words with word senses. In addition to making sar-graphs more adjustable to the varying granularity needs of possible applications, this positions sar-graphs as a link hub between a number of formerly independent resources (see Figure 1). Sar-graphs represent linguistic constructions for semantic relations from factual knowledge bases and incorporate linguistic structures extracted from mentions of knowledge-graph facts in free texts, while at the same time anchoring this information in lexical semantic resources. We go into further detail on this matter in Section 6.

The distantly supervised nature of the proposed construction methodology requires means for automatic and manual confidence estimation for the extracted linguistic structures, presented in Section 4. This is of particular importance when unstructured web texts are exploited for finding linguistic patterns which express semantic relations. Our contribution is the combination of battle-tested confidence-estimation strategies [18, 21] with a large manual verification effort for linguistic structures. In our experiments (Section 5), we continue from our earlier work [18, 22], i.e., we employ Freebase as our source of semantic relations and the lexical knowledge base BabelNet for linking word senses. We create sar-graphs for 25 relations, which exemplifies the feasibility of the proposed method, also we make the resource publicly available for this core set of relations.

We demonstrate the usefulness of sar-graphs by applying them to the task of relation extraction,

preimage(f)	A_f	**Example for Σ_f**
V from lexical tokens	word form, word lemma, word class, word sense	`married`, to `marry`, verb, bn:00085614v
V from entity mentions	entity type, semantic role	person, SPOUSE2
E from syntactic parsing	dependency labels	nsubjpass
E from resource linking	lexical semantic relation	*synonym*
$V \cup E$	frequency in training set	2
$V \cup E$	identifiers for sentences & dependency structures	[sent:16, sent:21], [pat:16#1, pat:21#2]

Table 1: Names and example values for attributes of sar-graph elements.

where we identify and compose mentions of argument entities and projections of n-ary semantic relations. We believe that sar-graphs will prove to be a valuable resource for numerous other applications, such as adaptation of parsers to special recognition tasks, text summarization, language generation, query analysis and even interpretation of telegraphic style in highly elliptical texts as found in SMS, Twitter, headlines or brief spoken queries. We therefore make this resource freely available to the community, and hope that other parties will find it of interest (Section 8).

2. Sar-graphs: A linguistic knowledge resource

Sar-graphs [16] extend the current range of knowledge graphs, which represent factual, relational and common-sense information for one or more languages, with linguistic knowledge, namely, linguistic variants of how semantic relations between abstract concepts and real-world entities are expressed in natural language text.

2.1. Definition

Sar-graphs are directed multigraphs containing linguistic knowledge at the syntactic and lexical semantic level. A sar-graph is a tuple

$$G_{r,l} = (V, E, s, t, f, A_f, \Sigma_f),$$

where

- V is the set of vertices,
- E is the set of edges,
- $s : E \mapsto V$ maps edges to their start vertex,
- $t : E \mapsto V$ maps edges to their target vertex.

As both vertices and edges are labeled, we also need an appropriate labeling function, denoted by f. f does more than just attaching atomic labels to edges and vertices but rather associates both with sets of features (i.e., attribute-value pairs) to account for the needed complexity of linguistic description:

$$f : V \cup E \mapsto \mathcal{P}(A_f \times \Sigma_f)$$

where

- $\mathcal{P}(\cdot)$ constructs a powerset,
- A_f is the set of attributes (i.e., attribute names) which vertices and edges may have, and
- Σ_f is the value alphabet of the features, i.e., the set of possible attribute values for all attributes.

The information in one instance of such a graph is specific to a given language l and target relation r. In general, r links $n \geq 2$ entities wrt. their semantic relationship in the real world. An example relation is *marriage*, connecting two spouses to one another, and optionally to the location and date of their wedding, as well as to their date of divorce:[1]

$$r_{mar.}(\text{SPOUSE1}, \text{SPOUSE2}, \text{CEREMONY}, \text{FROM}, \text{TO}).$$

The function of sar-graphs is to represent the linguistic constructions a language l provides for reporting instances of r or for just referring to such instances. A vertex $v \in V$ corresponds to a word in such a construction. The features assigned to a vertex via the labeling function f provide information about lexico-syntactic aspects (*word form* and *lemma*, *word class*), lexical semantics (*word sense*) and semantic points (*global entity identifier*, *entity type*, *semantic role in the target relation*). Additionally, they provide statistical and meta information (e.g., *frequency*). Table 1 presents an overview of the possible attributes.

[1] In the remainder of this article, we refer to the *arguments* of semantic relations at times via labels for the arguments (in SMALLCAPS, e.g., SPOUSE1) and at other times via the entity types of possible argument fillers (with sans-serif font, e.g., person), depending on the context.

The linguistic constructions are modeled as subtrees of dependency-graph representations of sentences. In this article, we refer to these trees as *dependency structures* or *dependency constructions*. Each such structure typically describes one particular way to express a semantic relation in a given language. Edges $e \in E$ are consequently labeled with dependency tags via f, in addition to frequency information.

In the literature, linguistic constructions of this kind are often referred to as *extraction patterns*, motivated by the application of such structures for the extraction of relations from sentences. A difference to sar-graphs is that individual dependency structures may or may not be present in a sar-graph as disjunct trees, i.e., we merge constructions or parts thereof. The joint representation of common paths of linguistic expressions allows for a quick identification of dominant phrases and the calculation of frequency distributions for sub-trees and their combinations. This merging step is not destructive, the information about the linguistic structures found in original sentences is still available. We believe that for language expressions, an exhaustive, permanent merging does not make sense, as it would mean losing the language variety which we aim at capturing.

The merging process is implemented with a conservative default strategy, which cautiously connects dependency constructions at their argument positions, followed by a customizable second step, which further superimposes nodes and paths in a non-destructive manner. We describe this two step process in Section 3.4. In the remainder of this section, we want to convey a general intuition of what sar-graphs are, hence a more abstract and uniform view on the merging process is assumed.

We expect that novel constructions emerge in sar-graphs, coming from the combination of two or more known phrases. See for example these two phrases, each connecting two arguments of relation *marriage*:

- `Ann wed in York.`
- `Ann wed on July 27, 2007.`

A joint representation of them in a sar-graph gives us a three-argument dependency structure, corresponding to the following sentence, in which both the location and date argument are attached to the verb, and not just one of them: `Ann wed in York on July 27, 2007.`

If a given language l only provides a single construction to express an instance of r, then the dependency structure of this construction forms the entire sar-graph. But if the language offers alternatives to this construction, i.e., paraphrases, their dependency structures are also added to the sar-graph. The individual constructions superimpose one another based on shared properties and labels of vertices and edges. Specifically, we merge

- *vertices without a semantic role* based on their word lemma or entity type
- *vertices with argument roles* wrt. their semantic role in the target relation
- *edges* on the basis of dependency labels.

Our data-driven approach to the creation of sar-graphs integrates not just constructions that include all relation arguments but also those mentioning only a subset thereof. As long as these constructions indicate an instance of the target relation, they are relevant for many applications, such as high-recall relation extraction, even though they are not true paraphrases of constructions fully expressing the n-ary relation.

A sar-graph for the two English constructions in Example 1, both with mentions of projections of the *marriage* relation may look as presented in Figure 2.

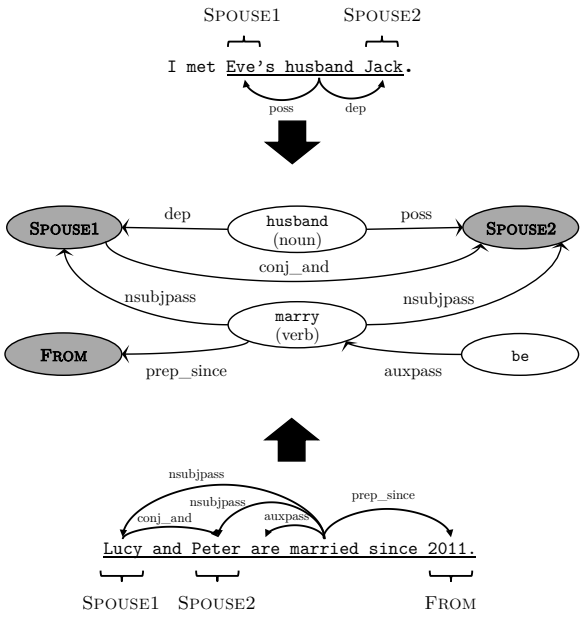


Figure 2: Sar-graph example generated from two English sentences. The sar-graph connects the dependency structures via their shared SPOUSE arguments and additionally includes edges and vertices linking the FROM argument from the second sentence.

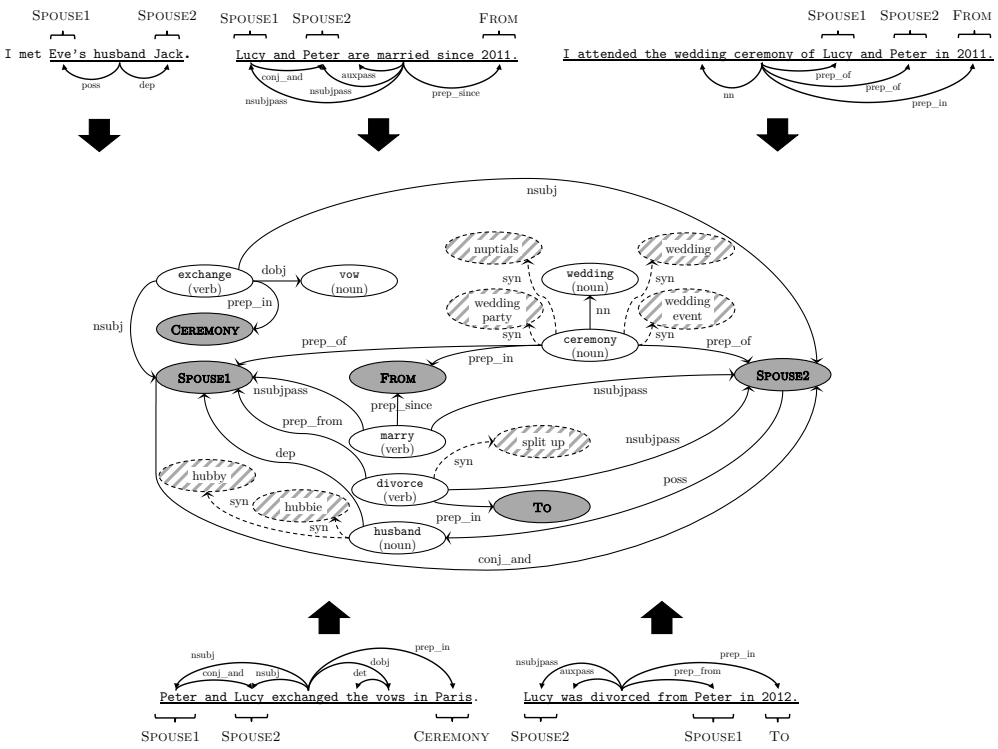


Figure 3: More complex example for a sar-graph. This graph also includes lexical semantic information (dashed vertices and edges) obtained by linking content words to a lexical semantic resource.

Example 1

- `I met Eve's husband Jack.`
- `Lucy and Peter are married since 2011.`

From the dependency parse trees of these sentences, we can extract two graphs that connect the relation's arguments. The first sentence lists the spouses with a possessive construction, the second sentence using a conjunction. In addition, the second sentence provides the marriage date. The graph we extract from the latter sentence hence includes the dependency arcs *nsubjpass* and *prep_since*, as well as the node for the content word *marry*. We connect the two extracted structures by their shared semantic arguments, namely, SPOUSE1 and SPOUSE2. As a result, the graph in Figure 2 contains a path from SPOUSE1 to SPOUSE2 via the node *husband* for sentence (1), and an edge *conj_and* from SPOUSE1 to SPOUSE2 for sentence (2). The dependency relations connecting the FROM argument yield the remainder of the sar-graph. Note that the graph contains two types of vertices: argument nodes labeled with their semantic role, and lexical semantic nodes labeled with their lemma and POS tag.

Figure 3 illustrates the structure and content of a more complex sar-graph example, again for the *marriage* relation. We extend the previous example with three more sentences, which provide alternative linguistic constructions, as well as the additional arguments CEREMONY and TO. The graph now includes the paraphrases *exchange vows*, *wedding ceremony of*, and *was divorced from*. Note that both sentence (2) and (4) utilize a *conj_and* to connect the SPOUSES. The sar-graph includes this information as a single edge, but we can encode the frequency information as an edge attribute. The graph also contains additional lexical semantic information, represented by the dashed vertices and edges (see Section 2.3).

2.2. Less explicit relation mentions

A key property of sar-graphs is that they store linguistic structures with varying degrees of explicitness wrt. to the underlying semantic relations. Constructions that refer to some part or aspect of the relation would normally be seen as sufficient evidence of an instance even if there could be contexts in which this implication is canceled; consider the

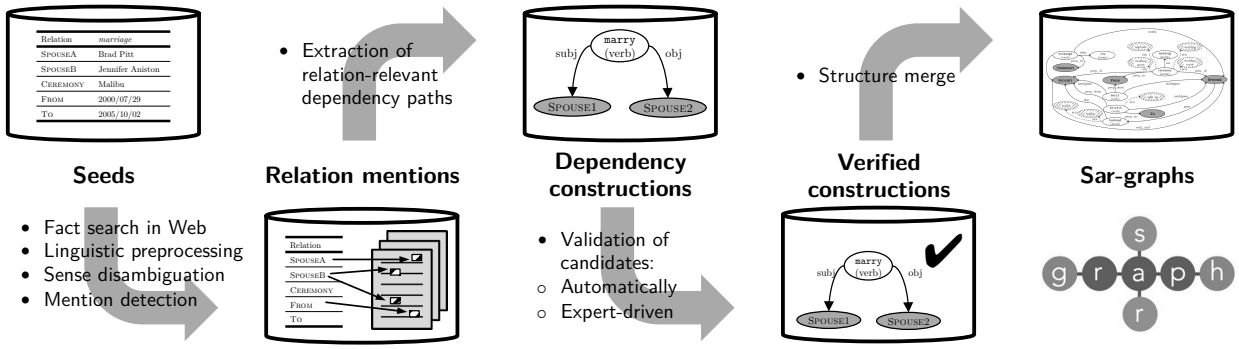


Figure 4: Outline of sar-graph construction. Arrows correspond to processing steps, while boxes show intermediate results.

sentences in Example 2:

Example 2

- `Joan and Edward exchanged rings in 2011.`
- `Joan and Edward exchanged rings during the rehearsal of the ceremony.`

Other constructions refer to relations that entail the target relations without being part of it:

Example 3

- `Joan and Edward celebrated their 12th wedding anniversary.`
- `Joan and Edward got divorced in 2011.`

And finally there are constructions referring to semantically connected relations that by themselves might not be used for safely detecting instances of r, but that could be employed for recall-optimized applications or for a probabilistic detection process that combines several pieces of evidence:

Example 4

- `I met her last October at Joan's bachelorette (engagement) party.`

Some entirely probabilistic entailments are caused by social conventions or behavioral preferences:

Example 5

- `Two years before Joan and Paul had their first child, they bought a larger home.`

2.3. Graphs augmented with lexical semantics

The lexico-syntactic and semantic information specified in sar-graphs is augmented with lexical semantic knowledge by disambiguating all content words in the dependency structures. This results in a labeling of content word vertices with sense identifiers and additional (synonymous) surface forms for the sar-graph vertices and also implicit lexical semantic links among words already contained in the sar-graph. These implicit links bear tags such as hypernym, synonym, troponym, or antonym.

In the sar-graph of Figure 3, additional surface forms are illustrated by dashed vertices and edges. For example, for the vertex representing the lemma `husband`, the colloquial synonyms `hubby` and `hubbie` are listed.

Among the benefits of this injection of lexical-semantic information into sar-graphs is a larger amount of resulting paraphrases. In sar-graph applications like relation extraction, additional paraphrases lead to a higher number of detected relation mentions. Furthermore, the disambiguation information allows us to employ a sophisticated confidence estimation method for the underlying dependency constructions, which we describe in Section 4. With these confidence assessments, we can reliably identify the constructions in a sar-graph which may only entail the target relation of interest, in contrast to those explicitly expressing it.

3. Sar-graph construction

In this section, we describe a general method for constructing sar-graphs. Our method is language- and relation-independent, and relies solely on the availability of a set of seed relation instances from an existing knowledge base. Figure 4 outlines this process. Given a target relation r, a set of seed instances $\mathcal{I}_r$ of this relation, and a language l, we can create a sar-graph $G_{r,l}$ with the following procedure.

1. Acquire a set of textual mentions $\mathcal{M}_{r,l}$ of instances i for all $i \in \mathcal{I}_r$ from a text corpus.
2. Extract candidate dependency constructions $\mathcal{D}_{r,l}$ from the dependency trees of elements of $\mathcal{M}_{r,l}$.
3. Validate the candidate structures $d \in \mathcal{D}_{r,l}$, either automatically or via human expert-driven quality control, yielding a derived set $\mathcal{D}'_{r,l}$ of acceptable dependency constructions.
4. Merge elements of $d \in \mathcal{D}'_{r,l}$ to create the sar-graph $G_{r,l}$.

We discuss each of these steps in more detail in the following sections.

3.1. Textual mention acquisition and preprocessing

The first step in the processing pipeline is to collect a large number of textual mentions of a given target relation, ideally covering many different linguistic constructions used to express the relation. Following [18, 23, 24], we collect textual mentions using as input a set of seed instances $\mathcal{I}_r$ of the target relation r. Every sentence which contains the entity tuples of the seed instances is regarded as a textual mention of the relation. As in standard distantly supervised approaches, this seed instance set can be easily obtained from an existing knowledge base (KB).

The seeds are used as queries for a web search engine to find documents that potentially contain mentions of the seeds. We construct a separate query for each seed by concatenating the full names of all seed argument entities. Documents returned by the search engine are downloaded and converted into plain text, using standard methods for HTML-to-text conversion and boilerplate detection.

We then perform standard NLP preprocessing of the text documents, including sentence detection, tokenization, named-entity recognition (NER), lemmatization, part-of-speech tagging, using off-the-shelf tools. To enable a better understanding and exploitation of the extracted dependency structures, we link their relevant elements (i.e., the content words) to a lexical-semantic resource. We also link entity mentions to seed entities with a simple dictionary-based linking strategy that matches name variations of the seed's entities as provided by the KB.

We discard all sentences not mentioning a seed instance, as well as sentences not expressing all essential arguments of the relation. Which arguments of a relation are essential or optional is defined apriori by the user. The remaining sentences are processed by a dependency parser outputting Stanford dependency relations[2] [25]. We use the output of the NER tagger to generalize the dependency parse by replacing all entity mentions with their respective NE tags.

3.2. Dependency structure extraction

The next step of the sar-graph construction process is to extract candidate dependency structures denoting the target relation from the full dependency parse trees of the source sentences. Typically, shortest path or minimum spanning tree algorithms are used to select the subgraph of the dependency tree connecting all the arguments of the relation instance mentioned in a given sentence [23]. In [22], we present an alternative, knowledge-driven algorithm which employs a large lexical semantic repository to guide the extraction of dependency structures. The algorithm expands the structure to include semantically relevant material outside the minimal subtree containing the shortest paths, and also allows us to discard structures without any explicit semantic content (e.g., highly ambiguous APPOS constructions).

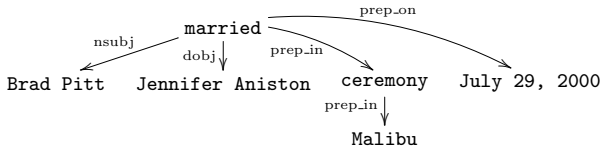
Figure 5 shows an example source sentence, along with a shortest-path dependency structure extracted from its parse tree. The example sentence (5a) mentions an instance of the *marriage* relation with the arguments ⟨Brad Pitt, Jennifer Aniston, Malibu, 2001/07/29⟩. In the figure, argument fillers are underlined. Figure 5b depicts the output of the dependency-structure extraction step. This structure is then generalized by replacing words with their lemmas, deriving coarse-grained part-of-speech tags, and replacing entity mentions with their respective NE tags (5c). We discard all structures which do not contain at least one content word, such as a verb, noun or adjective. We store word sense information for all content words as a property of the extracted dependency structure (not shown in the figure).

The use of the dependency-relation formalism for constructing sar-graphs is an important design choice. We assume that any given mention of a target-relation instance can be identified by a somehow characteristic pattern in the sentence's under-

[2] `http://nlp.stanford.edu/software/stanford-dependencies.shtml`

Brad Pitt married Jennifer Aniston in a private wedding ceremony in Malibu on July 29, 2000.

(a) Sentence with a mention of the *marriage* relation.



(b) Dependency parse of (a part of) (a).

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{lemma} & \texttt{marry} \\ \text{POS} & \text{V} \end{bmatrix} \\ \text{dobj} & \begin{bmatrix} \text{type} & \textsf{person} \\ \text{role} & \textsc{Spouse2} \end{bmatrix} \\ \text{prep\_in} & \begin{bmatrix} \text{head} & \begin{bmatrix} \text{lemma} & \texttt{ceremony} \\ \text{POS} & \text{N} \end{bmatrix} \\ \text{prep\_in} & \begin{bmatrix} \text{type} & \textsf{location} \\ \text{role} & \textsc{Ceremony} \end{bmatrix} \end{bmatrix} \\ \text{prep\_on} & \begin{bmatrix} \text{type} & \textsf{date} \\ \text{role} & \textsc{From} \end{bmatrix} \\ \text{nsubj} & \begin{bmatrix} \text{type} & \textsf{person} \\ \text{role} & \textsc{Spouse1} \end{bmatrix} \end{bmatrix}$$

(c) Generalized dependency construction derived from (b); WSD information omitted for clarity.

Figure 5: Data flow for gathering candidate dependency constructions from distantly labeled text.

lying dependency graph. This approach has limitations, e.g., it does not cover mentions requiring some kind of semantic understanding, or mentions with arguments spread across several sentences [26, 27], but it has been shown to work well in general [24, 28].

3.3. Dependency structure validation

Our approach to extracting relation-specific dependency structures is based on a distantly supervised learning scheme. Distant supervision makes several strong assumptions that may significantly affect the quality of the set of learned dependency structures. First, it assumes that for every relation tuple $r_i(e_{i_1}, \ldots, e_{i_k})$ in a knowledge base, every sentence containing mentions for $e_{i_1}, \ldots, e_{i_k}$ (or a subset thereof) expresses the relation r_i. This assumption typically does not hold for most sentences, i.e., the entity mentions may co-occur in a sentence without it actually expressing the relation. Extracted dependency structures may therefore be irrelevant or even wrong for a given relation, and should not be included in its sar-graph. Furthermore, distant supervision implicitly assumes that the knowledge base is complete: entity mentions without known relations are ignored during extraction. This may result in a loss of recall (of less frequent dependency structures), and in a bias of extracted dependency structures towards popular relations and entities.

The goal of the next step of sar-graph construction is therefore the validation of the quality of the candidate dependency structures. Validation can be performed automatically, e.g., by computing confidence values or similar metrics for each dependency structure, or by manually verifying structures. Candidate structures that have a low confidence score, or that are rejected during manual verification, are discarded. The remaining set of validated dependency structures is the output of this processing step.

We present and discuss several approaches to automatically and manually estimating the quality of candidate dependency structures in Section 4.

3.4. Dependency structure merging

Given the set of validated dependency constructions, we superimpose these structures onto one another to create a sar-graph. We follow a technically straight-forward approach to sar-graph creation by merging dependency constructions step-wise into larger graphs, based on the equality of properties of the graph elements. Initially, this process creates a graph by only merging argument nodes, while otherwise retaining the independence of structures. Figure 6 presents two functions in pseudocode that outline this first step. The input to the function **createSarGraph** is the set of dependency struc-

```
FUNCTION NAME: createSarGraph
INPUT:         DependencyConstruction[] dcs
OUTPUT:        a SarGraph
    // Initialize graph.
 1  SarGraph sg ← (V=∅, E=∅, s=∅, t=∅, f=∅, A_f, Σ_f)
 2  for each dc ∈ dcs :
        // Each dependency construction is a weakly connected, directed, simple graph.
 3      for each edge e in dc from n_1 to n_2 :
 4          e' ← new edge
 5          sg.E ← sg.E ∪ { e' }
 6          update function sg.s: Set sg.s(e') to result of addNode(sg, n_1)
 7          update function sg.t: Set sg.t(e') to result of addNode(sg, n_2)
 6          update function sg.f: Set sg.f(e') to attributes of e
 7  return sg
```

```
FUNCTION NAME: addNode
INPUT:         SarGraph sg, Node n
OUTPUT:        a Node
 1  if n ∈ sg.V then :
 2      return n
 3  elseif ∃n' ∈ sg.V | n, n' are derived from entity mentions
                       ∧ n, n' share entity type and argument role information then :
 4      update function sg.f: Extend sg.f(n') with attributes of n
 5      return n'
 6  else :
 7      sg.V ← sg.V ∪ { n }
 8      update function sg.f: Set sg.f(n) to attributes of n
 9      return n
10  endif
```

Figure 6: Pseudocode outlining the creation of a sar-graph from a set of dependency constructions. f, A_f, Σ_f are defined in Section 2. Nodes and edges of dependency constructions have the same attributes as sar-graph elements; see Table 1 for a list.

tures accepted by the previous validation step. The sar-graph is built by subsequently adding structures to the graph, one edge at a time. Whenever a node is to be added to the graph, it is first verified that the node is not already contained in the graph and checked whether there is a matching argument node present, in which case the history information of the currently handled node (identifiers of source sentences and dependency structure, statistical information) is merged with the information of the existing node. If neither is the case, the node is added to the sar-graph.

In order to deal with task-specific needs for the granularity of information in a sar-graph, applications can view sar-graphs at varying detail levels. For the task of relation extraction (see Section 7), the coverage of the original patterns is already very high [18], and merging paths would trade off higher recall with lower precision. Thus, the employed view does not impose any additional merging requirements and is identical to the originally constructed sar-graph. Figure 8b illustrates this strategy with a sar-graph constructed from the three example sentences shown in Figure 8a. The resulting sar-graph resembles the union of the original set of dependency structures, i.e., each path through the graph has a frequency of one.

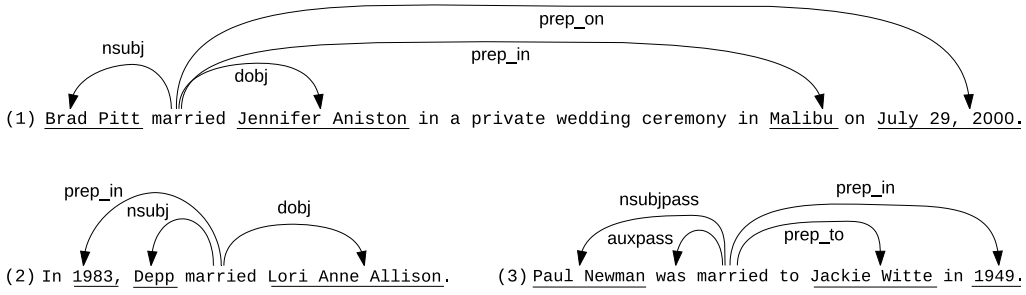
For analysis purposes, e.g., for carrying out an exploratory analysis of the linguistic expressions used to express particular target relations, a more condensed representation is advantageous. The pseudocode in Figure 7 shows the general workflow of the generation of sar-graph views in function **createCondensedSarGraphView**. Functions **exampleNodeCompressor** and **exampleEdgeCompressor** provide a custom implementation for the merging of nodes and edges. Two nodes are combined if they contain the same lexical information, likewise, edges between equal nodes are combined if the dependency labels attached to these edges are the same. In an application where a great number of linguistic expressions will be inspected, a user is likely just interested in a coarse-grained distinction of word classes, which is why **exampleNodeCompressor** generalizes the part-of-speech tags of all lexical nodes.

```
FUNCTION NAME: createCondensedSarGraphView
INPUT:         SarGraph sg, Function exampleNodeCompressor, Function exampleEdgeCompressor
OUTPUT:        a SarGraph
    // Initialize view on sar-graph.
 1  SarGraph sgView ← (V=∅, E=∅, s=∅, t=∅, f=∅, A_f, Σ_f)
 2  for each edge e ∈ sg.E :
 3      exampleEdgeCompressor(sg, sgView, e,
                              exampleNodeCompressor(sg, sgView, sg.s(e)),
                              exampleNodeCompressor(sg, sgView, sg.t(e)))
 4  return sgView
```

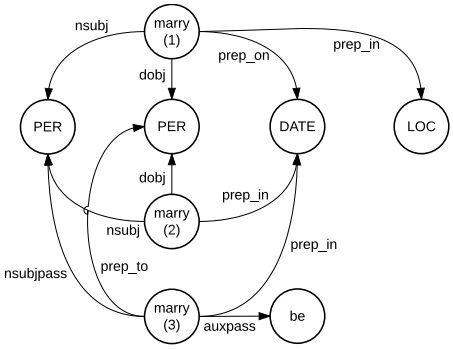
```
FUNCTION NAME: exampleNodeCompressor
INPUT:         SarGraph sg, SarGraph sgView, Node n
OUTPUT:        a Node
 1  if n ∈ sgView.V then :
 2      return n
 3  elseif n is derived from a lexical token then :
        // Generalize part-of-speech tag of n.
 4      update function sg.f: Replace ("word class", p) ∈ sg.f(n) with ("word class", upcast(p))
 5      if ∃n' ∈ sgView.V | n' is derived from a lexical token
                          ∧ n, n' share word form, word lemma, and word class then :
 6          update function sgView.f: Merge sg.f(n) into sgView.f(n')
 7          return n'
 8      endif
 9  endif
    // Neither is n contained in sgView, nor is there an equivalent node.
10  sgView.V ← sgView.V ∪ { n }
11  update function sgView.f: Set sgView.f(n) to sg.f(n)
12  return n
```

```
FUNCTION NAME: exampleEdgeCompressor
INPUT:         SarGraph sg, SarGraph sgView, Edge e, Node n1, Node n2
OUTPUT:        none
 1  if ∃e' ∈ sgView.E | e' originates from syntactic parsing
                      ∧ sgView.s(e') = n1 ∧ sgView.t(e') = n2
                      ∧ e, e' have the same dependency label
 2      update function sgView.f: Merge sg.f(e) into sgView.f(e')
 3  else :
 4      e' ← new edge
 5      sgView.E ← sgView.E ∪ { e' }
 6      update function sgView.s: Set sgView.s(e') to n1
 7      update function sgView.t: Set sgView.t(e') to n2
 8      update function sgView.f: Set sgView.f(e') to sg.f(e)
 9  endif
```

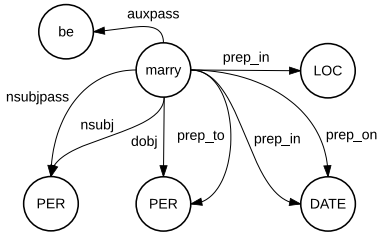
Figure 7: Pseudocode for producing a condensed view of a sar-graph, tailored for applications. f, A_f, Σ_f are defined in Section 2. In this example, the call **createCondensedSarGraphView**(*sg*, **exampleNodeCompressor**, **exampleEdgeCompressor**) generates a sar-graph suited for manual explorative analysis of linguistic phrases. The produced graph uses a coarse-grained inventory of part-of-speech tags. The function **upcast**() generalizes a given tag, e.g., it maps verb classes (*verb in past tense*, *verb in 3rd person singular present*, ...) to a single base verb class.



(a) Example sentences and dependency structures.



(b) A sar-graph retaining the independence of original structures.



(c) A more condensed representation of linguistic phrases.

Figure 8: Two different sar-graph views created from the same three sentences..

This strategy merges all nodes and vertices that are equal according to the above definition. Structures that fully or partially overlap (even with just a single edge or node) are merged. This could mean that in the resulting sar-graph, some of the paths connecting argument nodes are linguistically invalid. The frequency of a dependency edge in the sar-graph is equal to the number of dependency structures containing that edge. Since the same dependency structure can appear multiple times in the source data, with different arguments and / or context, we represent word sense information as a frequency distribution (over senses of content words for a given dependency structure). This approach enables a more flexible and noise-resistant annotation of word senses for the context words used to express the target relation. Figure 8c shows an example sar-graph created with this strategy.

In order to cope with applications which require a different balance of detail vs. generalization of the various sar-graph elements, all one has to do is to provide matching implementations of functions **exampleNodeCompressor** and **exampleEdgeCompressor**. For example, dependency structures could be generalized by merging all vertices belonging to the same synset in a lexical-semantic resource, ignoring differences on the lexical level.

4. Quality control

As discussed in the previous section, our approach to sar-graph construction uses distant supervision for collecting textual mentions of a given target relation. In this section, we present several approaches to automatically compute confidence metrics for candidate dependency structures, and to learn validation thresholds. We also describe an annotation process for manual, expert-driven quality control of extracted dependency structures, and

briefly describe the linguistic annotation tool and guidelines that we developed for this purpose.

4.1. Automatic ways of quality improvement

4.1.1. Data-oriented path-quality estimation

Semantic relations coming from the same domain might have a similar entity-type signature, in particular, they might share the types of their essential arguments. For example, numerous semantic relations can be defined for the great variety of ways persons interact and relate to one another. Whenever two relation definitions are similar in this particular way, we say they are of the same *essential type*.

Relations of the same essential type may have some instances in common, for example, the same two persons might be involved in various relations such as marriage and romantic relationships. This can be the case, for example, if the relations overlap, or if the relevant linguistic expressions are ambiguous. Most dependency constructions we find for two or more relations, however, are not appropriate for one or both relations. Such constructions might be learned for wrong relations because of erroneous entity recognition and dependency parsing, false seed facts, or false statements of a relation in texts. Especially when we extract the same dependency construction for two disjoint relations, something must be wrong. Either the construction exhibits a much higher frequency for one of the two relations, then it can be safely deleted from the other, or we consider it wrong for both relations.

In [18] we proposed a general and parameterizable confidence estimation strategy for dependency structures using information about their frequency distribution wrt. other relations of the same essential type. If a construction occurs significantly more often in a relation r than in another relation r', this construction probably expresses r in contrast to r'. Let $\mathcal{D}_{r,l}$ be the set of extracted dependency structures for r and language l, and let $f_{r,l}(d)$ denote the frequency of dependency structure d in r, l (i.e., the number of sentences for relation r and language l from which d has been extracted). We define the relative frequency of d for r, l as:

$$rf_{r,l}(d) = f_{r,l}(d) \Big/ \sum_{d' \in \mathcal{D}_{r,l}} f_{r,l}(d') \qquad (1)$$

Let $\mathcal{R}$ be a set of relations of the same essential type. The dependency structure d most likely expresses the relation $r \in \mathcal{R}$ in l if the relative frequency of d in r, l is higher than its relative frequencies for all other relations in $\mathcal{R}$ (i.e., if $\forall r' \in \mathcal{R} \backslash \{r\} : rf_{r,l}(d) > rf_{r',l}(d)$). Because this judgment about the semantic meaning of a dependency structure depends much on the specific set of selected target relations, we do not use it for the construction of sar-graphs in general, however, it proves useful for particular applications (Section 7).

Instead of excluding dependency structures with this *relation-overlap* heuristic, we augment individual dependency paths in the sar-graphs with information about their frequency wrt. a single relation. This allows applications to pick certain sub-parts of the sar-graphs for which there is much support in the training data, i.e., evidence that a structure belongs to the target relation from an absolute point of view. Depending on application needs, this can be combined with information from the relation-overlap criterion.

4.1.2. Utilizing external resources

Another automatic quality estimator for relation-specific dependency structures can be defined through the construction of the so-called relation-specific semantic graphs [21]. The considered dependency structures, and consequently the sar-graphs, already contain semantic information that can be exploited for different tasks (see Section 2.3). In this section, we show how we use this information to improve the quality of the generated sar-graphs. In comparison with statistical methods, the use of semantic analysis can better handle cases of high-frequency structures which do not express the considered relation (e.g., person $\xleftarrow{subj}$ met $\xrightarrow{obj}$ person for the relation *marriage*) and also in cases of low frequency structures which are indeed semantically relevant for the considered semantic relation (e.g., person $\xleftarrow{poss}$ widower $\xrightarrow{appos}$ person for the same relation).

Given the frequency distributions of content word meanings (i.e., senses) encoded within the dependency structures, we can produce an overall frequency distribution of all the considered meanings for a relation r. Then, thanks to the links to a lexical-semantic resource, we can induce a semantic graph from it which contains the most relevant meanings for the considered relation.

More precisely, we first get the top-k most frequent meanings (i.e., the core senses of the relation) from the overall distribution of meanings. For

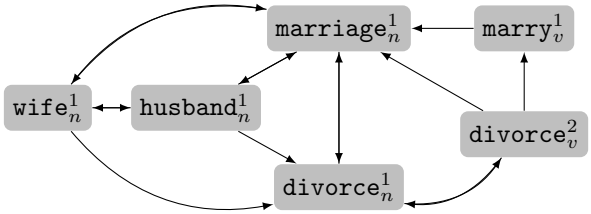


Figure 9: An excerpt of the semantic graph associated with the relation *marriage* with $k = 2$.

example, with $k = 2$ and the relation *marriage*, the core meanings are $\{\texttt{marry}^1_v, \texttt{wife}^1_n\}$.[3] Then, we add all the remaining meanings in the overall distribution if and only if they are connected to at least one of the core meanings. For example, with $k = 2$ and the relation *marriage*, we add $\texttt{husband}^1_n$, $\texttt{marriage}^1_n$ and $\texttt{divorce}^2_v$ to the graph, among others, but we do not add $\texttt{meet}^1_v$. In this manner we are able to extract a set of highly-relevant meanings for the considered relation (see Figure 9 [21] for an excerpt of the semantic graph for the *marriage* relation). These highly-relevant meanings likely constitute most of the senses of the lexical-semantic resource which are useful for expressing the target relation in natural language.

Finally, to filter out dependency structures which do not contain any relation-relevant lexical semantic elements, we check if any of the dependency structure's content words matches a lexicalization of the meanings contained in the semantic graph. If that is the case we mark it as a good structure, otherwise we filter it out. For instance, our filter recognizes person $\xleftarrow{subj}$ married $\xrightarrow{obj}$ person as a good rule, while it filters out person $\xleftarrow{subj}$ met $\xrightarrow{obj}$ person because it does not match any lexicalizations of the meanings contained in the semantic graph.

By generating relation-specific semantic graphs for various values of k and repeatedly applying the corresponding filter, we can estimate the degree of relevancy for all dependency structures. If a structure is accepted by the filter with a low k it is more likely to express the relation than a structure only accepted at a greater value of k. When constructing the sar-graph from the individual dependency structures, we choose not to filter out any structures, but rather associate the information about the filter output with them.

[3]For ease of readability, in what follows we use senses to denote the corresponding synsets. We follow [29] and denote with w^i_p the i-th sense of w with part of speech p.

4.2. Expert-driven quality control

The automatic estimation of dependency-structure quality described in the previous section is limited to statistical / distributional metrics and to a metric based on the lexical semantics of words appearing in the structure. These metrics, however, tell us only very little about the (grammatical) correctness and semantic appropriateness of the dependency structures themselves. Therefore, we developed a process for a manual, intrinsic evaluation of the learned dependency structures. This expert-driven quality control has two major goals: to validate the structures selected by automatic means for the subsequent construction of sar-graphs, and to identify common classes of extraction errors. In this section, we describe the tools and guidelines that we developed for the manual evaluation process.

4.2.1. Selection of dependency structures

Since the number of possible dependency structures expressing a given relation is potentially unbounded, a complete manual evaluation would be too resource-intensive. We therefore limit the expert-driven quality control to a subset of structures, as chosen by the following process: For each relation and dependency structure, we first compute an automatic quality metric (e.g., the semantic-graph score presented in the previous section), and also determine the structure's relation-specific occurrence frequency in a large web corpus. Per relation, we experimentally determine threshold values for these two measures to exclude low-confidence and low-frequency structures (see Section 7). We then sample a small set of sentences for each structure, and conduct an initial pass over the data with human annotators that judge whether these sentences express the target relation or not. We discard all dependency structures whose sentences do not express the target relation. The manual evaluation dataset is then created from the remaining dependency structures. For each structure and relation, the final dataset comprises all source sentences and not just the ones sampled for the initial judgments.

4.2.2. Quality control guidelines

Based on an initial, exploratory analysis of the dataset, we define three qualitative categories, "CORRECT", "CORRECT, BUT TOO SPECIFIC" and "INCORRECT", as well as a set of annotation guidelines for the evaluation of dependency structures.

We label a relation-specific structure as CORRECT (i.e., as useful for integration into a sar-graph) if it is grammatically *and* semantically correct. A dependency structure is grammatically correct if there are no parsing or other preprocessing errors, and it is semantically correct if its source sentences express the target relation. Correspondingly, we label a dependency structure as INCORRECT if it is grammatically incorrect, or if it does not express the target relation. Typically, the annotators aim to identify one or more of the error classes described in Section 5.4 to decide whether a pattern is incorrect.

For deciding whether a sentence expresses a given relation, we use the ACE annotation guidelines' conceptual definition of relations and their mentions [30], and define the semantics of relations based on Freebase descriptions (see Section 5). In contrast to the ACE tasks, we also consider n-ary relations in addition to binary relations. In the course of this evaluation, sentences must express the target relation explicitly, e.g., *"X won the Y award"* explicitly expresses the relation *award honor*. We treat implicit mentions as semantically incorrect, e.g., *"X won the Y award"* does not imply the relation *award nomination* as this implication requires knowledge about relation entailments. A dependency structure that captures only a subset of all arguments mentioned in a sentence (e.g., it covers only one of several children of the same parent listed in the same sentence) is still considered correct.

A third category, CORRECT, BUT TOO SPECIFIC, was added based on our initial analysis of the dataset, and applies to dependency structures mostly found in the long tail of the frequency distribution. Too specific structures, while both grammatically and semantically correct, are structures that are overly complex and/or include irrelevant parts of the sentence specific to a particular relation instance. Figure 10 shows an example structure, which includes the head word `voice`. Such dependency structures do not generalize well, and are hence unlikely to be very "productive" for many application tasks (e.g., they are unlikely to yield novel relation instances when applied to additional text). The distinction between CORRECT and CORRECT, BUT TOO SPECIFIC is often not clear-cut; to improve the consistency of annotation decisions, we collected illustrative examples in the annotation guidelines.

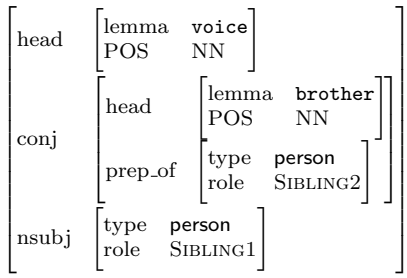


Figure 10: "CORRECT, BUT TOO SPECIFIC" dependency structure extracted from the sentence "`Jansen Panettiere is an American voice and film actor, and the younger brother of actress Hayden Panettiere.`" for the relation *sibling relationship*.

4.2.3. Evaluation tool - PatternJudge

To facilitate the manual evaluation of dependency structures, we have developed a simple annotation tool, dubbed *PatternJudge*. With *PatternJudge*, annotators can inspect dependency structures (patterns) and their associated source sentences (learning tracks), and evaluate the structures' grammatical and semantic correctness.

Figure 11 shows a screen shot of the user interface. The interface is split into three main components. The left part displays the list of relations and patterns available for judgment, and allows searching for specific pattern or sentences. The center part visualizes the currently selected dependency structure in attribute-value-matrix notation, and lists the source sentences this structure was observed in. The annotation tab on the right-hand side collects the human expert's feedback on the quality of this pattern. Current options include labeling the pattern as "CORRECT", "CORRECT, BUT TOO SPECIFIC", "INCORRECT" or "UNCERTAIN/DON'T KNOW". In addition, annotators can provide a comment. Comments are mainly used for discussion and clarification, but also for adding error class information in cases where the annotator decided to label a pattern as INCORRECT. All pattern judgments are persisted in a database. The tool includes a simple user management, which enables keeping track of different annotators, and undoing or updating previous judgments (which is particularly useful in the early stages of pattern exploration and analysis).

5. Implementation

So far, we have described our methodology for creating the proposed resource of combined lexical,

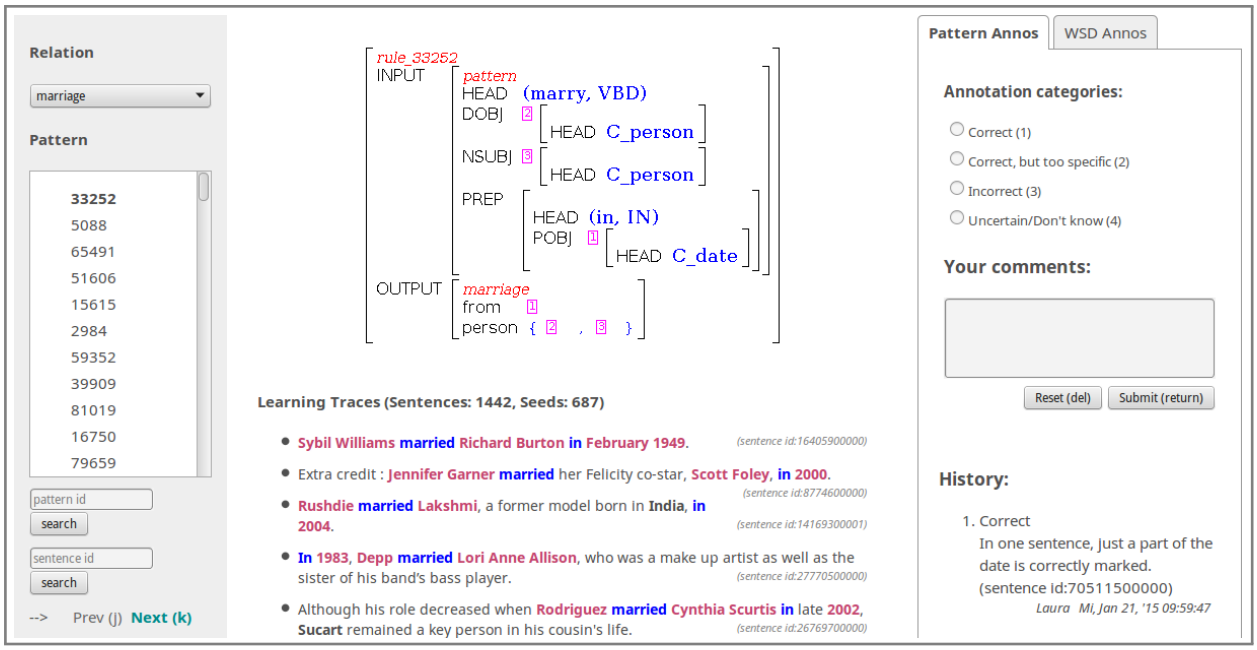


Figure 11: User interface of the *PatternJudge* tool. The tool allows annotators to judge the quality of automatically learned dependency structures.

syntactic, and lexical semantic information. In this section, we outline the concrete experiments carried out to compile sar-graphs for 25 semantic relations.

5.1. Leveraging factual knowledge

We kick-off the construction process by leveraging factual knowledge from *Freebase*, a large knowledge base containing millions of assertions about numerous entities such as people, locations, organizations, films, and books. For our experiments (see also [18]) we focus on a set of 25 relations from the domains *Award*, *Business*, *People* for which we expect to find mentions in human texts and which we deem fruitful wrt. application scenarios.

Table 2 lists the target relations along with their entity-type signature, grouped by solid horizontal lines wrt. their domain. Essential arguments are marked by ⊛. All relations from a domain have the entity type of the first essential argument in common. If two such relations share the entity type of another essential argument, we say that they are of the same *essential type*. For example, the following *Business* domain relations all belong to the same essential type since their first two arguments allow entities of type organization:

- *acquisition*,
- *foundation*,
- *organization alternate name*,
- *organization membership*,
- *organization relationship*,
- *sponsorship*.

All relation definitions used in this paper are based on the data available in Freebase. By utilizing Freebase' query API, we retrieved several thousand instances per target relation, yielding a total of 233K seeds for the 25 target relations. Table 3 lists the distribution of seeds per relation.

5.2. Creating sar-graphs from web text

The next step was concerned with the acquisition of a corpus of relation-mention examples. We decided against using an offline dataset (e.g., a Wikipedia crawl) because the processing of such would restrict collected phrases to a particular style of writing. Furthermore, the infamous long-tail problem of linguistic expressions would make finding sentences for less prominent facts improbable. Instead, we directly accessed the Web via a search engine.

Search-engine querying. The relation instances from Freebase were transformed to search-engine queries and submitted to *Bing*. We stopped the querying of Bing early in case one million search results per relation had already been retrieved. This

Relation	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
award nomination	⊛ prize	⊛ org/per	date	—	—
award honor	⊛ prize	⊛ org/per	date	—	—
country of nationality	⊛ per	⊛ loc	—	—	—
education	⊛ per	⊛ org	degree	area	date
marriage	⊛ per	⊛ per	loc (CEREMONY)	date (FROM)	date (TO)
person alternate name	⊛ per	⊛ per	—	—	—
person birth	⊛ per	(⊛) loc	(⊛) date	—	—
person death	⊛ per	(⊛) loc	(⊛) date	(⊛) cause	—
person parent	⊛ per (PERSON)	⊛ per (PARENTA)	per (PARENTB)	—	—
person religion	⊛ per	⊛ religion	—	—	—
place lived	⊛ per	⊛ loc	date (FROM)	date (TO)	—
sibling relationship	⊛ per	⊛ per	—	—	—
acquisition	⊛ org (BUYER)	⊛ org (ACQUIRED)	org (SELLER)	date	—
business operation	⊛ org	⊛ business space	—	—	—
company end	⊛ org	(⊛) date	(⊛) termination type	—	—
company product relationship	⊛ org	⊛ product	date (FROM)	date (TO)	—
employment tenure	⊛ org	⊛ per	position	date (FROM)	date (TO)
foundation	⊛ org (ORG)	⊛ org/per (FOUNDER)	loc	date	—
headquarters	⊛ org	⊛ loc	—	—	—
organization alternate name	⊛ org	⊛ org	—	—	—
organization leadership	⊛ org	⊛ per	position	date (FROM)	date (TO)
organization membership	⊛ org (ORG)	⊛ loc/org/per (MEMBER)	date (FROM)	date (TO)	—
organization relationship	⊛ org (PARENT)	⊛ org (CHILD)	date (FROM)	date (TO)	—
organization type	⊛ org	⊛ org type	—	—	—
sponsorship	⊛ org (SPONSOR)	⊛ org/per (RECIPIENT)	date (FROM)	date (TO)	—

Table 2: Definition of the 25 target relations of the domains *Award*, *Business* and *People*. ⊛ denotes the essential arguments of the relation, i.e., the core part of a relation instance defining its identity. (⊛) marks alternatives for essential arguments. loc/org/per are short for location/organization/person. Labels for arguments (in SMALLCAPS) omitted in unambiguous cases.

Relation	# seeds	# doc.	# sent.	# struct.	# nodes	# edges
award honor	11,013	50,680	16,651	10,522	4,349	18,101
award nomination	12,969	14,245	2,842	1,297	983	3,173
country of nationality	5,650	94,400	74,286	59,727	24,554	159,857
education	15,761	61,005	28,723	16,809	8,216	39,266
marriage	6,294	211,186	147,495	88,456	24,169	169,774
person alternate name	6,807	42,299	15,334	7,796	6,588	22,917
person birth	1,808	329,387	39,484	22,377	10,709	46,432
person death	1,437	241,447	38,775	31,559	14,658	73,069
person parent	3,447	148,598	58,541	45,093	15,156	85,528
person religion	8,281	48,902	39,439	37,086	19,221	113,651
place lived	5,259	89,682	57,840	48,158	20,641	120,239
sibling relationship	8,246	130,448	45,201	26,250	13,985	68,132
acquisition	1,768	40,541	30,116	26,986	11,235	64,711
business operation	12,607	51,718	31,274	15,376	10,657	47,116
company end	1,689	14,790	7,839	5,743	4,964	17,413
company product relationship	6,467	27,243	19,007	15,902	10,358	47,266
employment tenure	10,000	116,161	51,848	43,454	15,151	92,810
foundation	1,529	131,951	61,524	31,570	13,124	72,320
headquarters	1,987	79,731	33,255	23,690	11,420	54,715
organization alternate name	8,011	70,595	29,523	10,419	8,410	32,137
organization leadership	21,579	138,952	74,029	51,295	17,864	115,296
organization membership	4,180	50,061	32,646	29,220	13,326	76,532
organization relationship	70,946	37,475	17,167	12,014	7,030	32,247
organization type	4,625	3,939	1,391	843	1,445	3,474
sponsorship	1,513	11,009	5,395	4,599	4,030	13,813
average	9,355	89,458	38,385	26,650	11,690	63,600
sum	233,873	2,236,445	959,625	666,241	292,243	1,589,989

Table 3: Statistics for various steps of the sar-graph creation process, as well as for the produced sar-graphs. *# doc.* refers to the number of web documents in which a relation mention was found; no duplicate detection was performed. *# sent.* states the count of duplicate-free sentences with a relation mention, *# struct.* the number of unique dependency structures learned from these sentences. *# nodes/#, edges* corresponds to the number of respective elements in the sar-graphs created from the structures in column five.

limit was arbitrarily chosen, as at this point of the system run, no data was available on how much search results would be needed to create a satisfactorily large text corpus for a relation. One million seemed both large enough for generating enough samples and small enough to keep the needed processing time at a reasonable level. To reach the limit, for some relations not all the facts stored in Freebase had to be utilized. This is particularly true for the relations of the *People* domain, in contrast to the domains *Award* and *Business*. A possible explanation is that there are less web pages dealing with *Award* and *Business* related topics than there are for *People* relations. It might also be the case that the instances of the former domains are simply less prominent in current web pages and more of historical character. This concurs with an on average greater absolute number of instances for *People* relations, which is not surprising given that Freebase (in part) utilized *Wikipedia* for gathering knowledge.[4]

Text retrieval and entity recognition. The search results were subsequently processed by downloading the respective web page and extracting plain text from the HTML source code. This process suffered from various problems, leading to a fraction of "lost" documents up to 40 % for some relations (e.g., *person death*). Among the reasons for the losses are problems when accessing web pages (connection timeouts, ...), issues when extracting text from them (malformed HTML code, insufficiencies of text-extraction algorithm), and web pages which did not contain any article text at all.

After the creation of the text corpus, we ran the previously outlined entity-recognition components on it to find occurrences of named entities, in particular those of the respective documents source seed. For the recognition of coarse-grained types (i.e., persons, organizations and locations, we relied on the *Stanford Named Entity Recognizer* as part of the *Stanford CoreNLP* package[5] [31] and supplemented this with our own date recognizer. To identify the seed entities, we implemented a simple gazetter-based recognizer with the name variations of the seeds' entities as provided by Freebase.

Relation-mention detection. Only for a relatively small fraction of the search-result web-page addresses we got from Bing, we eventually end up with a plain-text document in which we detected an intra-sentential relation mention. The actual number of such documents per relation is given in column three of Table 3. Documents which were classified as not being written in English account for a large fraction of the successfully downloaded, but still unproductive documents. By far the most documents fail because for at least one essential argument of the source seed no entity occurrence was found anywhere in the text. This means that the search engine returned results which indeed did not contain all essential seed arguments, despite the queries always containing them. Another explanation for documents without a mention is of course that the NER component was unable to locate the seeds argument in the text.

Column four of Table 3 lists the number of *unique* sentences with relation mentions we extracted per relation. The large difference to the number of documents with an intra-sentential relation mention can be attributed to (a) duplicate and near-duplicate documents retrieved from the Web[6] and (b) errors in the sentence-processing pipeline.[7] Also note that these sentences are duplicate-free.

The average number of mention-containing sentences per document is approximately 1.5. This is reasonable given the underlying assumption [17] that any such sentence indeed expresses the target relation. A higher number of sentences might come with a low quality of the training examples, as for some relations it is not likely that an entity tuple is referred to multiple times within the same document as an instance of this very relation.

Dependency structure extraction and merge. After the identification of sentences containing mentions, the sentences were processed by a dependency parser (*MaltParser*[8], *MDParser*[9]) outputting Stanford dependency relations, followed by the extraction of the minimum spanning tree containing all the seed's arguments present in the sentence. Trees are also extracted for all argument

[4] Freebase contains 3M topics and 20M facts for the domain People, more than for the domain Business (1M topics and 4M facts). Retrieved from `http://www.freebase.com/` on 2015/03/25.

[5] `http://nlp.stanford.edu/software/corenlp.shtml`

[6] Multiple URLs for the same web page, web pages just differing in boilerplate/navigational elements, extensive reuse of paragraphs from old articles or from press agencies.

[7] Garbled sentences, overly long sentences, parser errors in next step, conflicts between NER and parser output.

[8] Release v1.7.2, engmalt-linear model v1.7, `http://www.maltparser.org/`

[9] `http://mdparser.sb.dfki.de/`

subsets where at least two essential arguments are present in the sentence, i.e., projections of the dependency tree which corresponds to the full set of arguments are extracted as well. The dependency structures were then assembled into the sar-graph, as described in Section 3.4. The final sar-graphs for the 25 relations range in size from 1k to 25k vertices and 3K to 170k edges each.[10]

5.3. Augmenting the sar-graphs with lexical-semantic information

We grounded the sar-graphs' dependency structures by linking them against the lexical-semantic resource *BabelNet*.

BabelNet. BabelNet [12] is a large-scale multilingual semantic network which, differently from the manually created WordNet [32], was automatically built through the algorithmic integration of resources like Wikipedia and WordNet, among others. Its core components are so-called Babel synsets, which are sets of multilingual synonyms. Each Babel synset is related to other Babel synsets via semantic relations obtained from WordNet and Wikipedia, such as hypernymy, meronymy and semantic relatedness. Moreover, since BabelNet is the result of the integration of lexical resources and encyclopedic resources, it goes exactly in the same direction of the multilingual linguistic Linked Open Data project [33] which consists of a vision of the Semantic Web in which a wealth of linguistic resources are interlinked to each other to obtain a bigger and optimal representation of knowledge [34].

BabelNet contains roughly 13M synsets, 117M lexicalizations and 354M relation instances. Given the multilingual nature of BabelNet (it considers 271 different languages), this resource can exploit multilinguality to perform state-of-the-art knowledge-based Word Sense Disambiguation [35] (in contrast to WordNet that encodes only English lexicalizations), thereby enabling new methods for the automatic understanding of the multilingual (Semantic) Web.

Semantic graphs. The sar-graph construction was finalized by utilizing the lexical semantic links to BabelNet for the creation of the relation-specific semantic graphs, as outlined in Section 4.1.2. The obtained information about the relation-relevancy of dependency structures was subsequently integrated into the sar-graphs.

5.4. Pattern observations from sar-graph validation

We conducted an error analysis of dependency structures for all relation types using the *PatternJudge* tool. The relation *sibling relationship* served as an initial test bed for establishing a set of shared evaluation principles among the annotators, who then screened the patterns in the remaining relations for recurring errors. There were three annotators in total; they identified six main error classes, which are listed in Table 4.

Three of the classes describe errors based on defective output of the preprocessing pipeline, namely sentence boundary detection, named entity recognition and parse tree generation. We label these error types *PIPE-S*, *PIPE-NER* and *PIPE-PT*. The other three types refer to errors that cannot be attributed to accuracy deficits of linguistic preprocessing. Semantic understanding is required in order to detect them. The first class corresponds to patterns that do not express the relation of interest (*NEX-P*), whereas the other two describe dependency structures generated from sentences that either do not express the target relation (*NEX-S*) or do so, but in a way that is too implicit (*IMP-S*). Because relevance and explicitness are, to some degree, vague criteria, this second set of error classes is more susceptible to debate than the first. We used the guidelines developed during the iterative discussion of the *sibling relationship* relation to define boundaries and to help evaluate borderline cases.

The category *PIPE-S* pertains to ungrammatical sentence and dependency structures resulting from sentence boundary detection errors. In example (1) in Table 4, the tokens "`Personal life`" (which are most likely a headline / news category identifier) are not only interpreted as part of the sentence, but also as relevant elements of the extracted pattern.

PIPE-NER is the error class that is used for patterns which contain arguments that are semantically or grammatically incongruent with the ones tagged in the sentence. In example (2), for the relation *award honor*, the title of the book has not been recognized as an entity. The inclusion of the lemmas "`leave`" and "`us`" as lexical nodes results in a pattern that is unlikely to be applicable to other text. The error class also includes instances

[10] Note that the sar-graphs also contain many dependency structures that do not always signal instances of the target relation. Instead of filtering these out, we associate them with confidence values determined by a semantic filter and by their positive and negative yield, see Section 4.

#	Error class	Description	Example
1	PIPE-S	Sentence segmentation error	Personal life On July 5, 2003, Banks married sportswriter and producer Max Handelman, who had been her boyfriend since she met him on her first day at college, September 6, 1992. *(marriage)*
2	PIPE-NER	NER tagging error	Rahna Reiko Rizzuto is the author of the novel, Why She Left Us, which won an American Book Award in 2000. *(award honor)*
3	PIPE-PT	Dependency parsing error	*Say won a Caldecott Medal for his illustrations in Grandfather's Journey. *(award honor)*
4	NEX-P	Relation is not expressed in pattern	Julian joined Old Mutual in August 2000 as Group Finance Director, moving on to become CEO of Skandia following its purchase by Old Mutual in February 2006. *(acquisition)*
5	NEX-S	Relation is not expressed in text	The 69th Annual Peabody Awards ceremony will be held on May 17 at the Waldorf-Astoria in New York City and will be hosted by Diane Sawyer, the award-winning anchor of ABCs World News. *(award honor)*
6	IMP-S	Relation is too implicit	The looming expiration of Lipitors patent in 2012 is a big reason Pfizer felt compelled to buy a company like Wyeth. *(acquisition)*

Table 4: Common error classes of dependency structures for the relations *marriage*, *acquisition* and *award honor*. Underlined token sequences denote relation arguments, concepts with a wavy underline are additional pattern elements. Error classes are described in more detail in Section 5.4.

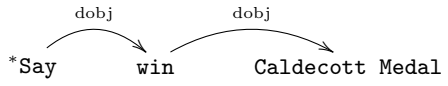


Figure 12: Erroneous dependency structure extracted from the sentence "Say won a Caldecott Medal for his illustrations in Grandfather's Journey."

in which named entities are assigned to the wrong semantic categories (e.g., a person mention appears in the parse tree as an organization).

The category *PIPE-PT* is applied to dependency structures extracted from defective parse trees. The example sentence (3) is erroneously parsed as shown in Figure 12, with the proper name Say interpreted as a finite verb. Other typical parsing errors we observed included wrong PP attachment, conjunctions attached as dependents of one of the conjuncts, or the inclusion of superfluous punctuation tokens in the parse tree.

The category *NEX-P* is used for dependency structures that do not include any relation-relevant content words occurring in the sentence. In example (4), the most explicit element expressing an acquisition is the lemma "purchase". The pattern does not include this word, but focuses on less relevant parts of the associated source sentence.

NEX-S applies to dependency structures that are based on sentences which do not express the relation of interest, i.e., sentences which violate the distant supervision assumption.[11] In example (5), the target relation *award honor* is not expressed. Instead, the host of the ceremony is erroneously identified as the winner of the prize. This example also highlights an additional difficulty of the distant supervision assumption: The source sentence reports an event that lies in the future. Similar errors may occur for sentences containing modal verbs or reporting fictional events.

The category *IMP-S* marks dependency structures based on sentences in which a relation is expressed merely implicitly. Judging from the source sentence in example (6), we cannot be entirely sure whether or not an acquisition took place because "felt compelled to" might only express a momentary mindset of the company's leaders that was not followed by action. If it was, it is not clear if "Wyeth" or "a company like Wyeth" (i.e., a similar company) was acquired.

We limit the expert-driven quality control to a subset of structures as described in Section 4.2.1. Table 5 presents the results of the error analysis. The second column lists for each relation the number of dependency structures contained in this manual evaluation dataset. The left part of Table 5 summarizes the distribution of correct and incorrect dependency structures for all relations. We find that between 25.7% and 80.4% of the learned

[11] Although we applied this criterion when creating the dataset, at the time it was based on a very small sample.

Relation	*Evaluation categories*					*Error classes (for incorrect patterns)*						
	Total	Correct	Correct, but too specific	Uncertain	Incorrect	PIPE			NEX		IMP-S	Other
						-S	-NER	-PT	-P	-S		
award honor	510	52.3%	12.9%	9.0%	25.7%	0.7%	56.8%	27.7%	4.1%	2.7%	3.4%	4.7%
award nomination	392	17.0%	19.5%	1.2%	62.3%	7.3%	56.8%	12.4%	16.2%	0.8%	5.0%	1.5%
country of nationality	560	14.6%	36.5%	1.4%	47.4%	10.7%	0.7%	17.8%	12.8%	19.1%	37.6%	1.3%
education	270	45.9%	22.6%	1.1%	30.4%	2.2%	2.2%	48.9%	32.6%	2.2%	9.8%	2.2%
marriage	451	39.1%	26.5%	7.2%	27.1%	7.9%	7.3%	13.9%	19.2%	1.3%	23.2%	27.2%
person altern. name	542	7.3%	18.9%	1.3%	72.5%	27.0%	6.9%	20.3%	30.3%	5.0%	8.5%	2.1%
person birth	151	40.4%	17.2%	0.7%	41.7%	10.0%	2.9%	24.3%	42.9%	4.3%	14.3%	1.4%
person death	306	64.0%	8.7%	0.3%	27.0%	10.0%	5.6%	32.2%	44.4%	0.0%	3.3%	4.4%
person parent	387	46.9%	17.6%	3.4%	32.0%	3.0%	3.0%	32.6%	51.5%	1.5%	1.5%	6.8%
person religion	142	42.9%	21.1%	0.0%	36.1%	0.0%	3.6%	14.3%	44.6%	7.1%	28.6%	1.8%
place lived	329	20.7%	5.5%	0.3%	73.6%	1.7%	3.8%	17.4%	30.7%	1.0%	45.1%	0.3%
sibling relationship	140	38.7%	22.7%	10.5%	28.2%	6.9%	1.4%	8.3%	5.6%	0.0%	6.9%	70.8%
acquisition	224	21.2%	14.8%	3.6%	60.4%	1.2%	1.9%	3.1%	12.5%	78.8%	0.6%	1.9%
business operation	264	34.0%	7.6%	2.1%	56.4%	2.9%	10.9%	24.0%	14.3%	40.0%	5.7%	2.3%
company end	144	9.7%	7.1%	2.8%	80.4%	5.5%	3.8%	13.1%	38.2%	10.9%	26.1%	2.4%
company product rel.	257	29.2%	22.3%	3.8%	44.7%	3.5%	1.4%	16.0%	36.1%	20.1%	16.7%	6.2%
employm. tenure	226	64.5%	5.9%	3.5%	26.2%	0.0%	1.4%	43.7%	31.0%	7.0%	5.6%	11.3%
foundation	397	48.6%	13.6%	0.5%	37.3%	4.3%	6.5%	42.9%	31.5%	0.5%	9.8%	4.3%
headquarters	273	33.2%	20.7%	3.2%	42.9%	13.3%	4.4%	15.6%	22.2%	29.6%	11.1%	3.7%
organiz. altern. name	280	20.3%	5.8%	1.7%	72.2%	6.5%	7.8%	56.0%	14.2%	2.6%	11.2%	1.7%
organiz. leadership	547	63.8%	1.6%	4.5%	30.1%	1.2%	5.8%	52.6%	29.8%	4.1%	1.8%	4.7%
organiz. membership	291	53.9%	8.8%	3.1%	34.2%	0.9%	8.5%	41.0%	31.6%	7.7%	6.8%	3.4%
organiz. rel.	303	30.9%	6.2%	0.7%	62.2%	5.4%	1.3%	36.8%	38.9%	1.3%	15.9%	0.4%
organiz. type	264	12.1%	15.9%	0.4%	71.6%	3.1%	35.0%	18.5%	34.6%	5.9%	2.4%	0.3%
sponsorship	336	36.0%	12.5%	0.3%	51.2%	6.6%	13.8%	43.1%	16.0%	9.4%	5.0%	6.1%
All relations	8307	35.0%	15.1%	2.9%	47.0%	7.0%	11.3%	25.3%	27.1%	10.3%	14.3%	4.6%

Table 5: Distribution of evaluation categories and error classes for dependency patterns manually reviewed using the PatternJudge tool. The table lists the total number of evaluated patterns per relation, and the distribution across categories. It also shows the distribution of error classes for the Incorrect category.

dependency structures are erroneous, between 7.3% and 64.5% are labeled as correct. For example, more than 70% of the patterns of the relation *organization alternate name* are labeled as INCORRECT. CORRECT, BUT TOO SPECIFIC patterns make up between 1.6% and 36.5% of the total number of patterns.

The right-hand part of the table gives details on the distribution of the error classes. The two predominant error classes parsing errors (PIPE-PT) and pattern extraction errors (NEX-P). We observe that the distribution of the error classes varies between the different relations: *PIPE-NER* is the error type most frequently occurring for *award honor* and *award nomination* patterns. Sentences in this category often mention the titles of works the prize was awarded for. If those titles are not recognized as entities by the *NER* tagger, the dependency parsing fails and parts of the title can erroneously end up in the extracted dependency structure. For the *acquisition* relation, the vast majority of errors can be assigned to the category *NEX-S*. In these cases, a relation between two or more organizations is often expressed in the source sentences, e.g., that company x is a subsidiary of company y, but no statement is made about the act of purchase. For the *marriage* relation, the most frequent error type (with the exception of the *Other* error class) is *IMP-S*, mainly due to sentences stating a divorce, which, according to our annotation guidelines, is not an explicit mention of the *marriage* relation. A final observation that can be made from Table 5 is that approximately 44.0% of the incorrect patterns result from preprocessing pipeline errors.

6. Related Work

In the previous sections, we have motivated the construction of sar-graphs and outlined a method of building them from an alignment of web text with known facts. Taking into account the implemented construction methodology, it may seem that sar-graphs can be regarded as a side-product of pattern discovery for relation extraction. However, sar-graphs are a further development of this, i.e., a novel linguistic knowledge resource on top of the results of pattern discovery.

In comparison to well-known knowledge bases such as YAGO [2, 3], DBpedia [4], Freebase [1], Google's Knowledge Graph [9] or the recent Google

Knowledge Vault [10], sar-graphs are not a database of facts or events, but rather a repository of linguistic representations expressing facts or events. As explained above, the acquisition of sar-graph elements is more related to pattern discovery approaches developed in traditional schema-based IE systems (such as NELL [5, 6], PROSPERA [8] or Web-DARE [18]) than it is to open information extraction (Open IE; e.g., ReVerb [36] and the work by [37, 38, 39, 40, 41, 42]). The principal idea of open IE approaches is that automatically learned patterns are not fixed to a certain schema or ontology. Even though post-processing steps are available for these systems which align the patterns in taxonomies or prepare them otherwise for various downstream tasks [43, 44, 45, 46, 47, 48, 49], sar-graphs are still somewhat closer to schema-driven knowledge graphs, meaning that sar-graphs can be directly applied to free texts for enlarging a structured repository of knowledge. In Sections 7.1 & 7.2, we compare sar-graphs to other systems based on lexico-syntactic patterns.

For the current sar-graphs, we employ a web-driven approach to collect and manage a wide variety of linguistic representations for each semantic relation. The formalism used for intermediate storing of phrases is based on own prior work [23]; we leave a more sophisticated methodology which could return expressions at various granularities [50] for the future. Our work is novel in comparison to traditional pattern-discovery approaches, since we reorganize the collected structures into a coherent, relation-specific linguistic resource, instead of viewing them as sets of independent, statistically-enriched patterns. Sar-graphs merge high-confidence linguistic structures, and combine syntactic information with lexical semantic and probabilistic information. The merged structures can be taken as input for further induction and generalisation.

In comparison to the various construction methods for knowledge bases discussed in [10], the generation of sar-graphs is more in the trend of the Knowledge-Vault approach, since 1) the dependency patterns in sar-graphs are automatically acquired from free texts on the Web, therefore scalable with the growth of the Web; 2) it is driven by a fixed ontology/schema and 3) the patterns are assigned with probabilistic confidence values, thus, adaptable to performance requirements of various applications. Since sar-graphs have been acquired automatically, the construction method is potentially scalable to any new schema.

In the context of knowledge graphs, sar-graphs are one of the first resources to link repositories of facts with linguistic knowledge (i.e., word meanings) in an automatic manner. Each sar-graph corresponds to a linguistic representation of semantic relations from knowledge bases, at the same time the arguments of facts and events are explicitly modeled in sar-graphs. Since a sar-graph is a generic resource, linguistic patterns automatically learned by other systems, e.g., NELL patterns, can also be employed as input. Therefore, sar-graphs can be further developed as a platform for merging and fusing all available extraction patterns from various sources.

Many linguistic repositories, such as WordNet [32], FrameNet [51], and VerbNet [52] already existed before the recent development of large knowledge bases. These linguistic resources have been constructed with the motivation of modeling the semantics of the language at the word or syntactic level, without an explicit link to the real world or applications. Most of them are relatively small scale, due to their manual construction. WordNet captures lexical semantic relations between individual words, such as synonymy, homonymy, and antonymy. FrameNet focuses on fine-grained semantic relations of predicates and their arguments. VerbNet is a lexicon that maps verbs to predefined classes which define the syntactic and semantic preferences of the verb. In contrast to these resources, sar-graphs are data-driven, constructed automatically, and incorporate statistical information about relations and their arguments. Therefore, sar-graphs complement these manually constructed linguistic resources. Furthermore, since word-sense information is integrated into sar-graphs, a linking to other linguistic resources via word senses is straightforward. Thus, sar-garphs can contribute to the linked open data movement (LOD). On the other hand, FrameNet and VerbNet can be employed as useful resources for validating the automatically learned dependency patterns in sar-graphs.

Parallel to the development of large fact databases, there is also increasing research and development in creating similarly sized linguistic resources, e.g., BabelNet, ConceptNet[13] and UBY [14] automatically. Many of them are build on top of existing resources like WordNet, Wiktionary and Wikipedia. As mentioned before, BabelNet is a new development of acquiring large-scale lexical seman-

tics network automatically. It merges Wikipedia concepts including entities with word senses from WordNet. BabelNet can thus be regarded as a knowledge base combining word sense and entity information. In sar-graphs, we employ BabelNet for our word sense disambiguation task since BabelNet is large-scale and its multilinguality is important for extending sar-graphs to other languages. Parallel to BabelNet, [11] create a multilingual word net, called Universal WordNet (UWN), by integrating different word nets, bilingual dictionaries, and information from parallel corpora. This resource is largely an extension of already available word nets and does not provide any explicit linking to a fact knowledge base.

ConceptNet is a semantic network encoding common-sense knowledge and merging information from various sources such as WordNet, Wiktionary, Wikipedia and ReVerb. The nodes are words and phrases expressing concepts and relations among them. Relations are represented by phrases such as "UsedFor" or "HasContext". In comparison to sar-graphs, there is no explicit linguistic knowledge like syntactic or word-sense information assigned to the content elements. The semantic relations among concepts are not fixed to an ontology or schema. However, ConceptNet is a very useful resource which can potentially be utilized to enrich and validate the sar-graphs.

UBY is an approach to combine and align various linguistic resources by employing the so-called ISO-standard Lexical Markup Framework (LMF). They provide a uniform and standardized representation of the individual resources to enable their interoperability. It includes WordNet, Wiktionary, Wikipedia, FrameNet, VerbNet, also the multilingual OmegaWiki. Since UBY is a platform for integration and is therefore open to various applications, sar-graphs can be integrated into UBY as a further resource and can be linked to other linguistic resources via UBY.

Finally, ontology formalizations such as *Lemon* [53] and manual and semi-automatic methods to enrich knowledge bases such as DBpedia and NELL in terms of relation lexicalizations have been presented [54, 55, 56, 57]. The main goal of these methods is to extend the capabilities of the knowledge bases for extracting novel relation instances by using a large set of patterns [58, 59, 60] and to generate NL descriptions of their content [61]. In this paper, we go one step further by not only extending the reference set of extraction patterns associated with the considered relations in terms of lexicalizations and ontological classes but also linking the automatically discovered patterns to word meanings by exploiting word sense disambiguation.

7. Applications and experiments

We believe that sar-graphs, in addition to their role as an anchor in the linked data world and as a repository of relation phrases, are also a very useful resource for a variety of natural-language processing tasks and real-world applications. In particular, sar-graphs are well-suited for (a) the generation of phrases and sentences from database facts and (b) the detection of fact mentions in text.

The first aspect makes sar-graphs a good candidate for the employment in, e.g., *business intelligence* tools aiming to generate natural-language reports for recurring review periods. Because of the range of paraphrases available in sar-graphs, generation could produce stylistic variation as extensively used in reports written by human authors. An application that combines both aspects is *summarization*, where sar-graphs permit to identify fact-heavy parts of a text (i.e., constructions that express all or most arguments of a relation in one sentence) and also allow these parts of a text to be rephrased in a shorter manner.

The most obvious application of sar-graphs, however, is information extraction. As the sar-graphs we have already built contain all the dependency structures we had learned and tested for several relations, we know that the information in a sar-graph can be successfully applied to regular relation extraction, i.e., the detection of fact mentions in sentences.

7.1. Potentials for relation extraction

In order to show that sar-graphs can be successfully applied for information extraction, we conducted a series of experiments [21] for six out of the 25 target relations for which sar-graphs are available. We used the Los Angeles Times/Washington Post (henceforth LTW) part of the *English Gigaword v5* corpus [62]. LTW is comprised of 400k newswire documents from the time 1994–2009. With the help of Stanford NER, we found 4.8M sentences in LTW that mention at least two entities which correspond to argument types defined in the six relations.

We match the individual dependency constructions in the sar-graphs with the dependency parses

Relation	Freq.-overlap	Sem.-graph	Combined
acquisition	50/23/32	62/31/41	89/25/39
marriage	63/32/43	63/33/43	98/32/48
person birth	81/30/43	65/30/41	93/27/42
person death	46/20/28	78/23/35	83/21/33
person parent	88/33/48	56/38/45	95/33/49
sibling rel.	99/22/36	27/25/26	79/23/36
average	71/27/39	59/30/40	90/27/42

Table 6: Statistics about RE performance of sar-graphs for six relations. *Freq.-overlap*/*Sem.-graph* are described in Sections 4.1.1/4.1.2, respectively. Each cell is displaying the precision/recall/f1-measure results in % from the respective f1-best parameter setting.

of the sentences in LTW and evaluate the correctness of the relation mentions detected this way. The different means of sar-graph quality control (see Section 4) are compared wrt. their impact on RE performance. To estimate the *precision* of RE, we manually check a random sample of 1K extracted mentions per relation, and we proceed similarly to get an estimate of the RE *coverage*.[12]

Table 6 presents the results of the experiment. While both types of dependency-structure filtering allow to obtain reasonably good results for the RE task, it is interesting to see that their combination gets an enormous precision boost at almost no cost in terms of recall. Note that the generally low recall values (around 30 %) are quite common among state-of-the-art RE systems. See, for example, [18] for an analysis of the relation mentions not extracted.

In a previous study [21], we compared the extraction performance of a sar-graph predecessor to NELL's [6] lexico-syntactic patterns. NELL is a system designed to learn factual knowledge from an immense, web-like corpus over a long period. We found that when applied to the English Gigaword corpus mentioned above, the amount of relation mentions covered by the patterns of NELL was substantially lower than what the sar-graph-like patterns covered. For a similar selection of relations as in Table 6, the NELL patterns matched approximately 10% of the number of facts extracted by our patterns. Interestingly, the overlap of facts was rather low, i.e., both systems extracted facts the other system was not capable of finding. By stating these results here, we do not aim to provide a comprehensive comparison of the capabilities of the two systems, but rather only want to provide evidence that sar-graphs are indeed a useful member of the diverse landscape of RE systems. We continue our explorative comparison of sar-graphs to other systems in the following section.

Sar-graphs for CALL. Another application for which we applied our sar-graphs is related to the area of *computer-assisted language learning* (CALL). We implemented a prototype [63] for the task of semi-automatic generation of reading-comprehension exercises for second-language learners. A language teacher, who has to prepare such exercises for an upcoming class, is presented with news texts retrieved from the Web, along with candidate multiple-choice questions and answers, relating to certain facts mentioned in the text. The teacher then has to pick the useful question-answer pairs.

Sar-graphs are utilized here both for the fact-finding phase (i.e., for the detection of true-answer candidates), and for the generation of paraphrases for true facts as well as the question asking about the facts. During the evaluation of this setting, we found that for average-length news texts, several correct and potentially useful question-answer pairs are generated for each input text, which led us to conclude that sar-graphs would be of real-world use in such an application setting.

7.2. Similarities with pattern stores

In this section, we present an explorative comparison of the linguistic expressions in the sar-graphs with a typical representative of a RE-pattern store from the literature. We selected the PATTY system [45][13] because it shows very good performance on the information extraction task and it implements a generic approach for the creation of a taxonomy of textual patterns, a goal similar to what we aim at with our sar-graphs.

PATTY implements an open-IE approach to the collection of phrases, which is followed by an alignment phase where a subsumption hierarchy of patterns is created. The PATTY authors provide a disambiguation of their patterns to a knowledge base, from which we select four relations with a considerable semantic overlap with the sar-graph relations

[12] We use our dataset from [21], where we manually verified a large number of Freebase-fact mentions found on a subpart of LTW, i.e., only sentences actually referring to the corresponding target relation are part of this dataset.

[13] `http://resources.mpi-inf.mpg.de/yago-naga/patty/data/patty-dataset.tar.gz`

	PATTY	Sar-graphs	
		sem.-graph	Sec. 5.4/Tab. 5
	3 ∗ (pattern count/lexical diversity/precision)		
employm. ten.	1,246/.05/44	15,656/.26/39	226/.32/70
marriage	3,426/.08/12	48,166/.16/38	451/.26/66
organiz. rel.	838/.06/46	4,344/.45/20	303/.65/37
person parent	2,327/.05/29	32,771/.18/31	387/.24/65

Table 7: Comparison of patterns from PATTY and sar-graphs, for a set of relations present in both resources.

and for which a reasonable number of patterns is available for both systems.

In order to get an estimate of the quality of PATTY's patterns, we took a sample of 200 patterns from each relation and, based on the entities with which a pattern co-occurred in Wikipedia, generated instantiations for all associated entity-type signatures. Three annotators were then asked to judge whether the majority of instantiations per pattern does express the respective target relation. For example, a person `joining` another person does not indicate a mention of the *employment tenure* relation, but a sports person `joining` a sports team does. For the sar-graph patterns, we followed a similar strategy, which resulted in 200 instantiated and string-serialized patterns with entities shown to three human raters. If the annotators could not make sense of a pattern (e.g., because it was overly specific or contained superfluous elements not required for matching a mention of the respective target relation), their guideline was to rate this pattern as wrong.

Table 7 compares PATTY with sar-graph patterns retained after applying the semantic-graph filter[14] (Section 4.1.2) and additionally presents statistics for the sar-graph patterns in the manual-evaluation dataset from the error analysis in Section 5.4. Along with the pattern numbers, we put a lexical-diversity score which states the average amount of distinct, non-function words a pattern has wrt. other patterns in the same set. This score allows to better interpret the absolute pattern numbers, which are subject to the different pattern-representation formalisms.

For the relations of this analysis, sar-graphs provide more linguistic expressions at a higher rate of lexical diversity, i.e., the sar-graph patterns are as least as well suited for RE as the PATTY system wrt. coverage of lexically variations of relation mentions. Furthermore, more than twenty percent of the sar-graph patterns in this analysis link three or more arguments to one another, in contrast to the PATTY patterns which are all binary. Note that while PATTY and sar-graphs were created from different corpora, the size of these corpora is similar, i.e., 2.2 million Web documents in the case of sar-graphs, and 1.8 million New York Times articles/3.8 million Wikipedia articles for PATTY.

Both systems produce patterns at similar levels of precision, where for some relations one system trumps the other. Looking for an explanation of the low *marriage* precision of PATTY, we found that on average the patterns contained less tokens than the ones in the sar-graphs, which makes them more dependent on the disambiguation power of the entity-type signature. *marriage* entity types are generic (two persons), in particular given the source corpus Wikipedia; consequently the precision of PATTY for this relation is lower than for the others. For *person parent*, the situation is similar.

7.3. Knowledge-base population

One of the main goals of relation extraction is the automatic identification of novel relation instances from raw text to populate a knowledge base with new facts. It is well known that existing knowledge bases, such as Freebase [1], while quite sizable, are still far from complete. For example, Dong et. al observe that 71% of the people in Freebase have no known place of birth, and 75% have no known nationality [10]. Large-scale text corpora, such as ClueWeb [64], are assumed to contain thousands of potentially interesting relations and facts, but it is unclear whether these are novel or already covered by the knowledge base. To answer this question, we conducted a small experiment to get a rough estimate of how many new facts we can identify with a sar-graph based relation extraction approach.

We create a test dataset by randomly sampling 12K documents from the FACC1 corpus [65], a preprocessed version of the ClueWeb dataset where entity mentions have been linked to their corresponding Freebase identifiers. From these documents, we select sentences containing at least two Freebase entity mentions. We parse sentences with the MaltParser [66], and extract relation mentions using sar-graph dependency patterns. For each extracted relation mention, we determine its true label(s) by looking up valid relations of the mentions' entities in Freebase. We estimate the percentage of

[14] We set $k = 3$ as this typically results in a good precision/recall trade-off.

Known Predicted	All Predicted	Est. Correct Novel Facts
1171	5552	≈ 1314

Table 8: Estimated number of novel "facts" found by sar-graph-based relation extraction on a ClueWeb dataset sample. The estimate is derived from the number of known and predicted facts, and combined with the approximate precision of 30% of the pattern-based approach.

novel facts by computing the difference of known and predicted relation mentions, multiplied by an estimated precision of 30% of the pattern-based approach. The precision value was determined on a fully-annotated reference dataset in prior work [22]. Table 8 lists the number of predicted, known and estimated correct novel facts. The rough estimate suggests that approximately 50% of the identified relation mentions are not yet included in the knowledge base.

8. Public availability, release

We release our sar-graph data to the public, hoping to support research in the areas of relation extraction, question answering, paraphrase generation, and others. The data is available for download at `http://sargraph.dfki.de`.

Properties of the release. The released dataset contains English sar-graphs for 25 target relations (Table 2) about corporations, award topics, as well as biographic information, many of them linking more than just two arguments. For now, we limit the released data to manually verified dependency structures (Section 5.4), i.e., only structures judged as correct are published. Sar-graphs were created from the dependency structures using the strategy depicted in the pseudocode in Figures 6 & 7. Properties of the currently released sar-graphs are shown in Table 9.[15] The sar-graph creation (see Table 3) involved the utilization of 230k instances of the semantic relations, which resulted in overall two million crawled web documents with supposed mentions of the seeds. From the resulting one million unique sentences we extracted 600k dependency structures, leading to sar-graphs for the target relations that add up to 300k vertices and 1.5 million edges. The curated set of sar-graphs we publish has 8k vertices connected by 20k edges.

	# nodes	# edges
award honor	242	707
award nomination	134	363
country of nationality	182	546
education	154	407
marriage	199	538
person alternate name	199	512
person birth	74	175
person death	114	292
person parent	110	396
person religion	110	230
place lived	114	256
sibling relationship	67	166
acquisition	121	253
business operation	143	297
company end	163	330
company product relationship	175	375
employment tenure	103	289
foundation	131	398
headquarters	121	293
organization alternate name	67	166
organization leadership	131	442
organization membership	150	405
organization relationship	94	257
organization type	199	348
sponsorship	271	604
average	143	362
sum	3,568	9,045

Table 9: Properties of released sar-graphs.

Along with the sar-graphs, we release the individual dependency constructions used to construct the published set of sar-graphs, as well as the corresponding word-sense information and the assessments from the automatic quality control.

As described in Section 6, we see our sar-graphs as a natural extension to the resources already established as part of the Semantic Web, complementing the existing information and linking to the available resources. We link the data to lexical-semantic resources via the synset information attached to the vertices and also to factual knowledge bases via the covered semantic relations. By additionally providing the sar-graphs in the *Lemon* format[16], we hope to facilitate further research on the interesting area of intersecting factual knowledge with linguistic information.

Java-based API. Accompanying the data, we provide a Java-based API which simplifies the loading, processing, and storing of sar-graphs, and also allows to visualize the individual dependency structures we exploited for the generation of sar-graphs.

[15] The sar-graph data available on our website constitutes a superset of the structures judged as correct in Section 5.4. The additional structures were manually verified in a further annotation effort with the PatternJudge tool (Section 4.2.3).

[16] `http://www.lemon-model.net/`

One particularly helpful feature of the API is the concept of *materialized views*, already broached in Section 3. The basic idea is that with different tasks and goals, varying aspects of a sar-graph become relevant. For example, an application having a lexical-semantic point-of-view on data might be interested in possible dependency connections between sets of synonymous words or word classes (hence a very broad view on the sar-graphs is needed), while another application might want to partition linguistic information based on the specific facts the knowledge was derived from, circumventing even the abstraction of entities to entity types. Hence, the sar-graph data should be presented in the respective most informative way to an application. The API from our recent releases provides this possibility.[17]

Future plans for the resource. For the coming releases, we plan (1) to publish the non-curated part of the sar-graph data, which, for example, proved to be useful for tasks like relation extraction (Section 7), (2) to provide more detailed information about the source of linguistic expressions (i.e., expand the public data with source sentences and seed facts, (3) to extend the sar-graph approach to more semantic relations and domains.

9. Conclusion

In this article, we present a new linguistic resource called *sar-graph*, which aggregates knowledge about the means a language provides for expressing a given semantic target relation. We describe a general approach for automatically accumulating such linguistic knowledge, and for merging it into a single connected graph. Furthermore, we discuss different ways of assessing the relevancy of expressions and phrases with respect to the target relation, and outline several graph merging strategies. We show the validity of our approach by implementing it on top of a large English web corpus. In our experiments, we created and evaluated sar-graphs for 25 relations from the domains *Award*, *Business* and *People*. A curated subset of these graphs is publicly available at `sargraph.dfki.de`.

[17] Although we are not providing all aspects of the potentially interesting information at the moment. For example, we are not yet publishing source sentences along with the derived dependency structures, due to licensing issues.

We believe linguistic resources like sar-graphs should be created in a bottom-up fashion, thereby being empirically grounded on the actual ways people communicate about semantic relations in different languages. Even though we admit that a fully automatic approach is hardly feasible due to shortcomings of the unsupervised quality assessments, we think that a fully curated approach, i.e., language-independent engineering of ontologies, would constitute a throwback to a research paradigm in which knowledge engineering precedes any attempt of language understanding.

From experience we have learned that there could be numerous different ontologies just for the thematic area marriage. Lawyers, event managers, relationship counselors, vital statisticians may all come up with completely different ways to select and structure the respectively relevant knowledge pieces. How could we decide on the best ontology for, e.g., the task of relation extraction? Would any of such intellectually created ontologies contain a relation for *exchanging the vows* and one for *tying the knot*? How would *the vows* and *the knot* be represented? The great advantage of an empirical bottom-up approach is that it is guided by the actual ways people use to refer to a relation (or event, process, etc.), and that one is not pressured to make such a-priori ontology-level decisions.

Another important choice we make is the association of graphs to specific languages. A Greek report on a wedding may refer to *wedding crowns* for bride and groom, while in an English sar-graph for the *marriage* relation, such crowns would not show up. In a Greek wedding the *betrothal* can be a part of the entire ceremony, in other cultures it must have taken place a certain period before the wedding. In some cultures, *exchanging the rings* means *getting married* in others there is no such concept.

We are convinced that we need the interaction of two strategies to build up a growing stock of structured knowledge in the spirit of a semantic web. One strategy starts from structuring growing portions of textual knowledge sources (such as Wikipedia) and extends this by already structured data (such as linked open data). Another strategy uses and extends the resulting repositories of structured knowledge by extracting from all sorts of texts much more facts, especially contingent ones. The novel type of resource we propose will on the one hand facilitate the latter process and on the other hand maintain the link of the accumulated

domain-sorted linguistic knowledge with structured resources from the Semantic Web.

Acknowledgements

This research was supported by the German Federal Ministry of Education and Research (BMBF) through the projects Deependance (contract 01IW11003), ALL SIDES (contract 01IW14002) and BBDC (contract 01IS14013E), as well as by the ERC Starting Grant MultiJEDI No. 259234, and a Google Focused Research Award granted in July 2013.

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proc. of SIGMOD, ACM, 2008. doi:10.1145/1376616.1376746.

[2] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: A core of semantic knowledge, in: Proc. of WWW, ACM, 2007. doi:10.1145/1242572.1242667.

[3] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: A large ontology from Wikipedia and WordNet, Web Semantics: Science, Services and Agents on the World Wide Web 6 (3) (2008) 203–217. doi:10.1016/j.websem.2008.06.001.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web Journal 6 (2) (2015) 167–195. doi:10.3233/SW-140134.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, T. M. Mitchell, Toward an architecture for never-ending language learning, in: Proc. of AAAI, 2010.

[6] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in: Proc. of AAAI, 2015.

[7] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[8] N. Nakashole, M. Theobald, G. Weikum, Scalable knowledge harvesting with high precision and high recall, in: Proc. of WSDM, ACM, 2011. doi:10.1145/1935826.1935869.

[9] A. Singhal, Introducing the Knowledge Graph: things, not strings, Google Official Blog, http://goo.gl/KCwaV6 (May 2012).

[10] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, W. Zhang, Knowledge vault: A web-scale approach to probabilistic knowledge fusion, in: Proc. of SIGKDD, ACM, 2014. doi:10.1145/2623330.2623623.

[11] G. de Melo, G. Weikum, Towards a universal wordnet by learning from combined evidence, in: Proc. of CIKM, ACM, 2009. doi:10.1145/1645953.1646020.

[12] R. Navigli, S. P. Ponzetto, BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artificial Intelligence 193 (2012) 217–250. doi:10.1016/j.artint.2012.07.001.

[13] R. Speer, C. Havasi, ConceptNet 5: A large semantic network for relational knowledge, in: The People's Web Meets NLP, Springer, 2013, pp. 161–176.

[14] I. Gurevych, J. Eckle-Kohler, S. Hartmann, M. Matuschek, C. M. Meyer, C. Wirth, Uby: A large-scale unified lexical-semantic resource based on LMF, in: Proc. of EACL, ACL, 2012.

[15] Wikimedia Foundation, Wiktionary. URL http://www.wiktionary.org/

[16] H. Uszkoreit, F. Xu, From Strings to Things SAR-Graphs: A New Type of Resource for Connecting Knowledge and Language, in: Proc. of NLP & DBpedia Workshop co-located with ISWC, CEUR-WS.org, 2013.

[17] M. Mintz, S. Bills, R. Snow, D. Jurafsky, Distant supervision for relation extraction without labeled data, in: Proc. of ACL-IJCNLP, ACL, 2009.

[18] S. Krause, H. Li, H. Uszkoreit, F. Xu, Large-scale learning of relation-extraction rules with distant supervision from the web, in: Proc. of ISWC, Springer, 2012.

[19] L. Jean-Louis, R. Besanon, O. Ferret, A. Durand, Using distant supervision for extracting relations on a large scale, in: Knowledge Discovery, Knowledge Engineering and Knowledge Management, no. 348 in Communications in Computer and Information Science, Springer, 2013, pp. 141–155. doi:10.1007/978-3-642-37186-8_9.

[20] B. Min, R. Grishman, L. Wan, C. Wang, D. Gondek, Distant supervision for relation extraction with an incomplete knowledge base, in: Proc. of NAACL-HLT, ACL, 2013.

[21] A. Moro, H. Li, S. Krause, F. Xu, R. Navigli, H. Uszkoreit, Semantic rule filtering for web-scale relation extraction, in: Proc. of ISWC, Springer, 2013.

[22] H. Li, S. Krause, F. Xu, A. Moro, H. Uszkoreit, R. Navigli, Improvement of n-ary relation extraction by adding lexical semantics to distant-supervision rule learning, in: Proc. of ICAART, SciTePress, 2015.

[23] F. Xu, H. Uszkoreit, H. Li, A seed-driven bottom-up machine learning framework for extracting relations of various complexity, in: Proc. of ACL, ACL, 2007.

[24] F. Xu, Bootstrapping relation extraction from semantic seeds, Ph.D. thesis, Saarland University (2007).

[25] M.-C. de Marneffe, C. D. Manning, Stanford Dependencies Manual (2008). URL http://nlp.stanford.edu/software/dependencies_manual.pdf

[26] M. Stevenson, Fact distribution in information extraction, Language Resources and Evaluation 40 (2) (2006) 183–201. doi:10.1007/s10579-006-9014-4.

[27] K. Swampillai, M. Stevenson, Inter-sentential relations in information extraction corpora, in: Proc. of LREC, ELRA, 2010.

[28] R. Grishman, Information Extraction: Capabilities and Challenges, Tech. rep., NYU Dept. CS (2012).

[29] R. Navigli, Word sense disambiguation: A survey, ACM Comput. Surv. 41 (2) (2009) 10:1–10:69. doi:10.1145/1459352.1459355.

[30] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, R. M. Weischedel, The Automatic Content Extraction (ACE) program - tasks, data, and evaluation, in: Proc. of LREC, 2004.

[31] J. R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by gibbs sampling, in: Proc. of ACL, ACL, 2005. doi:10.3115/1219840.1219885.

[32] C. Fellbaum (Ed.), WordNet: an electronic lexical database, Language, Speech, and Communication, MIT Press, 1998.

[33] C. Chiarcos, S. Nordhoff, S. Hellmann (Eds.), Linked Data in Linguistics. Representing and Connecting Language Data and Language Metadata, Springer, 2012. doi:10.1007/978-3-642-28249-2.

[34] R. Navigli, BabelNet goes to the (multilingual) semantic web, in: Proc. of Workshop on the Multilingual Semantic Web, 2012.

[35] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: A unified approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.

[36] A. Fader, S. Soderland, O. Etzioni, Identifying relations for open information extraction, in: Proc. of EMNLP, ACL, 2011.

[37] O. Etzioni, A. Fader, J. Christensen, S. Soderland, Mausam, Open information extraction: The second generation, in: Proc. of IJCAI, IJCAI/AAAI, 2011.

[38] Mausam, M. Schmitz, S. Soderland, R. Bart, O. Etzioni, Open language learning for information extraction, in: Proc. of EMNLP-CoNLL, ACL, 2012.

[39] E. Alfonseca, D. Pighin, G. Garrido, HEADY: News headline abstraction through event pattern clustering, in: Proc. of ACL, ACL, 2013.

[40] Y. Xu, M.-Y. Kim, K. Quinn, R. Goebel, D. Barbosa, Open information extraction with tree kernels, in: Proc. of NAACL-HLT, ACL, 2013.

[41] L. Del Corro, R. Gemulla, ClausIE: Clause-based open information extraction, in: Proc. of WWW, International World Wide Web Conferences Steering Committee, 2013.

[42] D. Pighin, M. Cornolti, E. Alfonseca, K. Filippova, Modelling events through memory-based, open-ie patterns for abstractive summarization, in: Proc. of ACL, ACL, 2014.

[43] A. Yates, O. Etzioni, Unsupervised resolution of objects and relations on the web, in: Proc. of HLT-NAACL, ACL, 2007.

[44] A. Moro, R. Navigli, WiSeNet: Building a Wikipedia-based semantic network with ontologized relations, in: Proc. of CIKM, ACM, 2012. doi:10.1145/2396761.2398495.

[45] N. Nakashole, G. Weikum, F. Suchanek, PATTY: A taxonomy of relational patterns with semantic types, in: Proc. of EMNLP-CoNLL, ACL, 2012.

[46] C. Zhang, D. S. Weld, Harvesting parallel news streams to generate paraphrases of event relations, in: Proc. of EMNLP, ACL, 2013.

[47] O. Levy, I. Dagan, J. Goldberger, Focused entailment graphs for open IE propositions, in: Proc. of CoNLL, ACL, 2014.

[48] A. Grycner, G. Weikum, HARPY: Hypernyms and alignment of relational paraphrases, in: Proc. of COLING, DCU & ACL, 2014.

[49] S. Krause, E. Alfonseca, K. Filippova, D. Pighin, Idest: Learning a distributed representation for event patterns, in: Proc. of NAACL-HLT, 2015.

[50] K. Beedkar, R. Gemulla, LASH: large-scale sequence mining with hierarchies, in: Proc. of SIGMOD, ACM, 2015. doi:10.1145/2723372.2723724.

[51] C. F. Baker, C. J. Fillmore, J. B. Lowe, The Berkeley FrameNet project, in: Proc. of ACL-COLING, ACL, 1998.

[52] K. K. Schuler, Verbnet: A broad-coverage, comprehensive verb lexicon, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA (2005).

[53] J. McCrae, D. Spohr, P. Cimiano, Linking lexical resources and ontologies on the semantic web with Lemon, in: Proc. of ESWC, Springer, 2011. doi:10.1007/978-3-642-21034-1_17.

[54] S. Walter, C. Unger, P. Cimiano, ATOLL - a framework for the automatic induction of ontology lexica, Data & Knowledge Engineering 94, Part B (2014) 148 – 162, S.I. following NLDB 2013. doi:http://dx.doi.org/10.1016/j.datak.2014.09.003.

[55] S. Walter, C. Unger, P. Cimiano, M-ATOLL: A framework for the lexicalization of ontologies in multiple languages, in: Proc. of ISWC, Vol. 8796, Springer, 2014. doi:10.1007/978-3-319-11964-9_30.

[56] C. Unger, J. McCrae, S. Walter, S. Winter, P. Cimiano, A lemon lexicon for DBpedia, in: Proc. of NLP & DBpedia Workshop co-located with ISWC, Vol. 1064, CEUR-WS.org, 2013.

[57] N. Lao, A. Subramanya, F. Pereira, W. W. Cohen, Reading the web with learned syntactic-semantic inference rules, in: Proc. of EMNLP-CoNLL, ACL, 2012.

[58] P. Cimiano, C. Unger, J. McCrae, Ontology-Based Interpretation of Natural Language, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2014. doi:10.2200/S00561ED1V01Y201401HLT024.

[59] M. Gardner, P. P. Talukdar, J. Krishnamurthy, T. M. Mitchell, Incorporating vector space similarity in random walk inference over knowledge bases, in: Proc. of EMNLP, ACL, 2014.

[60] M. Gardner, P. P. Talukdar, B. Kisiel, T. M. Mitchell, Improving learning and inference in a large knowledge-base using latent syntactic cues, in: Proc. of EMNLP, ACL, 2013.

[61] P. Cimiano, J. Lüker, D. Nagel, C. Unger, Exploiting ontology lexica for generating natural language texts from rdf data, in: Proc. of European Workshop on Natural Language Generation, ACL, 2013.

[62] R. Parker, English Gigaword fifth edition, linguistic Data Consortium, Philadelphia (2011).

[63] R. Ai, S. Krause, W. Kasper, F. Xu, H. Uszkoreit, Semi-automatic generation of multiple-choice tests from mentions of semantic relations, in: Proc. of Workshop on Natural Language Processing Techniques for Educational Applications, ACL, 2015.

[64] J. Callan, M. Hoy, C. Yoo, L. Zhao, The ClueWeb09 dataset.
URL http://lemurproject.org/clueweb09

[65] E. Gabrilovich, M. Ringgaard, A. Subramanya, FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

[66] J. Nivre, J. Hall, J. Nilsson, Memory-Based Dependency Parsing, in: Proc. of CoNLL, ACL, 2004.