# Semantic Technologies for Querying Linguistic Annotations: An Experiment Focusing on Graph-Structured Data

**Milen Kouylekov♣, Stephan Oepen♣♡**
♣ University of Oslo, Department of Informatics
♡ Potsdam University, Department of Linguistics
{ milen | oe }@ifi.uio.no

**Abstract**

With growing interest in the creation and search of linguistic annotations that form general graphs (in contrast to formally simpler, rooted trees), there also is an increased need for infrastructures that support the exploration of such representations, for example logical-form meaning representations or semantic dependency graphs. In this work, we heavily lean on semantic technologies and in particular the data model of the Resource Description Framework (RDF) to represent, store, and efficiently query very large collections of text annotated with graph-structured representations of sentence meaning.

**Keywords:** Semantic Dependency Graphs, Treebank Search, Resource Description Framework

## 1. Motivation

Much work in the creation and use of language resources has focused on *tree*-shaped data structures,[1] as are commonly used for the encoding of, for example, syntactic or discourse annotations. Conversely, there has been less focus on supporting general *graphs* until recently, but there is growing interest in graph-structured representations, for example to annotate and process semantic analyses. Even if syntax arguably can be limited to tree structures, this is obviously not the case in semantics, where a node will often be the argument of multiple predicates (i.e. have more than one incoming arc), and it will often be desirable to leave some nodes unattached (with no incoming arcs), for semantically vacuous classes as, for example, particles, complementizers, or relative pronouns.

Large, semantically annotated resources remain scarce, but several ongoing initiatives promise to deliver graph-structured linguistic annotations in large volumes (Flickinger et al., 2010; Rosén et al., 2012). Of these, we will use data from the so-called WikiWoods Treecache of Flickinger et al. (2010) as our motivating example, in part because in recent work Flickinger et al. (2012) have complemented this resource with a collection dubbed DeepBank, a fresh annotation of the venerable Wall Street Journal (WSJ) text from the Penn Treebank (PTB; Marcus et al., 1993), offering the same layers of semantic annotations as WikiWoods. With some 700,000 tokens of (manually annotated) text in DeepBank and some 48 million (automatically parsed) tokens in WikiWoods, general-purpose search over linguistic annotations at this scale is an R&D challenge in its own right, even more so seeing as both resources make central use of general graph-structured data. Figure 1 presents an example of a semantic dependency graph in DeepBank; here, we observe both node re-entrancies and partial connectivity, as *technique*, for example, is the argument of the determiner (as the quantificational locus), the modifier *similar*, and the predicate *apply*. Conversely, the predicative copula, infinitival *to*, and the preposition marking the deep object of *apply* have no semantic contribution of their own. We discuss this example further in §2. below.

In this work, we investigate the use of *semantic technologies*, and in particular of the data model of the Resource Description Framework (RDF), to represent, store, and search (very) large-scale linguistic graphs. We demonstrate how different types of semantic dependency graphs can be encoded in RDF and discuss some of the design choices involved; we then summarize a practical experiment, using increasing volumes of data from DeepBank and WikiWoods and benchmarking two open-source triple stores. This is in contrast to ongoing work by Meurer (2012), who (to query the annotations of Rosén et al., 2012) approaches the same abstract challenge through a generalization of the TIGERSearch query language (König & Lezius, 2000) and a custom-built indexing and search infrastructure. Ultimately, it will of course be desirable to compare the two candidate solutions with regards to, for example, efficiency, scalability, flexibility, and implementation cost.

## 2. Example: Semantic Dependency Graphs

Semantic annotations in the DeepBank Treebank and the WikiWoods Treecache are couched in the framework of Minimal Recursion Semantics (MRS; Copestake et al., 2005), a designer language for underspecified logical-form meaning representation. In ongoing work, we investigate the embedding of complete MRS graphs into RDF, but in the following we will focus on two 'reductions' of MRS for reasons of simplicity and space.

Oepen and Lønning (2006) propose a lossy (i.e. non-reversible) conversion from MRS into a variable-free dependency graph, duped Elementary Dependency Structures (EDS). Figure 2 shows the EDS analysis for our running example: graph nodes (one per line) correspond to elementary predications from the underlying logical form and are connected by directed arcs, listed in parentheses, which are labeled with MRS argument roles: ARG1, ARG2, etc. (where BV denotes what is the bound variable of a quantifier in the

[1]Formally, trees are a restricted form of graphs, where every node is reachable from a distinguished root node by exactly one directed path.

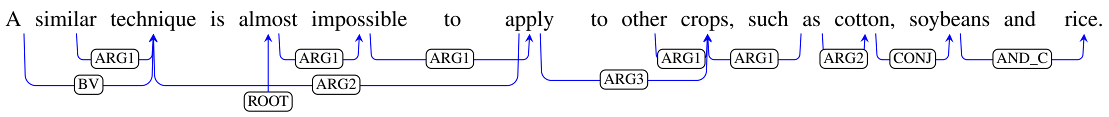

A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.
ARG1
BV
ARG1
ROOT
ARG2
ARG1
ARG3
ARG1
ARG1
ARG2
CONJ
AND_C

Figure 1: Example bi-lexical semantic dependencies (DM; taken from DeepBank).

full MRS).[2] Note that, while EDS already brings us relatively close to 'classic' depedency representations, there are graph nodes that do not correspond to individual words from our running example, for example the underspecified quantifiers for the bare noun phrases (udef_q) and the binary conjunction implicit_conj that ties together *cotton* with *soybeans and rice*; furthermore, just as we saw in Figure 1 already, some words of the underlying sentence are semantically empty; and the EDS does not form a tree, as there are re-entrancies in the graph (e.g. at node $x_6$).

Our original bi-lexical semantic dependency graph from Figure 1 above in fact resulted from a further simplification step—'projecting' predicate–argument information from the EDS onto individual tokens of the underlying utterance—defined by Ivanova et al. (2012) as DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM). Comparing the two graphs, we observe that some argument relations pertaining to covert quantifiers cannot be represented in the pure bi-lexical form, and the coordinate structure has been 'flattened' in the spirit of Mel'čuk (1988). Our discussion of RDF embedding in the following section is applicable to both types of dependency graphs (and others, similar in nature), but we will focus on the EDS format, because it is somewhat richer and more general.

{ $e_{12}$
$\_1$:_a_q(BV $x_6$)
$e_9$:_similar_a_to(ARG1 $x_6$)
$x_6$:_technique_n_1
$e_{12}$:_almost_a_1(ARG1 $e_3$)
$e_3$:_impossible_a_for(ARG1 $e_{18}$)
$e_{18}$:_apply_v_to(ARG2 $x_6$, ARG3 $x_{19}$)
$\_2$:udef_q(BV $x_{19}$)
$e_{25}$:_other_a_1(ARG1 $x_{19}$)
$x_{19}$:_crop_n_1
$e_{26}$:_such+as_p(ARG1 $x_{19}$, ARG2 $x_{27}$)
$\_3$:udef_q(BV $x_{27}$)
$\_4$:udef_q(BV $x_{33}$)
$x_{33}$:_cotton_n_1
$\_5$:udef_q(BV $i_{38}$)
$x_{27}$:implicit_conj(L-INDEX $x_{33}$, R-INDEX $i_{38}$)
$\_6$:udef_q(BV $x_{43}$)
$x_{43}$:_soybeans/nns_u_unknown
$i_{38}$:_and_c(L-INDEX $x_{43}$, R-INDEX $x_{47}$)
$\_7$:udef_q(BV $x_{47}$)
$x_{47}$:_rice_n_1
}

Figure 2: Elementary Dependency Structure (EDS).

## 3. Background: RDF and SPARQL

The RDF data model is based upon the idea of making statements about resources in the form of subject–predicate–object triples. The subject denotes the resource, and the predicate denotes traits or aspects of the resource, thus expressing a relationship between the subject and the object. A database that can store such expression and evaluate queries to them is called a triple store.

The simple data model of RDF and its ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications unrelated to Semantic Web activity. A collection of RDF statements intrinsically represents a labeled, directed multi-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models.

SPARQL is an RDF query language, that is, a formal language to search triple stores, allowing one to retrieve and manipulate RDF data. It is fully standardized and considered one of the key technologies of the Semantic Web. A SPARQL query can consist of triple patterns, conjunctions, disjunctions, and optional patterns. Figure 3 shows an example, searching for an EDS configuration where *apply* takes *crops* as its ARG3.

```
PREFIX ltg:<http://www.uio.no/ltg#>
select ?id ?text where {
  ?x ltg:label "_apply_v_to" .
  ?x ltg:in ?id .
  ?y ltg:label "_crop_n_1" .
  ?x ltg:ARG3 ?y .
  ?id ltg:text ?text
}
```

Figure 3: Example SPARQL query.

In this query each line represents a triple, and elements starting with question mark are variables. The query processor searches for sets of triples that match the patterns expressed in the query, binding variables in the query to the corresponding parts of each triple. The result of the SPARQL query is a set of instantiations of the variables `?id` and `?text`, the identifiers and surface strings for all matching EDS graphs.

The use of RDF to encode linguistic data is not new. Arguably the most popular type of data stored in RDF are lexica, instantiating models like Lemon[3] or the W3C Simple Knowledge Organization System (SKOS)[4]. Chiarcos (2012) discuss the encoding of linguistically annotated corpora in RDF, suggesting interoperability and query flexibility as key advantages of this approach. Another RDF framework for the representation of linguistic annotations is presented by Rubiera et al. (2012), demonstrating that

[2]In the textual rendering of our EDS in Figure 2, nodes are prefixed with unique identifiers, which serve to denote node reentracy and the targets of outgoing dependency arcs.

[3]http://lemon-model.net/.

[4]http://www.w3.org/2006/07/SWD/SKOS/xl/20080414.

```
@prefix eds:
  <http://www.delph-in.net/rdf/eds#>.

eds:predicate
    rdf:type owl:DatatypeProperty;
    rdfs:domain eds:Node ;
    rdfs:range xsd:string .

eds:carg
    rdf:type owl:DatatypeProperty ;
    rdfs:domain eds:Node ;
    rdfs:range xsd:string .

eds:Role
    rdf:type owl:ObjectProperty;
    rdfs:domain eds:Node;
    rdfs:range eds:Node.

eds:ARG1
    rdf:type owl:ObjectProperty;
    rdfs:subPropertyOf eds:Role.

...
```

Figure 4: Excerpt from EDS ontology (in Turtle syntax).

RDF is a suitable data model to capture multiple annotations on the same text segment, and to integrate multiple layers of annotations. However, both studies are focused on flat or tree-structured linguistic data.

## 4. Semantic Dependency Graphs in RDF

To store the MRS, EDS, and DM graphs from DeepBank and WikiWoods in RDF, we created a small ontology to represent the semantics of these data types. In a nutshell, the ontology provides a generic representation of a directed graph with (potentially complex) node and edge labels. The full MRS ontology (not discussed in detail here) distinguishes different types of nodes, corresponding to full predications vs. individual logical variables vs. hierarchically organized sub-properties of variables. The EDS and DM ontologies, on the other hand, essentially make do with a single type of graph node (corresponding to the vertices in the examples of Figures 1 and 2). The dependencies proper, i.e. labeled arcs of the graph, are encoded as object properties. A fragment of the ontology created for the EDS format is shown in Figure 4.

On the node class, the predicate property holds the node label in EDS.[5] In the DM ontology, there are three label-like node properties: form, lemma, and pos (of which the latter record additional lexical information, not shown in Figure 1). We define separate object properties for each argument label, ARG1, ..., ARG4, BV, L-INDEX, R-INDEX, etc. To demonstrate this encoding, Figure 5 shows parts of the RDF corresponding to the first EDS node in Figure 2. In general, we have found the RDF and OWL modeling facilities a good match for the linguistic properties of our three formats for meaning representation, including the use of multiple inheritance for underspecification in MRS.

```
<_1> eds:predicate "_a_q"^^xsd:string;
    rdfs:type eds:Node;
    eds:BV <x6>
```

Figure 5: Example RDF fragment of one EDS node.

## 5. A First Experiment: Sesame and Jena

As a first calibration experiment, we converted the EDS graphs from DeepBank and WikiWoods into RDF, yielding around 12 million and 4.3 billion triples, respectively (for the semantic dependencies of about 37 thousand and 48 million sentences in the two resources, respectively). Judging from experience reports with contemporary triple stores, these are large but not unrealistic counts. We have experimented with two widely used and freely available triple stores and evaluated their potential to store and query the triples generated in the two corpora we have converted in RDF.

One highly flexible triple storage framework is Sesame,[6] used in diverse industries such as pharmaceutical, healthcare, and manufacturing for integrating disparate data sources. It provides a generic application programming interface that allows experimentation with different back-end implementations of a triple store at low adaptation costs. Sesame comes with two built-in triple stores, called MemoryStore and NativeStore—keeping data and indexes either in-core or on-disk—with different space–time tradeoffs. Storing all semantic graphs from DeepBank in either MemoryStore or NativeStore was straightforward, requiring about 3 gbyte of memory for in-core storage. Both stores responded to complex SPARQL queries in fractions of a second.

The second triple store we have investigated is Apache Jena.[7] It too is an open-source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. In many respects, Jena is quite similar to Sesame; though, unlike Sesame, Jena also provides support for the Web Ontology Language (OWL), including various internal reasoners. Like Sesame, Jena offers two implementations of a triple store, viz. (a) SDB, a store based on a relational database; and (b) TDB, an on-disk implementation of a triple store. We measured the performance of these RDF stores in two types of experiments.

**Indexing** The first experiment involved the indexing process. We have measured the indexing time and the disk space occupied by each of the implementations of tripple store in the Sesame and Jena frameworks. For the memory store we measured the (virtual) memory footprint of the

[5]Predicate names in MRS (and thus EDS) by convention have internal structure, such that there is a design choice here; one could tease apart separate components as distinct properties, to make the structure explicit at the cost of increased triple counts, or simply rely on SPARQL regular expressions over our current predicate strings.

[6]http://www.openrdf.org/.

[7]https://jena.apache.org/

|  | Sesame |  | Jena |  |
|---|---|---|---|---|
|  | Memory | Native | SDB | TDB |
| **Time** | 13 sec | 25 sec | 35 sec | 20 sec |
| **Space** | 1.1GB | 800MB | 710MB | 720MB |

Table 1: Indexing time and space for DeepBank.

|  | Sesame |  | Jena |  |
|---|---|---|---|---|
|  | Memory | Native | SDB | TDB |
| **Time (ms)** | 52 | 88 | 122 | 60 |

Table 2: Average query repsonse time for DeepBank.

indexing process. The results obtained by this experiment, using the 'smaller' DeeepBank corpus, are show in Table 1.

As is evident from the table, for this data set indexing times are broadly comparable, and fast enough to store all of the RDF triples from DeepBank in less than a minute for the different implementations. Next we have proceeded to index the RDF triples derived from the WikiWoods corpora. However, our attempts at injecting increasing portions into Sesame turned up technical difficulties from about 100 million triples upwards, well below the total number of triples in the resource. We tried to index the triples in portions of 100 milion triples per file. The indexing time of the first file took than a day to finish. These observations were not encoraging enough to consider Sesame a good option for bigger collections of semantic graphs. The indexing speed of the Jena SDB store was not encoraging either, at about 18 hours for the first 100 million triples. On the other hand, the Jena TDB store succeded in indexing all the triples derived from the Wikiwods corpus in 42 hours. It occupied 750GB of disk space.[8]

**Search** In this second experiment, we seek to compar the response times for each of the triple store implementations in Sesame and Jena. The experiment consisted in evaluating ten hand-constructed queries with different complexity, measuring the average speed with which each implementaion returns the expected result. As we were not able to index the WikiWoods corpus in Sesame or Jena SDB, we compared only the indexes generated for the DeepBank data. The results obtained in this experiment are shown in Table 2.

Although a comparison in terms of artificially constructed queries may not be fully indicative of end user experience, assuming there would potentially be a larger variability in queries posed by actual users, our results for DeepBank suggest that all four RDF triple stores provide a fully viable solution to search over a substantive collection of semantic graphs. This even more so, as in our ongoing work on building on-line search services (see §6. below) we can at times obtain great reductions in query times by combining the RDF triple store with a family of simple textual indexes (hash tables, essentially), to perform query optimization, for example ording the sub-components of a SPARQL query according to 'specificity', i.e. the number of matches found for a token string or wildcard. In this benchmark, Jena TDB (using on-disk storage, where in our setup we keep all indexes on a RAID of solid-state drives) performs nearly competitive to the in-memory triple store of Sesame. Combined with its greatly superior scalability to the much larger WikiWoods corpus, we conclude that Jena TDB on balance is the best choice for our needs, among the four solutions benchmarked to date.

## 6. For Non-Experts: Query by Example

Using SPARQL to search for potentially complex semantic relations in an RDF triple store requires an understanding of both (a) the target representations and (b) the internal structures of the embedding RDF model. We believe that RDF technologies score high on expressivity as well as scalability, but obviously a query like the one in Figure 3 will be hard to pose for non-experts. To lower the technological barrier to entry, we are currently experimenting with a query-by-example front-end to the Jena query processor, essentially defining a generalization of the textual EDS representation to express search patterns over either the MRS, EDS, or DM formats.. For example, the `where` condition of our above SPARQL query could be characterized as follows:

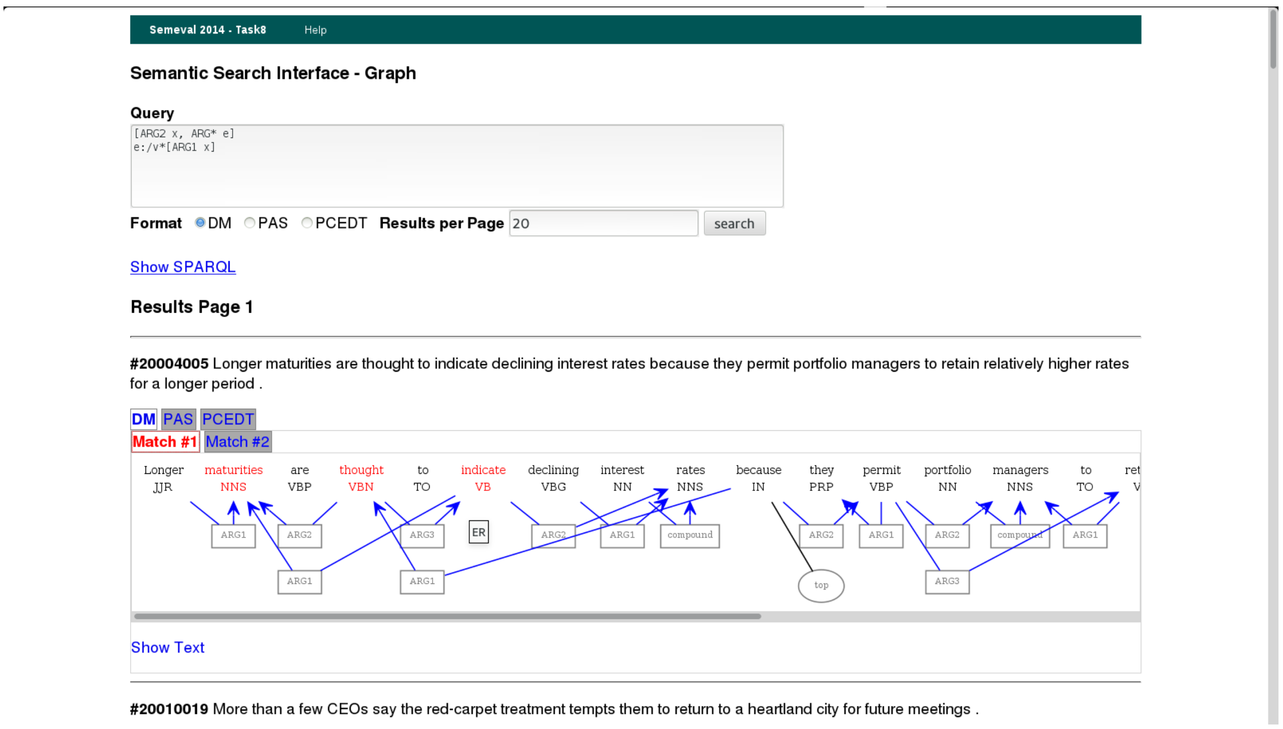
{_apply_v_to(ARG3 *x*), *x*:_crop_n_1}

With suitable user interface support, a pattern like this can be expanded into a full-blown SPARQL query by a procedure similar to the mapping from EDS to RDF triples. In a sense, this technique can be interpreted as a templatic meta-language for (parts of) SPARQL queries, and we anticipate using a similar approach for the incorporation of additional custom (linguistic) query operators (as defined natively in the infrastructure of Meurer, 2012, for example).

As a first 'production' instance of this paradigm, we have built a public search interface to the semantic dependency graphs used in Task 8 of the 2014 Workshop for Semantic Evaluation (SemEval).[9] This data combines the DeepBank DM graphs with another two formats for bi-lexical semantic dependency graphs (called PAS and PCEDT), over the same set of sentences. With minimal adaptations to the DM ontology, and defining a simple and compact query language along the lines above, we have been able to import this data into our infrastructure and have the impression that the resulting on-line search service is perceived well. Figure 6 shows a sample session, searching for object equi verbs, i.e. a configuration of two DM nodes (described one per line in the query interface) that both share an argument: the variable 'x' in this example identifies the shared argument, which is the ARG2 (deep object) of one DM node, and ARG1 (deep subject) of another. In addition to the shared argument between the two nodes, the second DM node in the query (identified by the variable 'e') is further required to

[8]There exist triple stores like AllegroGraph (`http://www.franz.com/agraph/allegrograph/`) with success stories of triple sets counting in the trillions, albeit on fairly high-end hardware. Part of our objective is to see how far current RDF technologies can be pushed on modern commodity equipment.

[9]`http://alt.qcri.org/semeval2014/task8/` provides more details on this task and on the data involved, as well as a link to the on-line search interface built from the techniques discussed here.



Semantic Search Interface - Graph

Query

```
[ARG2 x, ARG* e]
e:/v*[ARG1 x]
```

Format [✓ DM] PAS PCEDT Results per Page 20 search

Show SPARQL

Results Page 1

**#20004005** Longer maturities are thought to indicate declining interest rates because they permit portfolio managers to retain relatively higher rates for a longer period .

**#20010019** More than a few CEOs say the red-carpet treatment tempts them to return to a heartland city for future meetings .

Figure 6: DM search interface and result visualization, as used in conjunction with Task 8 of SemEval 2014.

have a verbal pos property (indicated by the '/' search operator in conjunction with '*' wildcarding), and be any type of argument to the first of the two nodes.

## 7. Ongoing & Future Work

This preliminary report on our ongoing efforts to adapt stock RDF technologies to the storage and retrieval of vast volumes of graph-structured linguistic annotations demonstrates the general feasibility of semantic technologies for storing and querying the substantial number of semantic graph in the DeepBank Treebank. In a sense, Jena provided us with a versatile search solution for the relatively complex linguistic annotations in a comparatively large treebank (of some 37,000 sentences) at quite moderate implementation cost.

In future work we seek to analyze more thoroughly the space of possible encodings and storage solutions, with the goal of documenting relevant trade-offs and limits to scalability and making available on-line additional search interfaces to common (semantic) annotation formats and data sets.[10] At the same time, we are working to generalize and parameterize our code base, so as to make it available for re-use and adaptation under an open-source license.

Besides such technical experimentation, we are also keenly interested in the application of additional semantic technologies, in the general realm of reasoning over RDF triple stores. To offer at least an indication of the direction of this work, in our running example *crops*, the third semantic argument to the *apply* predicate, is related to *cotton, soybeans, and rice* through a two-place relation introduced by the multi-word preposition *such as*. In this configuration, it is a common 'inference' to interpret the ARG2 of *such as* as an instance of the broader *crops* class; furthermore, the three-way conjunction allows a distributive reading—informally speaking, multiplying out incoming dependencies to all leaf nodes in the coordinate structure, as is for example done in the dependency scheme of de Marneffe and Manning (2008). There are several candidate solutions in the Semantic Web ecosystem that would seem to allow the encoding and processing of such rules of 'inference' over the RDF universe, thus in principle making it possible to also allow queries for (a generalized version of) the ARG3 of *apply* to match the nodes $x_{33}$ (*cotton*), $x_{43}$ (*soybeans*), and $x_{47}$ (*rice*) from Figure 2. This potential capability, in our view, further goes to show that semantic technologies—with built-in, general support for ontological modeling, directed labeled graphs, and (some) tractable reasoning capabilities—warrant more in-depth exploration for problems in language resource creation and use.

## Acknowledgements

We are grateful to our colleagues at the Oslo Language Technology Group and within the larger Deep Linguistic Processing with HPSG Initiative (DELPH-IN) for many fruitful discussions, as well as to three anonymous reviewers for insightful comments.

[10]See `http://wesearch.delph-in.net/` for an up-to-date list of existing semantic search interfaces built on top of our RDF-based infrastructure.

## References

Chiarcos, C. (2012). A generic formalism to represent linguistic corpora in RDF and OWL/DL. In *Proceedings of the 8th International Conference on Language Resources and Evaluation.* Istanbul, Turkey.

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. A. (2005). Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, *3*(4), 281 – 332.

de Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation* (p. 1 – 8). Manchester, UK.

Flickinger, D., Oepen, S., & Ytrestøl, G. (2010). WikiWoods. Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 7th International Conference on Language Resources and Evaluation.* Valletta, Malta.

Flickinger, D., Zhang, Y., & Kordoni, V. (2012). DeepBank. A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories* (p. 85 – 96). Lisbon, Portugal: Edições Colibri.

Ivanova, A., Oepen, S., Øvrelid, L., & Flickinger, D. (2012). Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop* (p. 2 – 11). Jeju, Republic of Korea.

König, E., & Lezius, W. (2000). A description language for syntactically annotated corpora. In *Proceedings of the 18th International Conference on Computational Linguistics* (p. 1056 – 1060). Saarbrücken, Germany.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpora of English: The Penn Treebank. *Computational Linguistics*, *19*, 313 – 330.

Mel'čuk, I. (1988). *Dependency syntax. Theory and practice*. Albany, NY, USA: SUNY Press.

Meurer, P. (2012). INESS-Search. A search system for LFG and other treebanks. In *Proceedings of the 17th International LFG Conference* (p. 404 – 412). Bali, Indonesia.

Oepen, S., & Lønning, J. T. (2006). Discriminant-based MRS banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation* (p. 1250 – 1255). Genoa, Italy.

Rosén, V., Meurer, P., Losnegaard, G. S., Lyse, G. I., De Smedt, K., Thunes, M., & Dyvik, H. (2012). An integrated web-based treebank annotation system. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories* (p. 157 – 167).

Rubiera, E., Polo, L., Berrueta, D., & Ghali, A. (2012). TELIX. An RDF-based model for linguistic annotation. In E. Simperl, P. Cimiano, A. Polleres, O. Corcho, & V. Presutti (Eds.), *The semantic web. Research and applications* (Vol. 7295, p. 195 – 209). Berlin, Germany: Springer.