Case Law Analysis using Deep NLP and Knowledge Graphs

Damir Cavar, Joshua Herring, Anthony Meyer
Indiana University, Semiring Inc.
Bloomington, IN
dcavar@iu.edu, joshua@semiring.com, anthony@semiring.com

Abstract

We present a system for mapping facts and knowledge in legal texts, in particular case law opinions and holdings to knowledge graphs, enabling advanced semantic search over the case law corpus, as well as matching of case descriptions onto case laws using graph similarity. The essential components for knowledge graph generations are deep linguistic NLP components. We discuss how the deep analyses provided by these components allow us to process not only the core semantic relations in the legal documents, but also to process advanced semantic and pragmatic properties, including implicatures and presuppositions.

Keywords: Case Law, Deep NLP, Knowledge Graph

1. Introduction

In this paper we discuss ongoing research and development activities in the domain of Deep Linguistic Natural Language Processing (NLP) technologies for the analysis of legal documents with a focus on court decisions, opinions, case law, and case documentation. Leveraging deep linguistic annotation technologies like semantic and pragmatic preprocessing to map cases, case law and opinions onto knowledge graphs enables us to bring the power of graph matching search and concept-based reasoning – involving automatically detected implicatures and presuppositions – to the world of legal AI (see e.g. Potts (2015)).

The goal of this project is to provide linguistically-informed, detailed analyses of case law based on professional-grade comparisons between individual cases. In pursuit of this goal, we model the content of individual case documents with knowledge graphs (KGs) built from semantic and pragmatic processing during the text-mining step. These KGs can later be used directly to compare individual cases, saying how they reinforce, contradict, or build on one another, eventually providing human-quality analysis of the current state of the body of law as well as automatic detection of trends that may have escaped human observers, all at a fraction of the cost in time and resources.

2. Previous Work

There are numerous commercial and free tools to search and process case law files. We will not go into details here with the existing commercial solutions. To our knowledge, none of these commercial solutions seems to provide a deep content analysis that is supported by fine grained linguistic and semantic technologies.

There are various documented and publicly available knowledge graph and ontology implementations for legal applications. Soria et al. (2007) describe an ontology of (Italian) *law paragraphs*, i.e., fundamental units of codified law. At the highest level, each paragraph belongs to one of three classes: *obligations*, *definitions*, or *modifications*. These are divided into subclasses. The class *obligations*, for instance, contains the subclasses *obligation*, *permission*, *prohibition*, and *penalty*. They also train a classifier to assign these classes to law paragraphs automatically. They report high precision and recall scores (96% and 92%, respectively).

3. Data

At present, our source material consists of the corpus provided by the Free Law Project (`https://free.law/`), which is an interface and mirror repository for the Public Access to Court Electronic Records (PACER) (`https://www.pacer.gov/`) service provided by the Administrative Office of United States Courts (`http://www.uscourts.gov/`) to facilitate public electronic access to federal court records. The bulk of the data we survey comes from the Free Law Project's CourtListener (`https://www.courtlistener.com/`) service and takes the form of compressed JSON files representing individual cases, organized by jurisdiction.

The JSON objects include meta-information and multiple content sections, but there is frequently no explicit separation of the opinion/holding from fact-finding and other components of the case. To detect the opinions in the case files, we use Machine Learning (ML) approaches, training automatic classifiers on a sub-corpus manually annotated by legal experts to separate the holding from residual contextual information about the facts of the case.[1]

4. Architecture

The primary step in processing is to extract core semantic relations, e.g. subject – verb – object, from clauses in the text, which we do by means of NLP components. We use the Natural Language Toolkit (NLTK) (Bird et al., 2009) components for basic segmentation and tokenization, followed by Part-of-Speech (PoS) tagging and WordNet-based[2] hypernym, hyponym, and synonym annotation of

[1] We use Scikit Learn (Buitinck et al., 2013; Pedregosa et al., 2011) and additionally various text classifiers based on Bayesian or Support Vector Machine approaches.

[2] For details on WordNet see Miller (1995) and Fellbaum (1998)

nominal elements. Providing the extended taxonomic relations for allows us to index textual content such that concept search and search over synonyms is made possible of the case law corpus. This information is also essential for mapping of concrete concepts to concepts in KRs, as explained below.

The Stanford CoreNLP (Manning et al., 2014) pipeline provides extended analytical components, including lemmatizer, a constituent parser, a dependency parser, and a coreference analyzer. The spaCy pipeline (an implementation of Honnibal and Johnson (2015)) is comparable to CoreNLP, except that it does not have constituent parsing and coreference analysis components.

All these components face performance issues and tend to fail on complex sentences or sentences that exceed a particular length. By way of example, the following sentence will receive some linguistic annotation of very limited use:

> *Their attack is anchored in a Fifth Circuit case, United States v. Whitfield, which involved two state judges who were convicted of accepting bribes from an attorney in exchange for favorable rulings in his cases.*[3]

It is not uncommon for the types of constructions found in formal documents to be misanalyzed, particularly clause level coordination, constructions with ellipsis or gapping, empty subject constructions, and many other constructions which require processing that goes beyond the level of combining textually represented words into composite meanings. Rimell et al. (2009) and Nivre et al. (2010) report that even the best parsers perform quite poorly where unbounded dependencies are concerned, i.e., dependencies such that there is no theoretical limit on the distance between head and dependent. Often, parses for such constructions are unsystematic and unpredictable, and we are consequently forced to augment them using the sub-optimal linguistic output across various NLP pipelines. For example, consider the sentence

> *They tasted the specimens to identify them.*

which contains a purpose clause, namely, *to identify them*. In CoreNLP's analysis for this sentence, the matrix verb *tasted* to *identify* via `advcl` (adverbial clause), the Stanford Dependency category that includes purpose clauses (De Marneffe et al., 2014). However, if *tasted* is replaced by *were tasting*, CoreNLP returns a parse in which *were tasting* is related to *specimens* via `dobj`, which is turn related to *identify* via the tag `acl` (adjectival clause). In other words, a simple change in verb tense can result in a fundamentally different analysis that overlooks a key semantic relation.

In addition to freely available NLP components and pipelines, we make use of in-house technology and infrastructure. Within the Free Linguistic Environment (FLE) project (Cavar et al., 2016) we developed multi-word morphological analyzers using a two-level transducer framework as made available in the Foma morphology compiler (Hulden, 2009). In this way, we are able to generate Finite State Transducers (FST) that recognize single- and multi-word named entities and jargon specific to the legal domain, such as *amicus curiae*. The recognized terminology is annotated using a "legal" tag, as well as semantic sub-type information, wherever applicable. In our case, the terminology is augmented with domain specific tags, for example indicating that an expression like "FMLA"[4] is typical in the labor law domain, while "exclusive rights" indicates the copyright law domain. An FST can read in a term like "FMLA" and output an analysis listing its tag(s) and sub-tag(s), much like a two-level morphological analysis in the manner of (Koskenniemi, 1983).

To be able to generate deeper linguistic analyses that cover linked constituent structure, functional relations, and morpho-syntactic and semantic properties, we work with a (Probabilistic) Lexical Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001; Cavar et al., 2016) based parsing system. Such a parsing system generates syntactic structures that encode scope relations between sentential (or clausal) elements, which is essential in, among other things, semantic processing of quantifiers, time reference, and negation. The FLE project is related to the Xerox Linguistic Environment (XLE) (Crouch et al., 2011) project, which is the most significant and complete implementation of the Lexical Functional Grammar (LFG) framework (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) in a grammar engineering environment. It comes with the additional advantage of being well-documented, in, among other sources, a grammar engineering textbook (Butt et al., 1999), official technical documentation (Maxwell and Kaplan, 1996), and various online material (Crouch et al., 2011).

The analytical strength of LFG-based parsers can be exemplified using a simple example. The sentence *They offer several justifications for this position*[5] would receive a constituent structure (or c-structure) analysis as in figure 1.

The corresponding functional structure (or f-structure) is given in figure 2.[6]

The two representations are linked, such that every tree node has a link to an attribute-value-matrix (AVM) in the f-structure. The f-structure provides a rich set of morpho-syntactic and semantic features that are extremely useful for higher level analysis of semantic relations between arguments. The c-structure provides a phrase-structure analysis that represents scope relations between the sentential arguments and modifiers. It is in particular essential for the processing of scope of negative elements, operators, and quantifier.

Such rich representational output allows us to map the con-

[3]United States v. Martinez-Maldonado, United States Court of Appeals, First Circuit Nos. 12–12 89, 12–1290 see `http://media.ca1.uscourts.gov/pdf.opinions/12-1289P-01A.pdf`.

[4]Family Medical Leave Act, Public Law No. 103–3, 29 U.S.C. Sections 2601–2654 (1993)

[5]See footnote 3.

[6]The output for the c- and f-structure in figures 1 and 2 was generated using the XLE-Web interface `http://clarino.uib.no/iness/xle-web`. See for example Meurer et al. (2016).

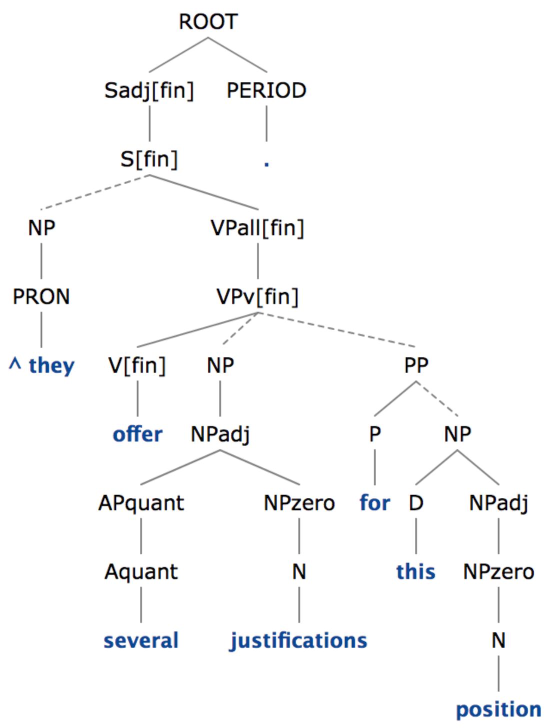

Figure 1: C-structure

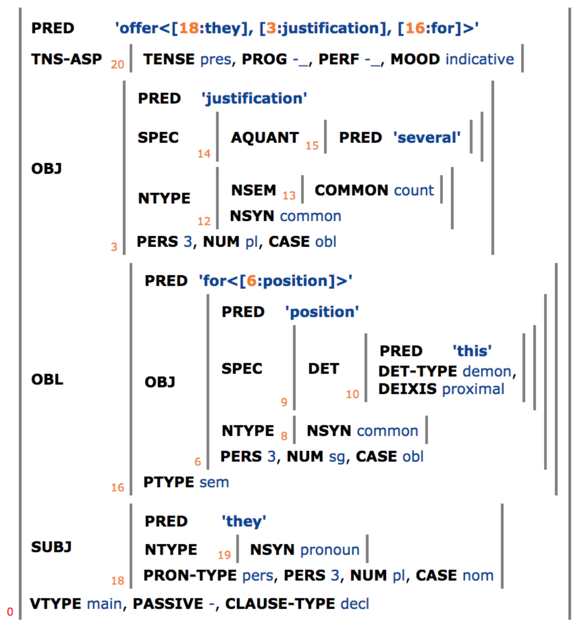
PRED 'offer<[18:they], [3:justification], [16:for]>'
TNS-ASP 20 TENSE pres, PROG -_, PERF -_, MOOD indicative
OBJ
PRED 'justification'
SPEC 14 AQUANT 15 PRED 'several'
NTYPE 12 NSEM 13 COMMON count
NSYN common
3 PERS 3, NUM pl, CASE obl
OBL
PRED 'for<[6:position]>'
OBJ
PRED 'position'
SPEC 9 DET 10 PRED 'this'
DET-TYPE demon,
DEIXIS proximal
NTYPE 8 NSYN common
6 PERS 3, NUM sg, CASE obl
16 PTYPE sem
SUBJ
PRED 'they'
NTYPE 19 NSYN pronoun
18 PRON-TYPE pers, PERS 3, NUM pl, CASE nom
0 VTYPE main, PASSIVE -, CLAUSE-TYPE decl

Figure 2: F-structure

tent of the target sentence to a KR by converting the main predicate to a triple, i.e. predicate *offer* – subject *they* – object *justification*. The subject *they* would be linked to a real referent (antecedent) in a given context using anaphora resolution and correference analyzers.

Additional properties for all arguments can be extracted from the f-structure in conjunction with sentential features, as well as dependency relations and functional roles of clausal elements.

To map content from unstructured text (a list of sentences and clauses), it is essential to identify the tense information, mood, or voice of the sentence, any scope relations between constituents and clauses within the sentence, and quantifiers or semantic operators, such as whether a negating element in a clause scopes over the entire clause or merely an isolated phrase, and whether it has scope over the matrix clause and thus potentially all embedded clauses in its scopal domain. It is also crucial to detect voice so as to determine the directionality of a relation in a tuple. The examples *The plaintiff accused the defendant of breach of contract* and *The defendant was accused of breach of contract by the plaintiff* would receive the same directionality representation of the KR predicate relation. This is obviously not true for a sentence like *The plaintiff was accused by the defendant of breach of contract*. Likewise, if the sentence uses future tense, it is not a factual or assertive statement that should be integrated into a common KR that represents factual knowledge, as for example in *Google will buy Apple*. Also, a past tense assertive statement embedded under a hypothetical or future tense matrix clause does not represent concepts and relations that should be part of a KR, as for example *We do not believe that Google bought Apple*. Our NLP components are capable of detecting mood, tense, and voice in the input sentences. Additionally, they can predict clausal structures of complex sentences as well as the scope relations between these clauses. This is in particular relevant, when it comes to the correct extraction and mapping of semantic relations in embedded contexts. If an embedded clause with assertive content of the kind *that the plaintiff transfered the funds to the bank account* can only be interpreted if the matrix clause as for example "*It is true*" is not negated, not using future tense or the subjunctive, and so on. The linguistic properties and the scope relations between the clauses are essential to be able to correctly differentiate hypotheticals from factive assertions, or guesses from wishful projections.

To extend this capability to legal language, we not only develop our own domain specific adaptations of NLP components and pipelines, we also post-process the outputs of the aforementioned openly accessible NLP components. The post-processing extends the linguistic analytical output and also corrects systematic errors of certain NLP components.[7]

In the FLE implementation we use a probabilistic model of the c-structure parser backbone as well as a probabilistic unification algorithm over Probabilistic Directed Acyclic Graphs (PDAG) for the AVM and f-structure representations. This specific version of an LFG-type of parser allows us to engineer more robust grammars that can cope with

[7]Some such errors are construction specific. Coordination of clauses as common in legal documents is systematically analyzed as local phrase coordination. Predicative modifiers as for example adverbial temporal constructions or prepositional location phrases in syntactic parse trees are frequently attached to adjacent noun phrases (low attachment), rather than the predicate. Many of these mistakes can be corrected using simple post-processing steps.

agreement violations or unification failures, as well as word order violations or complexities that other NLP pipelines cannot process. The grammars that underly such a parser can be engineered and trained using corpora and distributional models over syntactic trees and morpho-syntactic features.

Our architecture wraps all available NLP components in a Remote Procedure Call (RPC) (Microsystems, 1988) set of micro-services and lets each service process each input sentence in parallel. The outputs of each component are evaluated, scored and transformed into a synthesized unique Linguistic Data (LingData) data structure (or object). This LingData object decides on a uniform representation of the data, i.e. PoS-tags and dependency relations are normalized, clause boundaries are added or removed, etc. We implemented a specific extension that interprets constituent structure trees and dependency graphs into phrasal scope relations that allow us to query the hierarchical relations between all tokens, phrases and phrasal nodes in the sentence structure. This implies that we can not only identify a negation in a clause, but also the correct structural scope of it. In the sentence *the plaintiff did not violate corporate policies* the negation is correctly identified as sentential negation, while in the sentence *the plaintiff violated corporate policies and not federal law* the negation scope would be local over *federal law* only.

The resulting LingData objects thus contain complex linguistic properties and annotations. These are accessible using class specific methods. Currently the class is only available as a Python implementation. In future versions we will provide a C++ and a Go implementation as well.

The different LingData objects generated by parallel NLP pipelines are then combined into a single, hopefully complete and correct, analysis, using mapping of the linguistic analyses and extensions generated from the outputs of the NLP components in a Unification method.

Such a parallel architecture is complex and computationally expensive. It can, however, be easily scaled given our choice of a JSON-based RPC micro-services infrastructure regulated through a core processing dispatcher or manager. This infrastructure frees us from distracting programming language, operating system, or other technical dependencies, as components can be swapped, added, coupled, or removed as and when the need arises. For most of the advanced NLP components and pipelines the loading time of models is eliminated, since the components run in daemon mode and communicate over TCP/IP with the clients.

4.1. Knowledge Graph Mapping

As described above, the core semantic relations extracted by isolating the core predicate in a clause and its dependent functional phrases provide the core relational elements or sub-graph for the KR. Additional processing is necessary to map these attributes and properties to relations between concepts or concept attributes. For example, the construction *the plaintiff was employed as a clerk in the defendant's firm* could imply that the plaintiff is a clerk or that the concept of the asserted plaintiff in the KR has an attribute-value specification *profession – clerk*. We have an independent model for such mappings that allows us to generate graph relations for the specific domain or use-case.

The processing of semantic and pragmatic relations allows us to expand the KR even further. To be able to process implicatures or presuppositions, the NLP output needs to provide detailed information about nominal elements or phrases in the clauses. For example, if the direct object in a clause is a definite and specific noun phrase like *the plaintiff bought the blue car*, implicatures that can be generated to extend the KR representation of the situations and events would include factual statements like *there were multiple cars available that the plaintiff could have bought* and *no other of these cars is blue*. Likewise, a statement like *the plaintiff was petting his dog* presupposes that the statement *the plaintiff owns a dog* is true as well. While most of these implicatures and presuppositions will strike human readers as trivial, they can provide valuable information for automatic processing and KR generation. Moreover, some semantic and pragmatic side-effects so inferred might not be easily accessible to the reader at all – a situation that is particularly frequent in highly specialized, knowledge-based domains like legal reasoning.

For the processing of such relations we build construction-specific mappings for the domain, the particular language, and linguistic constructions in general. The mapping of definite and specific noun phrases to imply the existence of a super-set is an example of a linguistic property that can be applied universally in all linguistic domains. Other such mappings are language-specific and can depend on cultural peculiarities. By contrast, many of the semantic and pragmatic properties are domain-specific, and their specification and definition in specific NLP components requires the supervision and involvement of trained domain experts (legal professionals, for the present case).

At the graph level we use two commercial environments: Neo4J and Stardog. Both products are advanced graph databases with different capabilities when it comes to semantic processing. Neo4J serves as an experimental simple, but highly performant and scalable, graph representation system where we do not make use of extended semantic technologies like OWL-based ontologies (W3C OWL Working Group, 2012; W3C OWL Working Group, 2009) or reasoning (using a Description Logic framework).*See for example Antoniou and van Harmelen (2004).* Stardog, by contrast, functions as an extended graph with OWL-backing assertions of facts, concepts, relations, and attributes. We augment Stardog with Pellet (Sirin et al., 2007) as a reasoner. One goal is to use the ontology as a classification system for concepts that allows us to generate extended properties for asserted individuals. For example, if an ontology defines *CEOs* to be *humans*, and *humans* have *birthday*s, *gender* and *parents* as properties, when we assert that *John Smith is CEO*, the system can automatically extend the properties of the concept *John Smith* to include the implications (*John Smith*) *has a birthday*, *has gender*, *has parents*, etc. This level of semantic expansion using common reasoners (e.g. Pellet) augments potentially sparse assertions and makes hidden facts and circumstances explicit and available for search and graph-based comparison or analysis.

Another goal is to detect conflicts in assertions related to the types of concepts. For example, if we assert that *Grungle Inc.* is a company, and the ontology encodes taxonomic relations or the concept hierarchy that a CEO is a (*isA*) human, an assertion of the type *Grungle Inc. is the CEO of Sprackets Inc.* can be flagged or rejected as a violation of base relations formulated in the ontological concept relations. While common OWL-based assertion handling would not be able to catch such violations, extensions of reasoners like Pellet can be used to detect conflicting assertions of this kind.[8]

Our analysis of different graph databases for the back-end storage of a knowledge graph in our system did also include the Apache Jena (`jena.apache.org`) environment. Due to obvious limitations here, we will extend the discussion of the suitability of knowledge graph storages for our purposes to subsequent publications. In addition to these free systems we made arrangements to evaluate other commercial graph database systems as for example TigerGraph (`www.tigergraph.com`), where our main interest lies in performance for search and graph comparison with large knowledge graphs.

5. Discussion

Given the limits of this article, it is of course impossible to provide an exhaustive list of the capabilities and advantages that deep NLP and semantic processing using Description Logic, OWL ontologies, and reasoning can bring to legal language processing. We hope we have nevertheless been successful in conveying the impression that they are prodigious, representing an evolutionary leap in applicative power. Mapping case law documents, in particular the opinion and the holding, to KRs allows us to search over the document base via graph similarity.

Mapping specific concepts to hypernyms introduces an conceptual abstraction layer that allows us to identify cases with concrete reference to for example *injury involving a semi-truck* can be found by searching for *injury involving a vehicle* or even *car*.

Mapping concrete case files onto a graph representation in a systematic way allows us to use graph similarity search to identify semantically related cases, holdings, and opinions. Using comparisons of graphs within a similar concept and relation space allows us to identify conflicting opinions in the case law, or conflicting facts in other document types. These types of conflict studies open up new possibilities for the automatic analysis of case law holdings and opinions.

We are aware of the fact that we owe the reader a detailed explanation of the architecture, performance, and issues related to the NLP components and architecture.

One serious problem for us in the current situation is that we do not have any objective measure for the performance of our system, due to the lack of gold standard resources and corpora. While we can describe the technical and runtime behavior, the accuracy of some NLP components, we cannot yet easily assess on a larger scale the extraction of semantic relations and concepts.

Due to a lack of appropriate resources, our evaluation right now can only be based on a usefulness study with paralegals and law firm employees.

Due to space limitations, we defer the exposition and discussion of the results of experiments and concrete applications to the concrete conference presentation and subsequent publications.

6. Acknowledgments

We are grateful for useful comments, code and algorithm implementations, and suggestions related to the project to all our other team members, in particular Stefan Geissler, Matthew Rust, Jacob Heredos, Malgorzata E. Cavar, and Lwin Moe.

Many more colleagues have helped us with work on the FLE project. We are grateful to Ron Kaplan, Ken Beesley, Lionel Clément, Larry Moss, Mary Dalrymple, Agnieszka Patujek, Adam Przepiórkowski, Paul Meurer, Helge Dyvik, Annie Zaennen, Valeria de Paiva for many helpful suggestions, data sets, grammar samples, ideas, and comments. Numerous other colleagues have been supportive, providing suggestions and advise in various development phases. Our colleagues from the Kelley School of Business or the Maurer School of Law at Indiana University have provided valuable domain specific information and knowledge. Various local lawyers have helped us to understand the essential processes and tasks involved in legal research.

7. Bibliographical References

Antoniou, G. and van Harmelen, F., (2004). *Web Ontology Language: OWL*, pages 67–92. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Butt, M., King, T. H., Niño, M.-E., and Segond, F. (1999). *A Grammar Writer's Cookbook*. CSLI Publications.

Cavar, D., Moe, L., Hu, H., and Steimel, K. (2016). Preliminary results from the free linguistic environment project. In Doug Arnold, et al., editors, *Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*, pages 161–181. CSLI Publications.

Crouch, D., Dalrymple, M., Kaplan, R., King, T., Maxwell, J., and Newman, P. (2011). XLE documentation. Online document at http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html.

Dalrymple, M. (2001). *Lexical Functional Grammar*. Number 42 in Syntax and Semantics. Academic Press, New York.

[8]The developers of Pellet and Stardog informed us that this is a possibility in their system, and we assume that this is missing in other non-OWL-based graph-databases. Such restrictions can also be implemented in the free and open Apache Jena environment.

De Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–92.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September. Association for Computational Linguistics.

Hulden, M. (2009). Foma: A finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32. Association for Computational Linguistics.

Kaplan, R. M. and Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. Cognitive Theory and Mental Representation, pages 173–281. The MIT Press, Cambridge, MA.

Koskenniemi, K. (1983). Two-level morphology: A general computational model for word-form production and generation. *Publications of the Department of General Linguistics, University of Helsinki. Helsinki: University of Helsinki*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Maxwell, J. and Kaplan, R. M. (1996). An efficient parser for LFG. In *Proceedings of the First LFG Conference*. CSLI Publications.

Meurer, P., Rosén, V., and Smedt, K. D. (2016). Interactive visualizations in the iness treebanking infrastructure. In Annette Hautli-Janisz et al., editors, *Proceedings of the LREC'16 workshop VisLR II: Visualization as Added Value in the Development, Use and Evaluation of Language Resources*, pages 1–7, Portorož, Slovenia. ELRA.

Microsystems, S. (1988). RPC: Remote procedure call protocol specification. Request For Comment (RFC) 1057.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nivre, J., Rimell, L., McDonald, R., and Gomez-Rodriguez, C. (2010). Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841. Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Potts, C. (2015). Presupposition and implicature. In Shalom Lappin et al., editors, *The Handbook of Contemporary Semantic Theory*, pages 168–202. Wiley-Blackwell, 2 edition.

Rimell, L., Clark, S., and Steedman, M. (2009). Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 813–821. Association for Computational Linguistics.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semant.*, 5(2):51–53, June.

Soria, C., Bartolini, R., Lenci, A., Montemagni, S., and Pirrelli, V. (2007). Automatic extraction of semantics in law documents. In *Proceedings of the V Legislative XML Workshop*, pages 253–266.

W3C OWL Working Group. (2009). OWL 2 web ontology language document overview. Technical report, W3C, October. http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.

W3C OWL Working Group. (2012). OWL 2 web ontology language document overview (second edition). Technical report, W3C, December. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.