A FORMAL SPECIFICATION FOR SYNTACTIC ANNOTATION AND ITS USAGE IN CORPUS DEVELOPMENT AND MAINTENANCE: A CASE STUDY IN UNIVERSAL DEPENDENCIES

Guilherme Paulino Passos

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Gerson Zaverucha
Alexandre Rademaker

Rio de Janeiro
Agosto de 2018

A FORMAL SPECIFICATION FOR SYNTACTIC ANNOTATION AND ITS USAGE IN CORPUS DEVELOPMENT AND MAINTENANCE: A CASE STUDY IN UNIVERSAL DEPENDENCIES

Guilherme Paulino Passos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Gerson Zaverucha, Ph.D

Prof. Alexandre Rademaker, D.Sc.

Prof. Mario Roberto Folhadela Benevides, Ph.D.

Prof. Marcelo Finger, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2018

Paulino Passos, Guilherme

A Formal Specification for Syntactic Annotation and its Usage in Corpus Development and Maintenance: A case study in Universal Dependencies/Guilherme Paulino Passos. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIII, 165 p.: il.; 29, 7cm.

Orientadores: Gerson Zaverucha

Alexandre Rademaker

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 149 – 165.

1. Natural language processing. 2. Syntactic parsing. 3. Knowledge representation. I. Zaverucha, Gerson *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha família, em especial
meus pais, Fabio e Lucia, e
minha irmã, Juliana.

Agradecimentos

Agradeço aos meus pais, por todo o apoio, atenção e carinho. Foi graças aos esforços de vocês, assim como o do resto da família, que pude chegar onde estou. Agradeço também à minha irmã Juliana pelo companheirismo.

Aos meus orientadores, Professor Gerson Zaverucha e Professor Alexandre Rademaker. Todas as contribuições, discussões e lições foram de grande valia para crescimento acadêmico, profissional e pessoal.

À Professora Cláudia Freitas, do Departamento de Letras da PUC-Rio, pelas discussões e comentários sobre este trabalho e sobre Lingüística Computacional em geral, bem como às suas alunas, Luísa Rocha e Isabela Soares-Bastos.

Aos Professores Mario Benevides e Marcelo Finger, pela participação na banca e todos os comentários, críticas e sugestões, que melhoraram o trabalho e trouxeram novas idéias.

Aos professores do Programa de Engenharia de Sistemas e Computação, por contribuírem para meu aprendizado com suas aulas, conselhos e discussões.

Aos colegas de mestrado, pela colaboração e companhia, em particular ao Victor Augusto Lopes Guimarães, companheiro em diversas disciplinas e no laboratório.

Aos colegas do Brasil Research Lab da IBM Research, que contribuíram para uma experiência única e me fizeram compreender a vivência em um grupo de pesquisa desta magnitude. Não apenas o apoio e a colaboração profissional foi de grande valor, mas também a amizade e o apoio emocional mútuo nutridos.

Aos colegas do Supremo em Números, pela amizade e apoio. Ainda que minha participação tenha sido bastante limitada, não só aprendi com vocês mas também fui muito bem recebido e incluído por todos.

Aos amigos de diversas áreas, que sempre estiveram lá para escutar, aconselhar e apoiar. Também agradeço a compreensão com as ausências e restrições, mas principalmente agradeço a presença por todos os momentos que pude compartilhar.

Finalmente, agradeço à CAPES pelo suporte financeiro, bem como à IBM Research pela oportunidade e também suporte para as atividades que resultaram nesta dissertação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA ESPECIFICAÇÃO FORMAL PARA ANOTAÇÃO SINTÁTICA E SEU USO NO DESENVOLVIMENTO E NA MANUTENÇÃO DE CORPORA: UM ESTUDO DE CASO EM DEPENDÊNCIAS UNIVERSAIS

Guilherme Paulino Passos

Agosto/2018

Orientadores: Gerson Zaverucha
Alexandre Rademaker

Programa: Engenharia de Sistemas e Computação

Dados anotados linguisticamente são atualmente um recurso crucial para processamento de linguagem natural (NLP). Tais dados são necessários tanto para avaliação empírica de sistemas, quanto para o treinamento de modelos de aprendizado de máquina de linguagem. Contudo, produzir novos conjuntos de dados é muito custoso em tempo e trabalho humano. Usualmente algum domínio em linguística é necessário aos anotadores, e ainda assim a decisão de como anotar não é trivial. Em projetos com muitos anotadores ou abrangendo longos períodos de tempo, a consistência da anotação pode ser comprometida. Ademais, anotar dados de domínios específicos requer anotadores com conhecimentos correspondentes. Isso se torna um sério problema para domínios técnicos como ciências biomédicas, óleo e gás e direito. Neste trabalho, contribuímos para diminuir esta dificuldade na produção de textos com anotação sintática (*treebanks*) por métodos formais. Nós desenvolvemos uma specificação formal do padrão de anotação sintático Dependências Universais (*Universal Dependencies*), um projeto desenvolvido pela comunidade internacional de NLP e de crescente importância. Sustentamos que essa especificação formal é útil para melhorar a qualidade de *treebanks* e reduzir custos de anotação, pela imposição de consistência nos dados. Discutimos as características, decisões de projeto e limitações da nossa ontologia, implementada na linguagem OWL2-DL. Avaliamos experimentalmente a utilidade de nossa ontology na tarefa de detectar análises incorretas automaticamente, mostrando alta precisão em quatro idiomas. Finalmente, contextualizamos nossa contribuição revisando o estado da arte no desenvolvimento e manutenção de *treebanks*.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A FORMAL SPECIFICATION FOR SYNTACTIC ANNOTATION AND ITS USAGE IN CORPUS DEVELOPMENT AND MAINTENANCE: A CASE STUDY IN UNIVERSAL DEPENDENCIES

Guilherme Paulino Passos

August/2018

Advisors: Gerson Zaverucha
Alexandre Rademaker

Department: Systems Engineering and Computer Science

Linguistically annotated data are currently crucial resources for natural language processing (NLP). They are necessary for both evaluation and as input to training machine learning models of language. However, producing new datasets is a very time and labor-consuming. Usually some expertise in linguistics is required for annotators, and even so the annotation decision problem is far from trivial. This difficulty grows in scale: in projects with many annotators or spanning a long period of time, annotation consistency can be compromised. Furthermore, annotating data from specific domain requires annotators with corresponding knowledge. This is a serious problem for technical domains such as biomedical sciences, oil & gas and law. In this work, we contribute to solving the problem of producing syntactically annotated texts (treebanks) by formal methods. We develop a formal specification of the syntactic annotation standard Universal Dependencies, a project developed by the NLP community around the world which is growing in importance. We argue that this formal specification is useful for improving the quality of treebanks and reducing annotation costs, by enforcing consistency in the data. We discuss the features, design choices and limitations of our ontology, implemented in the OWL2-DL language. We evaluate experimentally the usefulness of our ontology in a task of automatically detecting wrong analysis, showing high precision in four languages. Finally, we contextualize our contribution by surveying state-of-the-art methods for developing and maintaining treebanks.

Contents

List of Figures **x**

List of Tables **xiii**

1 Introduction **1**

1.1 Dissertation overview and contributions 2
1.2 Computational syntactic analysis 2
1.3 A case study on syntactic annotation: Universal Dependencies 7

2 A Formal Specification For Universal Dependencies **15**

2.1 Motivation 15
2.2 Formal foundations and language 17
2.3 An Ontology for Universal Dependencies 18
2.3.1 Concepts in the Universal Dependencies annotation guidelines 18
2.3.2 Ontology summary 19
2.3.3 Modeling 20
2.3.4 Documentation inconsistencies 23
2.4 Validating UD CoNLL-U files 23

3 Experimental Evaluation **29**

3.1 Experiments 29
3.1.1 Finding “incorrect” analyses 29
3.1.2 Error analysis and discussion 31
3.1.3 Comparison with Udapi 33

4 Treebank Development and Maintenance **37**

4.1 Querying 38
4.2 Consistency verification 40
4.2.1 External consistency 40
4.2.2 Internal consistency 41
4.3 The Semantic Web approach 42
4.4 Literate programming for treebanks: a Jupyter notebook approach . . 43

4.4.1 System architecture 44
4.4.2 Example usage 45
4.5 Other approaches 47
4.6 Example practical application 48

5 Conclusion **51**
5.1 Discussion 51
5.2 Future Work 52

A Ontology **54**

Bibliography **149**

List of Figures

1.1 Example of dependency analysis. Sentence and analysis from the UD 2.2 English EWT treebank. 3
1.2 Example of phrase-structure analysis, in the style of the Penn Treebank. Sentence from the UD 2.2 English EWT treebank. 3
1.3 UD analysis of a verbal clause, with part-of-speech tags and morphosyntactic features. Adaptation of sentence found in Wikipedia, Current Events: `https://en.wikipedia.org/wiki/Portal:Current_events`. Original sentence under CC-BY-SA license. The analysis was simplified by omiting punctuation from the tree. 9
1.4 UD analysis of a verbal clause with a subordinate clause. Sentence found in Wikipedia, Current Events: `https://en.wikipedia.org/wiki/Portal:Current_events`. Original sentence under CC-BY-SA license. The analysis was simplified by omiting punctuation from the tree. 9
1.5 UD analysis of a copular clause, with part-of-speech tags. Adaption of sentence found in Wikipedia:About: `https://en.wikipedia.org/wiki/Wikipedia:About`. Original sentence under CC-BY-SA license. 10
1.6 UD analysis of coordination between the nominals "*project*" and "*community*". Sentence and analysis from the UD 2.2 English EWT treebank. For readability, we show only the analysis of the subphrase "*with the Mozilla project and community*". 11
1.7 UD analysis of coordination between two nominals. Example of (arguably) non-shared dependent. Sentence and analysis from the UD 2.2 English EWT treebank. 11
1.8 Slightly altered version of the sentence in Figure 1.7. 11
1.9 A modified version of the sentence in Figure 1.11 without any ellipsis. It modifies sentence in Figure 1.10 by inserting the highlighted word "*cigarros*". 12

1.10 A modified version of the sentence in Figure 1.11 without the ellipsis of the verb "*fuma*". 13
1.11 Sentence and analysis from the UD 2.2 Portuguese Bosque treebank. The `orphan` relation in highlight indicates that an elided predicate word would have both "*mulher*" and "*5*" as dependents. A word-for-word translation is accompanying the original sentence. 13
1.12 Enhanced dependencies version of the sentence in Figure 1.8. The adjectival modifier (`amod`) relation of "*lovley*" is propagated to "*chips*", meaning that it refers to it as well. 13
1.13 Tabular CoNLL-U format of the sentence in Figure 1.7. 14

2.1 Architecture of the UD Theory ontology. Each node is an OWL file. . 20
2.2 Concepts in `ud-theory`. 22
2.3 Validation pipeline. 24
2.4 Example sentence with an error on the direction of `appos` relation. . . 26
2.5 Example sentence with an error on a `appos` relation headed by a `VERB`, not by a nominal. 26
2.6 Explanation automatically given to analysis 2.4. 27
2.7 Explanation automatically given to analysis 2.5. 27

3.1 Pipeline for experiment described in Section 3.1.1. 30
3.2 Example of "not" in a sentence of the UD EWT English corpus. . . . 31
3.3 Example of an `ADP obl` word in a sentence of the UD EWT English corpus. 32
3.4 Example of an `ADP obl` word in a sentence of the UD EWT English corpus. 32
3.5 Partial analysis for a sentence of German GSD with a part-of-speech error. 33
3.6 Sentence of UD Bosque Portuguese corpus with clausal subject (`csubj`) in nonverbal clause. It could be translated as "*Stopping this attack in blockage is an almost impossible mission.*" 33

4.1 Example of query for right-headed `conj` relation in the SETS DEP_SEARCH tool (LUOTOLAHTI *et al.*, 2017). 38
4.2 Example of query for `conj` relation between words with different part-of-speech tag in the Grew tool (GUILLAUME *et al.*, 2012). 39
4.3 Result for the query in Figure 4.2. 39
4.4 Architecture of Jupyter for treebanks, including the validation service. 45
4.5 Example of printing analysis of two sentences in the CoNLL-U format. 46

4.6 Example of printing a drawing of the tree of the first sentence (MUNIZ *et al.*, 2017). 46
4.7 Example of validation output of the two sentences. 47

List of Tables

2.1 UD Theory ontology metrics and expressivity used. 19

3.1 Results of the experiment described in Section 3.1.1. It was run in 25 folds cross validation. Values in the columns marked with an * are averages on every fold, with standard deviation in parentheses. 33

3.2 Results of the Udapi experiment. It was run in 25 folds cross validation. Values in the columns marked with an * are averages on every fold, with standard deviation in parentheses. 34

3.3 Difference between ontology-based validator and Udapi in each dataset, in percentage points. Values marked with $\star$ mean that the difference is significant by a level of 0.001. Significance is measured by paired Student's t-tests. 35

Chapter 1

Introduction

Natural language processing (NLP) requires *annotated* (also called *labeled*) linguistic data, that is, instances of language use enhanced with additional linguistic information about it. Not only it is essential for developing supervised machine learning applications, but it is also required for the evaluation of any NLP system (PALMER and XUE, 2010; RESNIK and LIN, 2010). In syntactic annotation, the focus of this work, labeled datasets are usually called *treebanks*.

While treebanks exist for different languages and domains, they are not available for every case and their quality may be uncertain. Indeed, even for high-resource languages, many treebanks are originated from newswire text, for example the Wall Street Journal part of the Penn Treebank (MARCUS *et al.*, 1993). Large and reliable treebanks for specific domains such as legal texts, oil & gas and mining are still missing. Besides, treebanks come in different annotation formats, which can be a problem in using for practical applications, as conversion can be noisy.

However, producing this kind of data is very costly and labor-intensive. This problem is intensified as in many cases datasets contain errors (that is, they are *noisy*), and so greater amounts of data are needed in order to compensate for this reduced quality. For solving this problem, three lines of research are possible: (i) methods for improving the quality of datasets, that is, improving their correctness; (ii) methods for reducing annotation cost, in both labor, time or money; and (iii) methods for reducing the need for labeled data.

In this work, we contribute to this problem by presenting a formal specification of a syntactic annotation standard, which allows an exact definition of possible annotations. We argue that this resource is potentially helpful in all three lines of research, as: (i) a formal specification yields a method for automatically detecting (possible) errors and for improving queries on data; (ii) this detection of errors can be exploited for producing new annotated data; (iii) a formal specification can potentially be explored by machine learning tools for reducing possible hypothesis to be considered, which would allow more precise learned models with less labeled

data.

1.1 Dissertation overview and contributions

- We present an OWL-DL ontology for the Universal Dependencies standard, presenting its features, design choices and limitations (Chapter 2). It covers most of the guidelines for part-of-speech tagging and dependency relation annotation.
- We evaluate its usefulness in the task of detecting wrong analyses with respect to a gold standard test set, showing that it has high precision in different languages, although not in all of them (Chapter 3). We compare with an already existing (but non-declarative) validator for Universal Dependencies.
- We survey methods for developing and maintaining datasets for syntactical annotation (treebanks), contextualizing the contribution of our ontology (Chapter 4).

In the rest of this chapter we will present the background of our work.

1.2 Computational syntactic analysis

The study of *syntax* consists in understanding how words are related to each other to form (meaningful) sentences and texts (VAN VALIN, 2004; VAN VALIN and LAPOLLA, 1997). This combination is usually expressed in some form of *representation*. Syntactic parsing is the task of, on receiving a sentence, producing such representation of the syntactic structure that describes in some way these relationships between words in that sentence[1]. Examples of syntactic representations are in Figures 1.1 and 1.2. A dataset of natural language text is called a *corpus*, while a corpus annotated with structure analysis, is called a *treebank*. In the rest of this work we will use *corpus* and *treebank* in the sense of *syntactically annotated natural language text*, unless otherwise specified.

On the scientific side, annotating syntax is useful for empirical analyses of language, such as typological studies: classifying languages, hypothesizing universals and explaining linguistic structure (CROFT, 2002, chapter 1). A powerful motivation for the study of linguistics has been understanding the historical evolution of languages, for which syntax can be informative (SAG *et al.*, 2003, p. 7). Some

[1]This definition is not entirely uncontroversial. STEEDMAN (2000) argues that syntactic parsing is "merely the characterization of the process of constructing a logical form, rather than a representational level of structure that actually needs to be built".

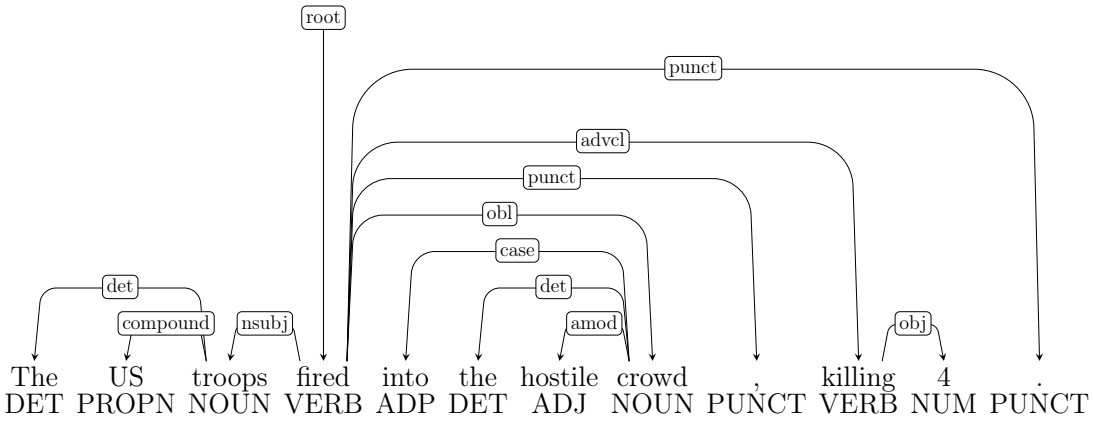

Figure 1.1: Example of dependency analysis. Sentence and analysis from the UD 2.2 English EWT treebank.

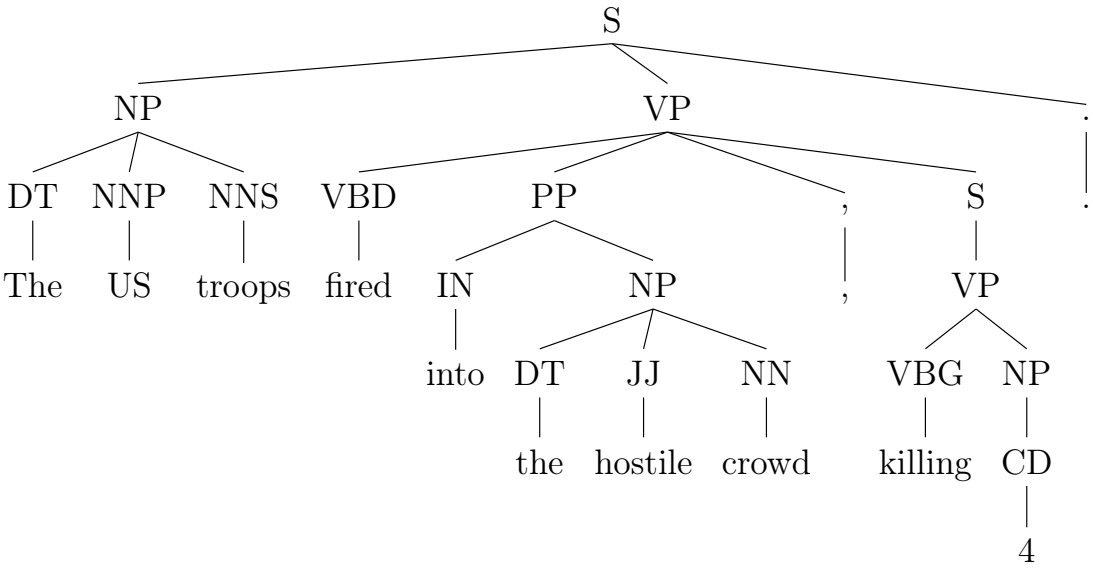

Figure 1.2: Example of phrase-structure analysis, in the style of the Penn Treebank. Sentence from the UD 2.2 English EWT treebank.

argue as well that understanding syntax could proportionate insights on human cognitive abilities or even on the structure of the mind (SAG *et al.*, 2003, pp. 9-14). Thus an automatic parser would be able to produce annotated data from (naturally occurring) text for scientific use.

On the engineering and applications side, syntactic parsing is generally considered useful for further downstream applications in natural language processing, such as machine translation (WAY, 2010, p. 551), question answering (WEBBER and WEBB, 2010, p. 635; YAO and DURME, 2014), information extraction (MAUSAM *et al.*, 2012) (CORRO and GEMULLA, 2013; GRISHMAN, 2010; MAUSAM *et al.*, 2012), and textual entailment BELTAGY *et al.* (2014). Some techniques use syntactic parsing for producing intermediate formats with more semantic content, such as logical forms, a task called *semantic parsing* (BELTAGY *et al.*, 2014; REDDY *et al.*, 2016), or more lightweight semantic formats which make predicate-argument structure, events and semantic roles more explicit[2] the work of WHITE *et al.* (2016).

One purported advantage of using syntactic representations for NLP tasks is that it allows *compositionality*, the principle that the meaning of a complex expression is determined recursively by the meaning of its parts (ABEND and RAPPOPORT, 2017, Section 6; BENDER *et al.*, 2015, Section 2). In the case of sentences, it means that there exists a sentence meaning which can be built from its words according to word meaning and to syntactical structure (SZABÓ, 2017). Although not every aspect of meaning is compositional (BENDER *et al.*, 2015, Section 2), supporters of compositional methods argue for its advantages for comprehensiveness, consistency of annotation and scalability (BENDER *et al.*, 2015, Section 4). For instance, contextual and pragmatical aspects of meaning are isolated, which would allow the

[2]For a more complete view of available formats of semantic representation, as well as motivation and discussion on goals and future research, see (ABEND and RAPPOPORT, 2017).

development of reusable task-independent methods for finding the linguistic signal[3].

It should be said that the usage of syntactic methods for downstream NLP tasks is not without its criticism: CLARK (2010, p. 352) points out that "*it is still the case that convincing evidence of the benefits of parsing for NLP applications is lacking, especially for MT* [Machine Translation] *where phrase-based models currently provide the state of the art*", even if at least for some languages this criticism may not necessarily be the case anymore (SENNRICH and HADDOW, 2015). ABEND and RAPPOPORT (2017) argue that machine learning allows mapping from text to semantic structure without worrying about syntax. A more recent challenger is Neural Machine Translation, which achieves state-of-the-art results without any kind of morphosyntactic analysis, by the usage of end-to-end learning with Deep Neural Networks (BOJAR *et al.*, 2016; JOHNSON *et al.*, 2017; WU *et al.*, 2016). Further discussion on the comparison of such methods is beyond the scope of this work. For the rest of this work, we will take the relevance of syntactic parsing for applications for granted.

Syntactic representations come in many forms, but two types are usually iden-

[3] (BENDER *et al.*, 2015) argue for these advantages in the context of *grammar-based* methods for finding linguistic structure and sentence meaning. The extension of this argument for every syntactic representation is subject to criticism, but it is not our goal to pursue a lengthy discussion on this issue, nor on the validity of this argument itself. We will only point out cases for which this general sentence meaning argument more clearly does not hold in its entirety:

1. Many syntactic parsers are designed only for the task of *robust disambiguation* syntactic parsing. This means that, for every sentence, the parser should always return one and exactly one syntactic analysis (NIVRE, 2006, pp. 1, 5-6). This should always hold no matter how strange, ambiguous or even ungrammatical the sentence may look like. Robust disambiguation is the usual scenario for Machine-Learning-based parsers, which are commonly trained and evaluated in datasets where each sentence contains exactly one analysis.

   As sentences can hold many possible readings, the disambiguation task being inserted into syntactic parsing implies that parsing does not necessarily capture exactly the linguistic signal contained in a sentence, as the parser will be required to make some kind of choice.

   Of course, downstream semantic tools may build on top of the generated syntactic representation using some kind of compositionality, but the entire linguistic signal is not entirely preserved and consistency may be reduced. Thus, robust disambiguation does not share all advantages of compositional methods claimed by BENDER *et al.* (2015).

2. In task-specific semantic parsing with syntactic methods, a successful approach is using machine learning for learning semantic representations, while keeping the syntactical ones as latent (hidden) variables in the model (ARTZI and ZETTLEMOYER, 2013; ZETTLEMOYER and COLLINS, 2005). This means that syntactic parsing is "implicitly" learned during the semantic parsing task even without explicit annotated syntactic data. Even if the resulting method includes transparent and observable syntactic analysis as well as a compositional semantic structure with clear syntax-semantics interface, the learned syntactical analysis does not necessarily generalize well for out-of-domain scenarios. Contextual and pragmatical clues may be available in the semantic data and probably are. Using the terminology of BENDER *et al.* (2015), in task-specific semantic parsing the semantic annotation corresponds to *speaker meaning*, not to *sentence meaning*, and speaker meaning is not necessarily compositional. Thus, as in the case of robust disambiguation, there is some kind of compositionality, but without a commitment to a speaker-independent sentence meaning.

tified: *phrase structure* representations and *dependency* representations, usually in the form of *trees* (RAMBOW, 2010; NIVRE, 2006, p. 10-12). Each one of these two types more conveniently represent different linguistic phenomena: respectively, syntactic constituency structure and syntactic dependency.

A phrase structure tree is a tree for which leaf nodes correspond either to words in the language or to empty strings, while internal nodes correspond to special (called *non-terminal*) symbols. This more naturally captures syntactic constituency structure, which is a recursive decomposition of a sentence in smaller pieces which have some role as a syntactic unit. Each one of these smaller pieces is called a *constituent* or *phrase* and usually receive some form of classification, according to syntactic role. In phrase structure trees, this classification occurs by the non-terminal symbols used. An example is in Figure 1.2, using the tagset of the Penn Treebank, one of the first large-scale syntactically annotated corpora (MARCUS *et al.*, 1993).

On the other hand, dependency trees are trees for which all nodes correspond to either words in the language or empty strings. This kind of representation is more commonly used to represent syntactic dependency, which are linguistic directed binary relations between words, usually identified with grammatical function. In this relation, one of the words, the *child* or *dependent*, *depends* on the second word, which is the former word's *head* or *governor*. These relations usually receive a classification according to syntactic role. An example is in Figure 1.1. This is the structure that we will use from now on, as Universal Dependencies is a dependency representation.

However, as RAMBOW (2010) emphasizes, it is possible to represent linguistic dependency structure in phrase structure trees by the structural conventions and specific non-terminal labels, as well as syntactic phrase structure can be expressed in dependency tree representations by the usage of features and relation labels.

There are many distinct methods for producing such representations. It is usual to divide two families of parsing methods: *grammar-based* and *data-driven* (NIVRE, 2006, p. 20-40; KÜBLER *et al.*, 2009, p. 7; IVANOVA, 2015, chapter 1). It must be emphasized that this division is stereotypical, in that many current methods are at least informed by both approaches. In a brief explanation, the grammar-based methods consists in developing a formal grammar with desired scope and using this grammar for attributing syntactic representations to strings. This grammar may be manually designed, learned from data or any combination of both. On the other hand, a data-driven method consists in an algorithm which generates a syntactic parser from a sample (a finite set of examples). If every example in the sample is annotated with their syntactic representation, it is an instance of *supervised learning*. Otherwise, if no example is annotated, it is *unsupervised learning*. Finally, if some are annotated and others are not, then it is an *semi-supervised learning* scenario.

Of course, one possibility for a data-driven method is generating a grammar, which would be a clear case of a hybrid method. However, the data-driven method allows for other, more general methods. An interesting work exploring the comparison between grammar-based methods and "purely" data-driven ones and how one could inform the other is the thesis of IVANOVA (2015).

In this work we will not explore any parsing method, as our methods are applicable to syntactic annotation, regardless of how it was produced. While supervised data-driven methods are currently more common[4], this is not a necessary relation.

1.3 A case study on syntactic annotation: Universal Dependencies

In this work, we focus on the Universal Dependencies (UD) annotation scheme, an initiative for cross-linguistically consistent syntactic annotation (NIVRE *et al.*, 2016). It allows comparative linguistic studies and multilingual NLP development (NIVRE *et al.*, 2016). This is a recent project building upon earlier work on creating consistent morphological and syntactic annotation for different languages, such as (universal) Stanford dependencies (DE MARNEFFE *et al.*, 2014), the universal Google dependency scheme (MCDONALD *et al.*, 2013), the Google universal part-of-speech tags (PETROV *et al.*, 2012) and the Interset interlingua for morphosyntactic sets (ZEMAN, 2008). Its current full release is version 2.2[5], it has 122 treebanks for 71 languages. It was used for the Conference on Computational on Natural Language Learning (CoNLL) 2017 and 2018 shared tasks on multilingual syntactic parsing (ZEMAN *et al.*, 2017), respectively in its 2.1 (NIVRE *et al.*, 2017) and 2.2 (NIVRE *et al.*, 2018) releases.

UD is defined by both general principles and more specific instructions for phenomena and tags. It is a dependency tree formalism aiming to capture syntactic dependency, as well as, at word level, parts-of-speech and morphological features. Thus it follows *lexicalism*, that is, the basic units of annotation are words in sentences.

UD's goal is to maximize *parallelism* between different languages. This means that similar constructions in different languages should be annotated in similar ways. However, it also follows a kind of minimalism: it avoids annotating elements which do not occur merely because they occur in other languages. If necessary, languages can refine their analyses by specifying categories sub-types. Thus, as linguistic phenomena should be annotated in similar ways across languages, the guidelines are

[4]Specially for Universal Dependencies, which is used for the Shared Tasks of the Conference on Natural Language Learning, a competition in supervised learning of syntactic dependency parsers.

[5]All quotations from the documentation in this paper refer to the 2.2 release.

explained in terms of such phenomena and motivated by many linguistic concepts which are important in defining annotation decisions, as we will see soon.

Linguistics differentiates between *content words* and *function words*. This distinction is usually identified with parts-of-speech, even if imperfectly (JEŽEK, 2016, p. 14). Function words complement the meaning of content words. For instance, they are frequently used for determining some specific kinds of semantic information, such as number, gender, case, and so on. They are closed class, that is, there is a fixed list, in a way that it is infrequent to add or remove this kind of word in a language. Examples are prepositions, conjunctions, determiners, and pronouns. As for content words, they usually have (stronger) "autonomous" semantic content, expressing entities, events, properties, among others. They are open class, in other words, new content words are frequently incorporated in a language. Examples are nouns, adjectives and verbs (JEŽEK, 2016, pp. 14-15).

In UD, dependency relations between content words have higher priority, which means that they should usually be annotated higher up in the syntactic tree. This is a consequence of the parallelism principle, as function words may vary more among languages, because they may express features of content words which in some languages are specified in morphology (NIVRE *et al.*, 2016, p. 1663). Therefore, priority to content word relations makes it more likely to find parallel syntactic structure between languages.

A second characteristic of UD is that syntactic structures are distinguished by types and this informs dependency relations labels. For UD 1, NIVRE *et al.* (2016) identifies nominals, clauses and modifier words, while in UD 2, the types identified in the index of dependency relation labels are nominals, clauses, modifier words and function words.

In order to illustrate better some linguistic aspects, we will present some examples along with documentation on how some phenomena are represented.

A simple verbal clause is represented as in Figure 1.3. The root of the entire clause is the predicate (the verb "*sinks*" in this case). Its subject (`nsubj`) "*amphibious*" is a direct dependent, as well as "*Lake*", the head of the subphrase "*Table Rock Lake*". Notice how function words (such as the determiner "*An*" and the case-marking preposition "*in*") and modifier words (such as the adjective "*amphibious*") are lower on the tree.

As for complex clauses, the head of a subordinate clause is linked as a dependent of the main clause's head. For instance, consider Figure 1.4, the complete version of the sentence in the previous example. The subphrase "*leaving 17 people dead*" is a subordinate clause of "*An amphibious "duck boat" sinks in Table Rock Lake, United States*". The dependency type `advcl` indicates that the subordinate clause is an *adverbial* clause, that is, modifies the main clause as an adverb.

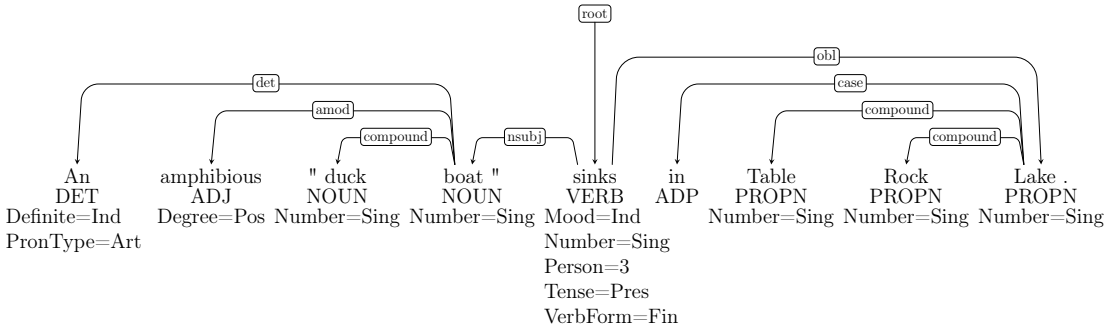

Figure 1.3: UD analysis of a verbal clause, with part-of-speech tags and morphosyntactic features. Adaptation of sentence found in Wikipedia, Current Events: `https://en.wikipedia.org/wiki/Portal:Current_events`. Original sentence under CC-BY-SA license. The analysis was simplified by omiting punctuation from the tree.

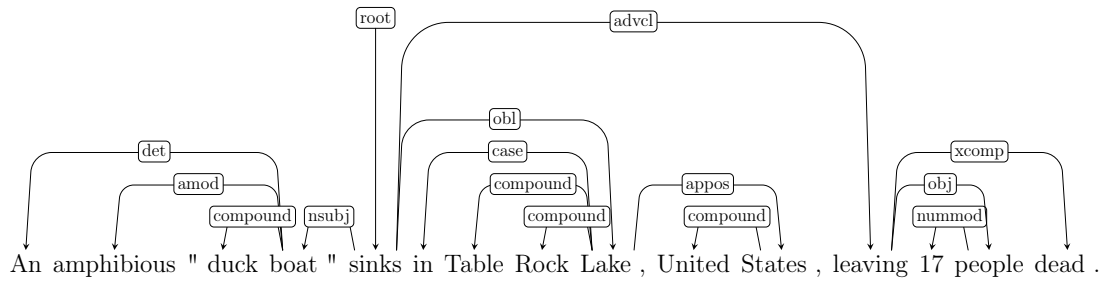

Figure 1.4: UD analysis of a verbal clause with a subordinate clause. Sentence found in Wikipedia, Current Events: `https://en.wikipedia.org/wiki/Portal:Current_events`. Original sentence under CC-BY-SA license. The analysis was simplified by omiting punctuation from the tree.

Copular clauses are represented as in Figure 1.5. The same principle is followed in that the root of the clause is the (nominal or adjective) predicate. In the example, it is a nominal predicate "*(web-based free-content encyclopedia) project*". The copular verb "*is*" is considered a function word and thus is not the root.

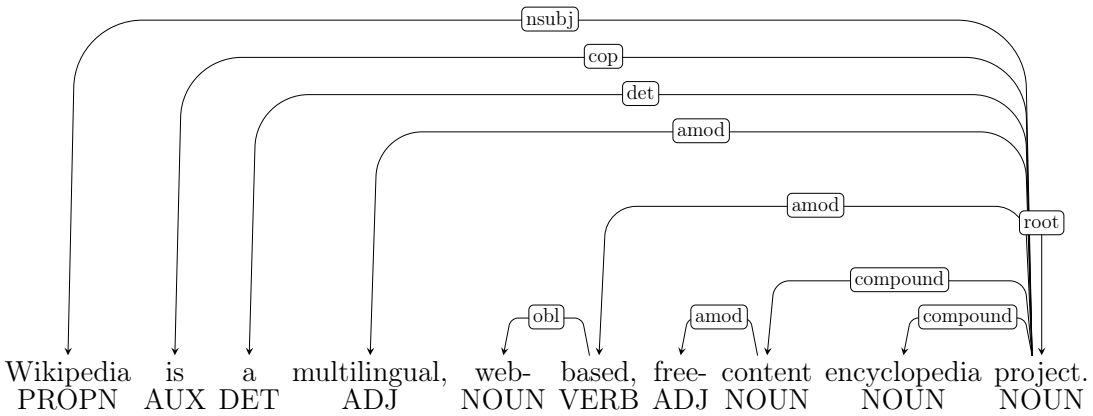

Figure 1.5: UD analysis of a copular clause, with part-of-speech tags. Adaption of sentence found in Wikipedia:About: `https://en.wikipedia.org/wiki/Wikipedia:About`. Original sentence under CC-BY-SA license.

As for coordination, an example is in Figure 1.6. Coordination traditionally presents problems to dependency representations. If either conjunct is selected to be the head, an asymmetry that arguably does not exist in language is created. Else, the coordinating conjunction could be selected as head of the structure. However, this is not possible in UD. As the coordinating conjunction is a function word, it would violate the "content word first" principle. A third alternative would be creating in the tree an empty node that does not correspond to any word, but to the conjunction itself. However, by the lexicalist hypothesis, only words occurring in the text are nodes in the tree. Besides, it is arguably a violation of minimalism, as it inserts an artificial element.

UD opts for the first alternative, by choosing the first occurring word in a conjunction to always be the head. This asymmetry is not merely an elegance problem. Consider Figures 1.7 and 1.8. By reading only the text of the sentence shown in Figure 1.7 we could expect the adjective "*lovley*" ("*lovely*") to be read as modifying only the noun "*food*", while "*fab*" ("*fabulous*") modifies only "*chips*". As for sentence of Figure 1.8, a perhaps natural reading would be the one where both "*food*" and "*chips*" are "*lovley*". However, notice how in both cases "*lovley*" is represented as a dependent of "*food*". This representation creates an ambiguity of whether the dependent word "*lovley*" modifies only the head "*food*" or is shared by the conjunct "*chips*" as well[6]. While this possibly creates problems for downstream applications,

[6]Notice that this is an issue of representing syntactic constituency structure in a dependency for-

it is a problem recognized by the documentation itself. It could be argued that this is part of a trade-off in order to guarantee the principles of lexicalism, minimalism and content-word first, seen as more important for cross-linguistic analysis.

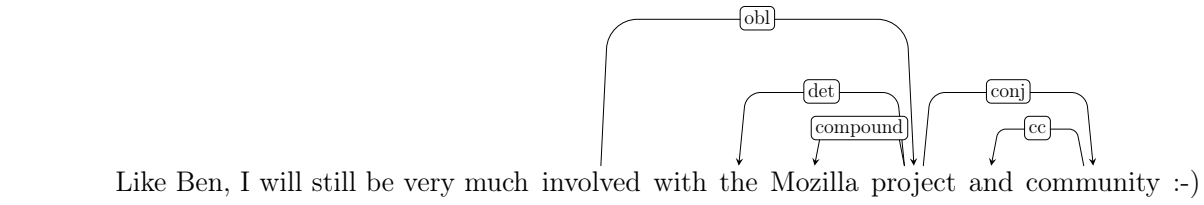

Figure 1.6: UD analysis of coordination between the nominals "*project*" and "*community*". Sentence and analysis from the UD 2.2 English EWT treebank. For readability, we show only the analysis of the subphrase "*with the Mozilla project and community*".

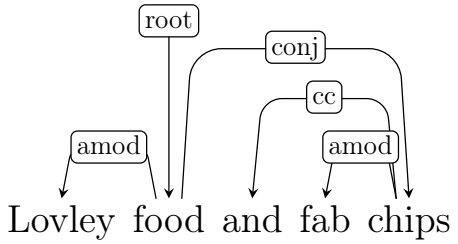

Figure 1.7: UD analysis of coordination between two nominals. Example of (arguably) non-shared dependent. Sentence and analysis from the UD 2.2 English EWT treebank.

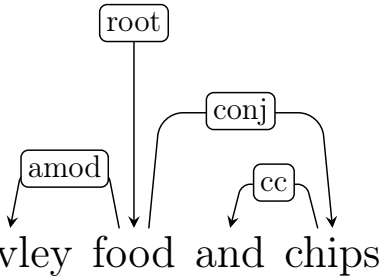

Figure 1.8: Slightly altered version of the sentence in Figure 1.7.

As for ellipsis, the annotation guideline is defined with respect to a hypothetical sentence in which the elided word is present. It seems to be motivated by the desire to disrupt the syntactic representation as little as possible, "promoting" one of its

malism, due to ambiguity between modifying the word only or the entire subphrase defined by the subtree rooted on this word. Theoretically it could be handled by relation types or word features, but this is not done. Possible reasons for this are that it could make annotation, comprehension by non-linguists or parsing more difficult.

dependents to become the head of the subtree. More specifically, the documentation states the following three rules:

1. If the elided word does not have a dependent, then nothing is done, that is, only the elided word is missing.

2. Otherwise, one of this word's dependents is promoted, taking the role of the head of the subtree. Thus, all other dependents of the elided word become dependents of the promoted word.

3. If the elided word is a predicate and the promoted word is an argument or adjunct, then a special relation type (`orphan`) is used for linking other non-functional dependents to the promoted word.

The promoted word is chosen accordingly to a specific order of dependency types (with the first one, according to word order, being chosen if there is more than one child with the best-ranked same dependency type). As for the third rule, it is motivated by not creating "very unnatural and confusing relations", according to the documentation.

See the examples in Figures 1.9, 1.10 and 1.11. Figure 1.9 has no elided word, it is an usual sentence. By the ellipsis of the word "*cigarros*", one of its direct dependents has to be promoted. As "*5*" is the only dependent, it becomes the object (`obj`) of the verb "*fuma*", creating the tree in Figure 1.10. Then, by eliding the word "*fuma*", once again we must promote one of its dependents. According to the documentation, we must promote the subject (`nsubj`). Thus, "*mulher*" is promoted an is now the target of the coordinate relation (`conj`) with the first clause. Furthermore, it must become the head of "*5*". However, as "*5*" is a non-functional dependent, a special relation (`orphan`) is used instead.

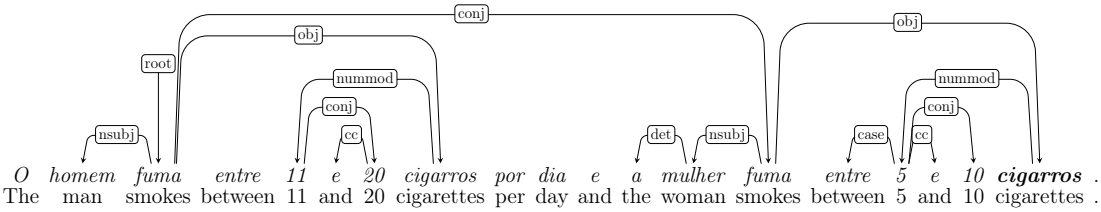

Figure 1.9: A modified version of the sentence in Figure 1.11 without any ellipsis. It modifies sentence in Figure 1.10 by inserting the highlighted word "*cigarros*".

There is ongoing work on an extension to UD called *Enhanced Dependencies* (NIVRE *et al.*, 2016, p. 1663; SCHUSTER and MANNING, 2016). In this extension, the analysis is not *tree-structured* anymore, but *graph-structured*, as in Figure 1.12. This allows more expressive representation and more directly presented

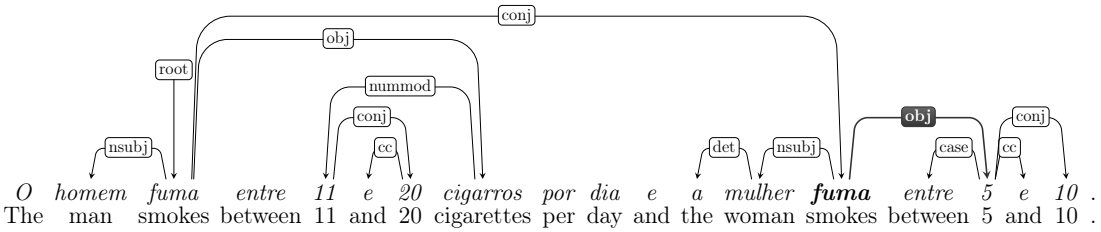

Figure 1.10: A modified version of the sentence in Figure 1.11 without the ellipsis of the verb "*fuma*".

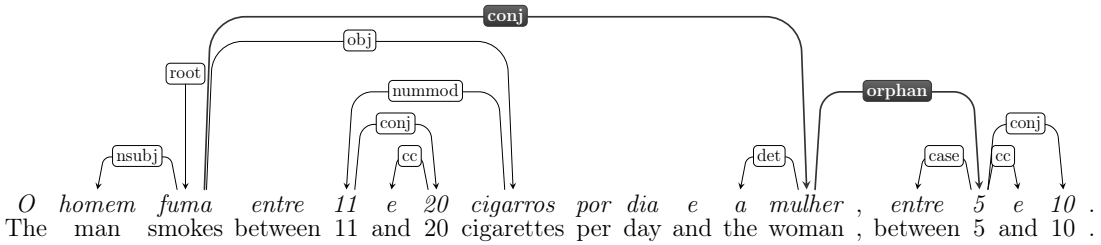

Figure 1.11: Sentence and analysis from the UD 2.2 Portuguese Bosque treebank. The `orphan` relation in highlight indicates that an elided predicate word would have both "*mulher*" and "*5*" as dependents. A word-for-word translation is accompanying the original sentence.

predicate-argument structure, which can be useful for semantic tasks. For instance, shared dependents in coordination can be explicitly marked as so. However, work in Enhanced Dependencies is still initial. Better standards are still being developed and very few treebanks contain Enhanced Dependencies annotation. Thus, we will not consider it for the rest of this work.

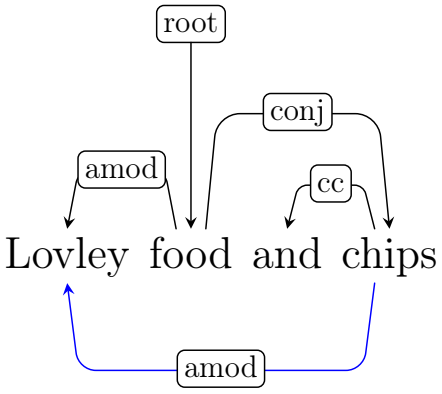

Figure 1.12: Enhanced dependencies version of the sentence in Figure 1.8. The adjectival modifier (`amod`) relation of "*lovley*" is propagated to "*chips*", meaning that it refers to it as well.

Finally, UD has been used for semantic tasks as an intermediate step (KANAYAMA and TAKEDA, 2017; REDDY *et al.*, 2017). Besides, it has been

used in the 2018 edition of Extrinsic Parser Evaluation Initiative (EPE 2018) at the Conference on Computational Natural Language Learning (CoNLL 2018), a competition to evaluate the impact of UD syntactic parsing on downstream tasks[7]. Other approaches further process UD syntactic trees in order to generate semantic representations, such as logical forms (REDDY *et al.*, 2017), graphical representations (KALOULI and CROUCH, 2018) and other predicate-argument indication and lexical semantic enhancements on top of syntactic representation (WHITE *et al.*, 2016). This shows that UD is not merely an academic proposal and suggests that UD is increasingly becoming a standard in the field.

UD is annotated in a tabular format with 10 columns called the CoNLL-U format. Each line corresponds to a word in the sentence. The column fields include word index (indicative of order in the sentence), form in which it appears, lemma (dictionary form), part-of-speech tag, morphological features, index of the head word and dependency relation to head word.

An example corresponding to the analysis in Figure 1.7 is in Figure 1.13.

```
# newdoc id = reviews-258042
# sent_id = reviews-258042-0001
# text = Lovley food and fab chips
1       Lovley  lovley  ADJ     JJ      Degree=Pos      2       amod    _       _
2       food    food    NOUN    NN      Number=Sing     0       root    _       _
3       and     and     CCONJ   CC      _       5       cc      _       _
4       fab     fab     ADJ     JJ      Degree=Pos      5       amod    _       _
5       chips   chip    NOUN    NNS     Number=Plur     2       conj    _       _
```

Figure 1.13: Tabular CoNLL-U format of the sentence in Figure 1.7.

[7]In EPE 2018, the tasks are: Biological Event Extraction, Fine-Grained Opinion Analysis and Negation Resolution.

Chapter 2

A Formal Specification For Universal Dependencies

In Computer Science, an *ontology* is a description, or representation, of knowledge about some domain of interest using some formal language, that is, being able to be read and manipulated by a machine (HITZLER *et al.*, 2010, p. 2). In this chapter, we present our ontology for the Universal Dependencies standard, the main contribution of this thesis.

2.1 Motivation

In the Introduction (Chapter 1), we asserted that building a formal specification of an annotation scheme, such as UD, is potentially useful for syntactic annotation. We highlighted three specific points: finding errors in data, helping in production and maintenance of annotated data, and being used as background knowledge by machine learning algorithms. We will discuss these specific points in later sections, but we shall now discuss the general idea in more detail.

A specific motivation for building a formal specification is *having an unambiguous language for describing something and making inferences about this description*. It is even more desirable if such inferences are *automatic*. *Description* and *inference* are basic goals of the area of *knowledge representation and reasoning* (KRR) (BRACHMAN and LEVESQUE, 2004).

In producing linguistic data, specially in a community endeavor such as UD, great effort is expended in writing a documentation in natural language, in order to guide annotators, developers and users of data following the standard. Verifying possible inconsistencies in the documentation or between examples and the documentation is a hard and non-obvious task. As the documentation is open to discussion, it is expected that it can change with time, and as it is modified by different authors,

errors may be made, possibly hard to find.

Therefore, the UD community, or at least a treebank manager, instead of simply having the documentation in natural language, could also have a formal specification. This formal specification could be automatically checked against examples in the documentation and treebanks, allowing for consistency checks between what is asserted as requirements and restrictions, and what actually occurs in data. A formal specification could be maintained by collaboration, in the same way natural language documentations are.[1]

It could be questioned why a formal specification should be built for representing annotated linguistic data. At least in some extreme version of skepticism about knowledge representation and reasoning, an argument could be formulated as following: *hand-building a formal specification requires much work from specialists in both the domain (in this case, linguistics) and formal logic. Not many people have knowledge on both areas, therefore it becomes expensive and difficult to develop such specification. Besides, explicit hand-build knowledge is frequently wrong or incomplete, as humans are not particularly good in capturing every relevant property from phenomena. Finally, formal reasoning is very slow, sometimes even undecidable. Therefore, building formal specifications should be dispensed with and only data-driven methods without explicit knowledge should be used.*

We will dispute this argument on three grounds. First, explicit knowledge is not necessarily hand-built and derived only from experts' opinions. Not only an specialist can and should use data while developing formal specifications, but there are as well data-driven methods for producing explicit knowledge. Indeed, the areas of statistical relational learning (SRL) and inductive logic programming (ILP) are about such methods (GETOOR and TASKAR, 2007; MUGGLETON *et al.*, 2012).

Second, the annotation standard is already created by specialists in computational linguistics. The success of Universal Dependencies builds on it being designed by a community of specialists in this area. Using the correct tools, a deep knowledge of logic is not strictly necessary. Semantic technologies and declarative methods are built in a way so that it is not necessary to understand implementation details in order to use them. Indeed, this is one of our motivations in using OWL: the Semantic Web community has largely adopted it and there are many out-of-the-box tools, at least in comparison to less used formalisms. Besides, it is unlikely that in a community effort there is no collaborator with enough knowledge on logic in order to maintain an ontology. A basic knowledge of logic is usual knowledge for both computer scientists and linguists with some background in Semantics.

Thirdly, computation speed is not necessarily a problem. While it certainly make

[1] This is greatly facilitated by code repositories in the Web, such as `github`, and version control tools, such as `git`.

some use cases unfeasible, such as using unrestricted reasoning in application runtime for end users, it is not a hard problem for other uses. For instance, we assert that one practical use for our ontology is treebank maintenance. This is hardly a task in which time bounds are strict. Finally, undecidability is not a problem when using a representation method with guarantees on decidability, which is the case for description logics, for instance, which we use.

Let us now briefly remind the specific advantages highlighted before and where they will be discussed. A formal specification produces almost directly a method for automatically detecting errors in data. Given a reasoner, a tool that can produce inferences from a specific formal language, finding errors in data is possible by verifying the consistency of the data with the formal specification. This is the method that we will use and will be explained in more detail in this chapter, Section 2.4.

A formal specification is also a valuable resource for producing and maintaining annotated linguistic data. This method for finding errors can also be used for reducing the cost of creating new datasets. Besides, it also improves methods for querying data, due to hierarchies and relations in ontology, which allow for inference. Both issues will be addressed in Chapter 4.

Finally, a formal specification could potentially be used by machine learning methods in order to learn with greater precision with less data. Unfortunately this is not an idea which we will be able to experiment and present in much detail, but some discussion is in order in Section 5.2, on future work.

2.2 Formal foundations and language

There are many possible ways of formalizing an annotation standard or, for that matter, for representing knowledge in general. Many formal languages (with well-defined semantics) exist which could be used, with differences in their expressive power and computational complexity of reasoning. In practice, some scientific communities developed clear standards for representation languages, some of which have wider adoption.

In this work, we use OWL 2 DL: the Web Ontology Language, Description Logics semantics (OWL WORKING GROUP, 2009). OWL is a current standard for *Semantic Web* communities and industries. It evolved with the goal of representing semantic content in the World Wide Web by common practices and technologies, in addition to supporting logical inference. It is currently in the OWL 2 version.

OWL 2 offers two different semantics: OWL 2 RDF-Based Semantics, also known as OWL 2 Full (CAROLL *et al.*, 2012); and OWL 2 Direct Semantics, also known as Description Logic semantics, or OWL 2 DL (MOTIK *et al.*, 2012a) (which we may

refer here merely by OWL-DL). OWL-DL is designed to keep decidability, while OWL Full is undecidable (HITZLER *et al.*, 2009). This means that, for OWL-DL but not for OWL Full, in principle any reasonable task can be correctly solved in finite time, disregarding resource restrictions (time and space). Description logics are a family of logics regarding the notions of classes (unary predicates, or concepts) and roles (binary predicates). Many description logics are decidable subsets of first-order logic. This is the case for the description logic $\mathcal{SROIQ}$, on which OWL-DL is based (HORROCKS *et al.*, 2006; RUDOLPH, 2011).

Our choice of OWL 2 DL is motivated by: i) its wide usage by the Semantic Web community, which means that there are supported out-of-the-shelf tools for it; ii) its guarantee of decidability, a property that first-order logic does not have; iii) expressivity apparently sufficient for relevant constructions in UD, although the limit of this hypothesis can only be discovered in practice.

2.3 An Ontology for Universal Dependencies

2.3.1 Concepts in the Universal Dependencies annotation guidelines

Even if the UD project does not present itself as a grammar or linguistic theory (NIVRE, 2015, pp. 2-3), its guidelines make reference to many linguistic concepts in such a way that understanding them is necessary not only for justifying design decisions, but also for applying them. Consider, for instance, the definition of `acl` (clausal modifier of noun):

> `acl` stands for finite and non-finite clauses that modify a nominal. The `acl` relation contrasts with the `advcl` relation, which is used for adverbial clauses that modify a predicate. The head of the `acl` relation is the noun that is modified, and the dependent is the head of the clause that modifies the noun.

Even though *clause* and *nominal* are usual concepts in syntax, they are not explicitly stated in the UD CoNLL-U format, and thus are not directly available for a tool without background knowledge on UD.

Some concepts in UD's documentation are: *content word*, *function word*, *core arguments*, *oblique modifiers*, *nominal phrase*, *clause*, *modifier word*, *promotion*, among others. Although they are explained in the documentation, the target audience is researchers and annotators, and thus these concepts are not explicitly present in annotated sentence files.

2.3.2 Ontology summary

Our ontology is divided in different files according to the architecture in Figure 2.1. It is built on top of the OLiA `system` ontology[2] (CHIARCOS and SUKHAREVA, 2015).

ud-annotation-model contains the classes and relations necessary for describing the contents of a UD analysis in the CoNLL-U format.

ud-structure contains axioms regarding the structure of dependency tree, such as the requirement that each sentence has exactly one `root` (in its dependency analysis).

ud-theory contains concepts referred by the documentation. This is where the relation between the concepts and the annotated content is made, as well as restrictions contained in the documentation are declared. This is our main ontology and it will be used in the experiments (Section 3).

ud-svalidations contains restrictions contained in the official *syntactic validation* [3] from the UD project. While many syntactic validation tests are already contained in `ud-theory`, some do not seem to be implied by the documentation and are not considered to mean that the data is invalid. These are included only in this file. Our main goal in developing this ontology was testing if OWL-DL (and our formalism) was expressive enough to reproduce every syntactic validation.

ud-theory-and-svalidations The union between the last two ontologies.

Table 2.1: UD Theory ontology metrics and expressivity used.

Classes	232
Object properties	53
Data properties	6
Individuals	124
Logical axioms	532
Declaration axioms	418
Annotation axioms	39
DL expressivity	$\mathcal{ALCROIQ}(\mathcal{D})$

[2] Available at `http://purl.org/olia/system.owl`.

[3] Currently it covers all restrictions except the ones regarding morphological features.

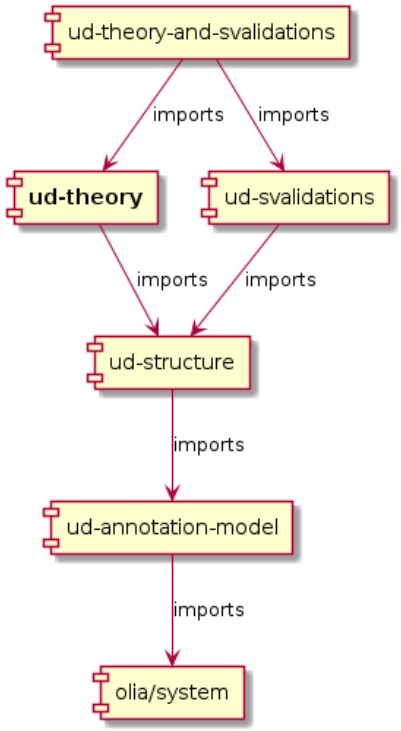

Figure 2.1: Architecture of the UD Theory ontology. Each node is an OWL file.

2.3.3 Modeling

Following the OLiA `system`, words and sentences are represented as instances of subclasses of `UnitOfAnnotation`; lemmas, morphological features[4] and parts-of-speech as instances of subclasses of `Feature`. However dependency relations are doubly represented: not only reified as subclasses of `Relation`, as in OLiA `system`, but also are represented as OWL relations on words[5]. Dependency relations are organized in the following way: there is a general `isDepOf` property, with subproperties such as `isDepOf:nsubj` or `isDepOf:obj`. There are some sub-types of dependency relations, such as `isDepOf:flat:name`. During our conversion of CoNLL-U files (discussed in Section 2.4), we produce some sub-type extensions of this kind on run-time if they occur in the data.

Our main concepts are in `ud-theory.owl` (Figure 2.2). Most of our restrictions are asserted as axioms on these classes.

Many of the restrictions we formalize regard the structural aspect of UD.[6] According to the UD documentation, languages structurally involve three things: nomi-

[4]They are represented but currently there are no axioms added in order to constrain or make inference from them.

[5]This was done in order to implement restrictions regarding word order in the case of `conj` (copula), `appos` (appositive), `flat` and `fixed` relations, as we will discuss in the end of this section.

[6]Explained on `http://universaldependencies.org/u/overview/syntax.html#a-mixed-functional-structural-system`

nal phrases, clauses headed by a predicate and "miscellaneous other kinds of modifier words". Motivated by this assertion and by the tabular classification of dependency relations according to these concepts[7], we consider that words are a disjunction between the classes `NominalHead`, `ClauseHead`, `ModifierWord` and `FunctionWord`. That is, we assert that `Word` $\equiv$ (`NominalHead` $\sqcup$ `ClauseHead` $\sqcup$ `ModifierWord` $\sqcup$ `FunctionWord`). These classes are not disjoint, as can be noticed from nonverbal clauses / copular constructions: the root word, a nominal predicate, is both a `NominalHead` and a `ClauseHead`.

Important restrictions are that `Word` is the disjoint union of `ContentWord` and `FunctionWord` and that `ClauseHead` $\sqcap$ (`ModifierWord` $\sqcup$ `NominalHead`) is a subclass of `NonverbalClauseHead`[8].

Another interesting example is that `FunctionWords` have no dependents except if the dependent relation is `advcl`, `advmod`, `amod`, `cc`, `conj` or `fixed`, or in cases of ellipsis, covering the four possible exceptions for function words not taking dependents: multiword function words, coordinated function words, function word modifiers and promotion by head elision. Currently this is an axiom which is not very useful for finding errors automatically, as there are no restrictions (such as negation or cardinality constraints) on ellipses, but it could be used for querying.

Such concepts are linked to the "concrete" annotation labels by specific axioms. For instance, we assert that words with an `advcl` (adverbial clause) dependency relation are instances of `ObliqueModifier` and dependents of a `ClauseHead`. None of our axioms make reference to lemmas or word forms. Therefore, all of our inferences and constraints are about delexicalized analyses.

Another issue which deserves attention is the difficulty in encoding the restriction that some dependency relations have a specific direction. For instance, `fixed` is a relation meaning that two words form a specific fixed multiword expression. One example is "*of course*", which forms a expression with specific meaning. In this case, in UD, words should be encoded in a structure where the first word in the expression is the head and all others are direct dependents of the first with the `fixed` relation. In our example, "*course*" would be a dependent of "*of*". A corollary is that `fixed` should never occur from the right to the left, that is, from a word that occurs later to a word that occurs earlier. We modelled word order using an object property called `nextWord`. Thus (a, b) : `nextWord` holds if and only if the word right after b is a. We also define a property called `nextWordTrans`, which is meant to be the transitive closure of `nextWord`. In order to encode the restriction on

[7] `http://universaldependencies.org/u/dep/index.html`

[8] A more problematic axiom currently present in the ontology is that every `NonverbalClauseHead` does not have a `VERB` part-of-speech, because it invalidates many acceptable annotations of non-finite verb forms, such as participles and infinitives. It generates some of our false positives in the experiment of Section 3.1.1.

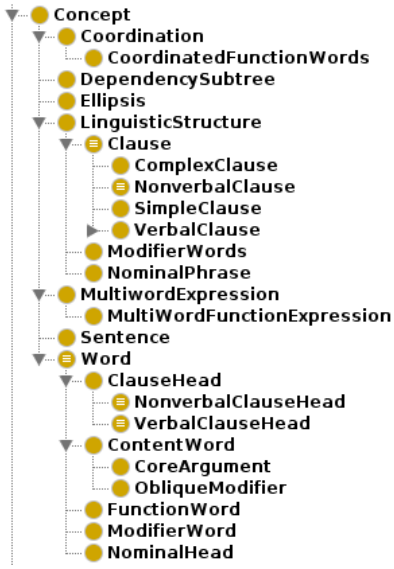

Figure 2.2: Concepts in `ud-theory`.

right-headed `fixed` relations, we assert the axiom that `nextWordTrans` is disjoint with the property corresponding to the `fixed` relation (`isDepOf:fixed`). However, this prohibits specifying in OWL-DL that `nextWordTrans` is a transitive relation, because any object relation which is disjoint with another is "non-simple", and in $\mathcal{SROIQ}$ only simple relations may be asserted as transitive (HORROCKS *et al.*, 2006, p. 59; MOTIK *et al.*, 2012b). A practical solution to this issue is guaranteeing transitivity in the ABox, that is, any representation of a sentence used with the ontology should have a `nextWordTrans` that is a transitive close of `nextWord`. We do this in our validation tool, as we will see soon.

Finally, an important construction in the guideline which we did not capture fully is ellipsis. As we presented in Section 1.3, UD captures guidelines on how to annotate ellipsis with respect to a hypothetical sentence in which the elided word is present. However, elided words are not annotated, making it hard to verify directly if the method was followed.[9] This creates some difficulties for our method, as it is expected that ellipsis are a important cause to exceptions to general classifications such as "content word" and "function words". We noticed this in practice. On the one hand, this implies that annotating every elided word would make it easier to verify if the annotation is consistent, as well as making the analysis more informative, for instance for downstream semantic tasks, or linguistic studies. On the other hand, annotating elided words could make the annotating task harder, as there may appear sentences for which it is not obvious whether a structure is or not elliptical. This ambiguity in annotation may hurt annotation consistency.

[9]Enhanced dependencies are an exception: elided predicates are annotated as special null nodes. However, other elided elements are not annotated, such as nominals.

2.3.4 Documentation inconsistencies

During the formalization, some problems in the documentation itself were found. One of them regards the relation `appos` (appositional modifier). According to the documentation itself:

> An appositional modifier of a noun is a nominal immediately following the first noun that serves to define, modify, name, or describe that noun.

Besides, it is clearly located in the the index of dependency relations as a relation between a nominal dependent and a nominal head. However, it contained the assertion (recently removed) that in some cases an `appos` relation between a nominal and a clausal dependent was possible, such as when describing facts or events. This is a clear contradiction with the dependency relation definition and UD's structural classifications.

Another problem found in the documentation is an imprecision regarding `aux` (auxiliaries), a relation between a function word (an auxiliary, such as an auxiliary verb) and a clause. The documentation states that

> An `aux` (auxiliary) of a clause is a function word associated with a verbal predicate that expresses categories such as tense, mood, aspect, voice or evidentiality.

That is, it restricts the relation to *verbal* predicates. This is not only an unnecessary constraint, but also the documentation itself contains a counter-example: "She has been happy."[10]. Here, "happy" is an adverbial predicate, and therefore non-verbal.

2.4 Validating UD CoNLL-U files

Our validation process consists in transforming a CoNLL-U file with an annotated sentence into an OWL file and verifying joint consistency with the ontology. That is, let S be the representation of the annotated sentence and O be the ontology. We validate the sentence by testing whether $S, O \vDash \bot$. In the language of Description Logic, this is the same as testing whether $Thing \sqsubset Nothing$, where $Thing$ is the top class (encompassing all things) and $Nothing$ is the empty class. It should be noted as well that S, the representation of the annotated sentence, is always limited to the assertion component (also called ABox) of the ontology, that is, it contains only assertions about individuals, such as relations between then or individual membership to classes, not terminological knowledge (such as subclass relations).

[10]Example 77 in `http://universaldependencies.org/u/overview/simple-syntax.html`

Our pipeline for validating UD CoNLL-U files is modularized in 3 main components:

1. Conversion from CoNLL-U file to RDF: for this we use the CL-CONLLU library for Common Lisp (MUNIZ *et al.*, 2017), which reads CoNLL-U files and has a feature for writing it in a RDF format. It is a "shallow" structure in the sense that it is simply a remapping of the file in triples structure.[11]

2. Preprocessing RDF: it is necessary to adapt the RDF format to another one which makes the sentence a compatible ABOX for our ontology. We do this by using SPARQL Update operations. One example of such processing is constructing the transitive closure of the relation `nextWord` (for word order in a sentence).

3. Reasoning: in this step, we pass the processed RDF and the ontology to the reasoner and check for joint consistency. In case an inconsistency is found, an explanation is returned. We use the HermiT reasoner (GLIMM *et al.*, 2014), by the interface provided by the BUNDLE tool (RIGUZZI *et al.*, 2013).

BUNDLE uses the explanation module of OWL API, which uses justifications: minimal sets of axioms which are sufficient for proving the entailment (HORRIDGE and BECHHOFER, 2011). In particular, it means that for large axioms, the entire axiom is presented as part of the explanation, instead of making clear only the part of the axiom relevant for entailment (HORRIDGE *et al.*, 2008). Of course, it also means that there is no derivation tree, which could give a better view on the entailment.

A diagram of our pipeline is in Figure 2.3.

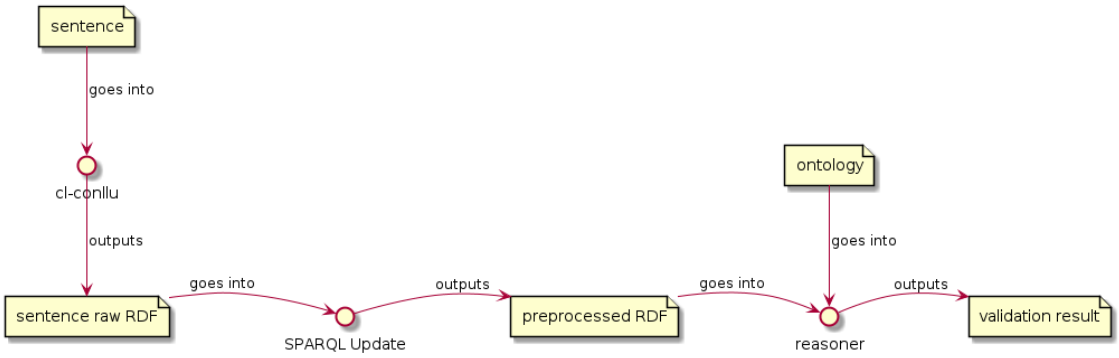

Figure 2.3: Validation pipeline.

[11]This output is similar to the of one (CHIARCOS and FÄTH, 2017), although slightly less shallow, as morphological features are parsed in individual triples.

We will illustrate the output on two invalid analyses. Consider the sentences in Figures 2.4 and 2.5. In Figures 2.6 and 2.7 we present the validation output for them, respectively. As the validator returned an explanation for each one of them, they are both wrong.

Notice that in the sentence of Figure 2.4, the `appos` (appositive) relation is headed by a closing parenthesis and has as target the word "*Breivik*" (a proper noun). For this sentence, our method found the explanation in Figure 2.6. It points out that the word 25 ("*Breivik*") comes before word 30 (the closing parenthesis, *")"*), but that an `appos` relation can never occur to a word that comes after the dependent word. This is encoded by the disjointness between properties `nextWordTrans` and `appos`.

Another possible explanation, not found in this case, is that a punctuation mark (part-of-speech `PUNCT`) should never have a dependent. The explanation module from OWL API can produce many justifications. Currently we return only one explanation, but this is easily changed by modifying the call to BUNDLE. Unfortunately, OWL API explanations are often hard to understand and may contain many superfluous parts and redundancy (HORRIDGE, 2011). Thus, reading multiple explanations is very time-consuming. A possible solution left for future work is using methods which produce more concise explanations, such as the ones proposed by (HORRIDGE *et al.*, 2008).

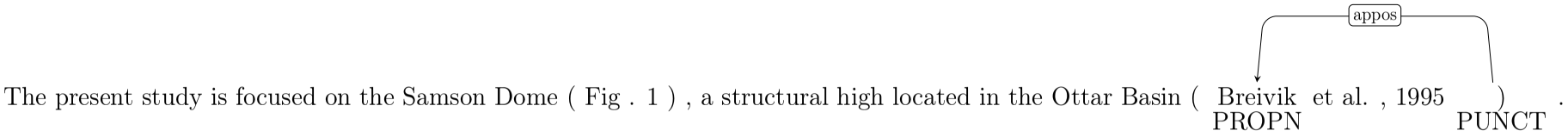

Figure 2.4: Example sentence with an error on the direction of `appos` relation.

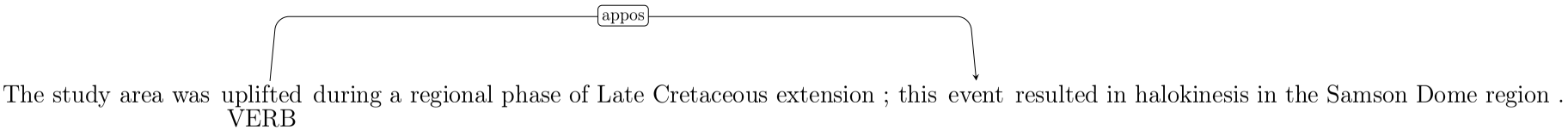

Figure 2.5: Example sentence with an error on a `appos` relation headed by a `VERB`, not by a nominal.

Explain unsatisfiability of owl:Thing
Axiom: Thing SubClassOf Nothing

Explanation(s):
1) nextWordTrans DisjointWith appos
sentence-4-25 nextWordTrans sentence-4-30
sentence-4-25 appos sentence-4-30

Figure 2.6: Explanation automatically given to analysis 2.4.

Explain unsatisfiability of owl:Thing
Axiom: Thing SubClassOf Nothing

Explanation(s):
1) sentence-2-5 hasUniversalPartOfSpeech VERB
NonverbalClauseHead EquivalentTo ClauseHead and (hasUniversalPartOfSpeech some (not (VERB_CLASS)))
VERB_CLASS EquivalentTo VERB
Relation EquivalentTo (hasSource min 1 Thing) or (hasTarget min 1 Thing)
hasUniversalPartOfSpeech some VERB_CLASS SubClassOf ClauseHead
sentence-2-16.edge hasSource sentence-2-5
Functional: hasUniversalPartOfSpeech
ClauseHead and (ModifierWord or NominalHead) SubClassOf NonverbalClauseHead
sentence-2-16.edge Type appos
acl or amod or appos or case or clf or det or nmod or nummod SubClassOf hasSource some NominalHead
Relation SubClassOf hasSource exactly 1 Thing

Figure 2.7: Explanation automatically given to analysis 2.5.

As for Figure 2.5, in short the problem is that the `appos` relation is headed by a `VERB`. Its justification in Figure 2.7 stands for the following argument : The edge of word 16 ("*event*") is an `appos`, and has as source the word 5 ("*uplifted*"). Moreover, the head of an `appos` relation is always a `NominalHead`. Thus, the word 5 is a `NominalHead`. On the other hand, the part-of-speech of word 5 is `VERB`. Every word with part-of-speech `VERB` is a `ClauseHead`. However, any word which is both a `ClauseHead` and a `NominalHead` is a `NonverbalClauseHead`. Finally, a `NonverbalClauseHead` always has a part-of-speech different than `VERB`. This is

absurd, because word 5 is a `VERB` and the part-of-speech is unique (it is a functional relation). Thus, the analysis is invalid.

Unfortunately, current explanations are hard to read. An alternative to reading these explanations is pinpointing which words and which values are problematic in an analysis. Another possible improvement is using laconic justifications (HORRIDGE *et al.*, 2008). We leave to future work improving the readability of explanations.

Chapter 3

Experimental Evaluation

3.1 Experiments

3.1.1 Finding "incorrect" analyses

Although we propose an ontology for finding errors not only in automatically produced UD analyses, but also in treebanks, and many officially released treebanks are semi-automatic conversions (and thus prone to errors), it is necessary to use data in order to have a quantitative assessment of the resource.

A possible method for evaluating our ontology, given a treebank, consists of the following: partitioning the treebank in training and test subsets, training a parser in the training set, then running the parser on the test set, validating the predicted analyses with our reasoner and measuring if the predicted analysis is "incorrect" given that it was considered invalid by the reasoner. Here, we considered a predicted analysis incorrect if it is not an exact match with the test analysis: two analyses are an exact match if and only if, for every word, its head and dependency relation to head are the same in both analyses.[1]

Thus, here we use the following definitions:

precision is the fraction of sentences for which the predicted analysis is not an exact match with the test analysis among the sentences for which the predicted analysis was considered wrong by our validator

recall is the fraction of sentences whose predicted analysis was considered wrong by validator, among all sentences whose predicted analysis is not an exact match with the test analysis

For parsing, we used UDPipe (STRAKA and STRAKOVÁ, 2017).[2] We present

[1] We measure exact matches by ignoring punctuation tokens and dependency relation subtypes.

[2] We used the swap transition system, with static lazy oracle and 20 iterations for parsing training. Otherwise we used default configurations. We did not do tokenization or tagging.

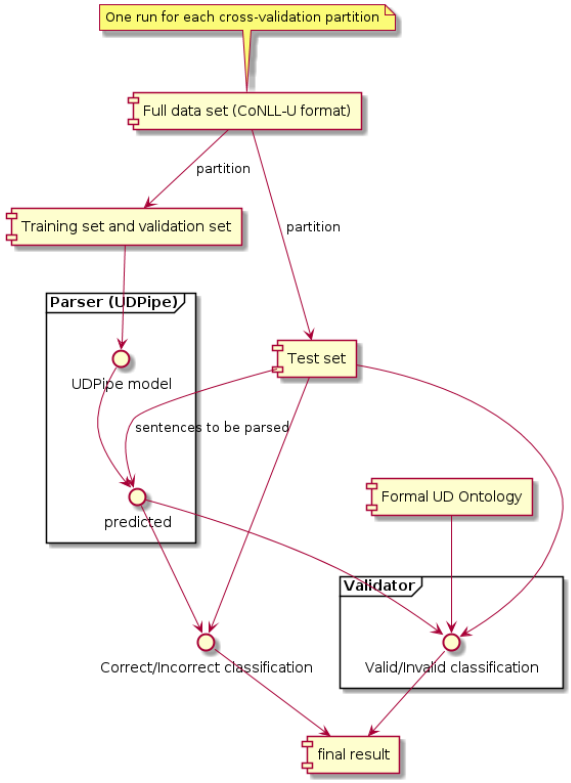

Figure 3.1: Pipeline for experiment described in Section 3.1.1.

the workflow of our experiment in Figure 3.1.

We follow this methodology for the following datasets: UD English EWT, UD French GSD, UD German GSD, UD Portuguese Bosque, and UD Spanish AnCora.[3]

If our modeling is reasonable and the treebanks sufficiently error-free, we would expect:

- precision to be high, as invalid analyses would be incorrect and therefore not part of the gold standard treebank;[4]
- recall to be possibly low, as an analysis being valid only makes it *possible*, but not necessarily correct *in the context* and with regard to words used. More precisely, an analysis being valid ideally means at most that there could possibly exist a sentence in some language with such analysis as correct, not that this analysis is correct for any sentence.

Our results are in Table 3.1.

[3] All available on the project webpage: `http://universaldependencies.org/`.

[4] Of course, it is expected that most of the predicted analyses are not exact matches, so high precision means it should be considerably higher than random choice.

3.1.2 Error analysis and discussion

As expected, for French GSD, German GSD and Portuguese Bosque, precision was over 90%, while it was almost at this value for Spanish AnCora. However, results were much poorer for English EWT, motivating further inspection. As for recall, it was around expected for French GSD, German GSD and Portuguese Bosque, but a bit higher for English EWT and surprisingly high for Spanish AnCora, which even had an especially low number of exact matches, without a big impact on precision.

On the English EWT treebank, one of the main responsible for false positives is the word "*not*", such as in Figure 3.2. In this corpus, the annotation decision is that "not" should have `PART` (particle) as part-of-speech, with a dependency relation `advmod` (adverbial modifier). However, to our ontology, every `advmod` is an `ObliqueModifier`, which is a `ContentWord`. However, every `PART` is a `FunctionWord` and, finally, `FunctionWord` and `ContentWord` are disjoint concepts, raising a contradiction in this case. This may raise some doubts about the annotation decision for this treebank, or even about the concepts themselves. Except in some corner cases (such as ellipsis, where a function word may be promoted), it would seem reasonable to assume that function words and content words are disjoint concepts. This is in incompatible with "*not*" being annotated as both `PART` and `advmod`. It is not entirely clear why it could not be annotated as an `ADV` (adverb), exactly as "*nowhere*", "*never*", particularly considering that the criteria for being `PART` is by exclusion, that is, not satisfying the definition of any other part of speech. Of course, the complete reasoning raised by this problem allows many other possibilities. One of them is allowing `advmod` for function words as well. However, this would break down the structural classification of the dependency relations, as it would collapse one dependency relation for both function words and modifier words. A second one is not using `advmod` in this case. However, it is a modification of a predicate, apparently satisfying the definition. On the other hand, this also points out to the lack of a dependency relation between function word and predicate that would be adequate for this specific case.

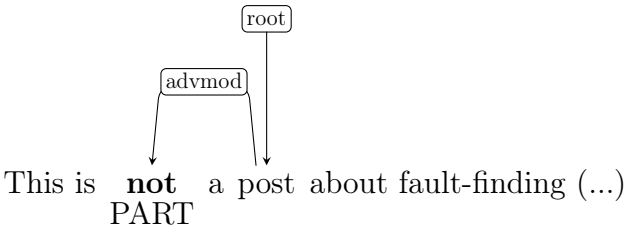

Figure 3.2: Example of "not" in a sentence of the UD EWT English corpus.

There are other interesting cases. One of them is the occurrence of `ADP` (adposi-

tion) words with an `obl` (oblique nominal) dependency relation to their heads, such as in Figures 3.3 and 3.4. Our ontology finds problems in this construction due to the assertion that every `obl` is an `ObliqueModifier`, and therefore an `ContentWord`, as well as the aforementioned disjointedness between `ContentWord` and `FunctionWord`. This annotation could be defended in terms of ellipsis: for instance, in Figure 3.3, it can be argued that "*at*" depends on a missing word after it ("*military*"), which would be an `obl` of "*scoff*". By ellipsis and promotion, "*at*" becomes an `obl`. On the other hand, in Figure 3.4, this is much harder to defend, as there is clearly no ellipsis.[5]

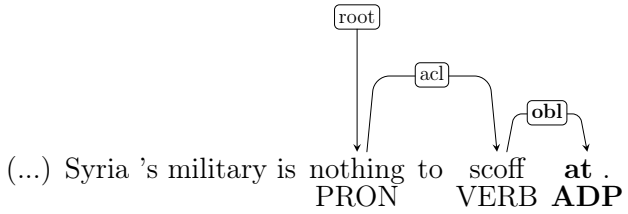

Figure 3.3: Example of an `ADP obl` word in a sentence of the UD EWT English corpus.

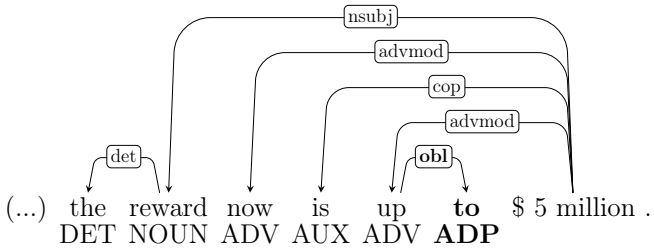

Figure 3.4: Example of an `ADP obl` word in a sentence of the UD EWT English corpus.

Inspecting the false positive of the other treebanks, we can find both modeling errors and dataset errors. For instance, for German GSD, while we found many cases of "*nicht*" with the exact same problem as "*not*" in English EWT, there were other more clear errors, such as in Figure 3.5. Less clear cases were found, such as the usage of part-of-speech `DET` for relative pronouns (`PronType=Rel`).

By inspecting Portuguese Bosque results, we found out another imprecision in the documentation that creates an excessively strong requirement. It states that:

> A nonverbal predicate (nominal or adjective) takes a single argument with the `nsubj` relation.

[5]This suggests that it would be useful if ellipses were explicitly annotated somewhere in UD, as it would allow automatic verification. While *enhanced dependencies* seem to make a move in this direction by adding special null nodes of elided predicates, nominal ellipses (for instance) are still "invisible".

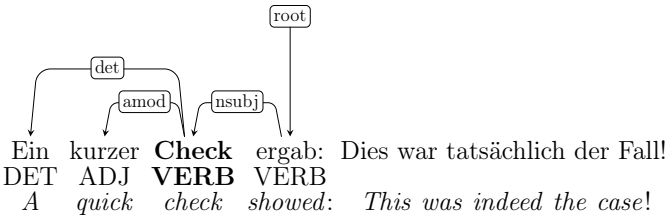

Figure 3.5: Partial analysis for a sentence of German GSD with a part-of-speech error.

However, it seems perfectly reasonable to have `csubj` (clausal subjects) in copular sentences. Indeed, consider Figure 3.6.

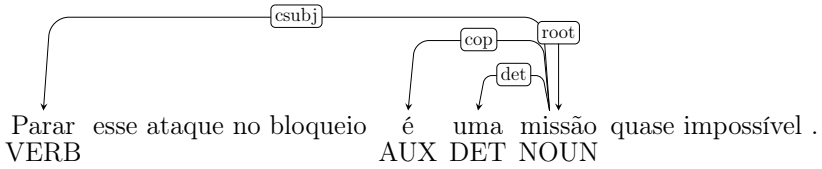

Figure 3.6: Sentence of UD Bosque Portuguese corpus with clausal subject (`csubj`) in nonverbal clause. It could be translated as "*Stopping this attack in blockage is an almost impossible mission.*"

Another problem in our ontology is regarding the morphological feature `VerbForm`. As we have not yet modelled morphological features, there are no axioms that consider them. This creates problems about forms such as infinitives, participles, as they allow verbs to occur in contexts similar to other parts-of-speech (such as nouns, adjectives and adverbs). Thus, many of our axioms regarding verbal clauses are too restrictive, and we found counterexamples in different corpora, such as EWT English and Bosque Portuguese.

Table 3.1: Results of the experiment described in Section 3.1.1. It was run in 25 folds cross validation. Values in the columns marked with an * are averages on every fold, with standard deviation in parentheses.

Dataset	Precision*	Recall*	Exact matches*	*#sentences*
UD English EWT	76.53%(±02.85%)	29.62%(±01.67%)	45.65%(±01.44%)	16621
UD French GSD	90.26%(±03.37%)	14.87%(±01.47%)	27.92%(±01.82%)	16448
UD German GSD	90.19%(±03.15%)	20.23%(±01.94%)	28.47%(±01.28%)	15590
UD Portuguese Bosque	90.80%(±04.29%)	19.79%(±02.44%)	28.84%(±02.43%)	9366
UD Spanish AnCora	89.95%(±01.61%)	53.83%(±03.46%)	18.36%(±01.59%)	17680

3.1.3 Comparison with Udapi

Udapi is a framework providing an API for using UD, implemented in different languages, but best supported in its Python implementation (POPEL *et al.*, 2017).

Udapi offers functionalities such as visualizing syntactic trees, conversion between data formats and querying. Among these functionalities, Udapi includes a validation module, in order to capture possible annotation errors. This validation module is a improved version of a set of queries officially maintained by the UD Project (about which we will have more to say in Section 4.1). While our ontology can also be seen as an improved version of this set of queries, as it covers these restrictions, Udapi does that not by a declarative method, but using a programming language (in this case, Python).

In order to have an assessment of our method, we will reproduce the same experiment for Udapi, instead of our validator. It should be noted a priori that Udapi has two advantages over our method. Firstly, it uses the expressivity of a full programming language (Python), and therefore is less restricted than our ontology. This implies that it is possible that Udapi includes tests which can not be replicated by our method. Secondly, as it is a programming language, there is a explicit control flow, and thus it can be faster.[6] Our method, on the other hand, is a declarative implementation in the form of concepts and restrictions on them. This has advantages typically associated to Semantic Web or Knowledge Representation methods, such as reusability and transparency (we will discuss more about Semantic Web in NLP in Section 4.3). In any case, the evaluation is useful as it compares the ontology to an existing approach for one of its possible tasks, besides being able to indicate in which ways our ontology is currently lacking.

Table 3.2: Results of the Udapi experiment. It was run in 25 folds cross validation. Values in the columns marked with an * are averages on every fold, with standard deviation in parentheses.

Dataset	Precision*	Recall*	Exact matches*	*#sentences*
UD English EWT	66.64%(±02.21%)	63.88%(±01.72%)	45.65%(±01.44%)	16621
UD French GSD	77.92%(±02.33%)	59.19%(±02.49%)	27.92%(±01.82%)	16448
UD German GSD	83.92%(±02.60%)	30.61%(±02.17%)	28.47%(±01.28%)	15590
UD Portuguese Bosque	97.24%(±02.57%)	15.15%(±02.43%)	28.84%(±02.43%)	9366
UD Spanish AnCora	90.09%(±01.16%)	57.86%(±02.32%)	18.36%(±01.59%)	17680

Results for Udapi are in Table 3.2. In order to compare more easily to results of our ontology on Table 3.1, we also present in Table 3.3 the difference between the two on precision, recall, and also on F1 score (the harmonic mean between these two values). We test the significance of the differences by paired Student's t-tests. While we consider the significance levels 0.05, 0.01, and 0.001, every value was significance to a point of 0.001, with the exception of precision in Spanish AnCora, which was insignificant even by the level of 0.05%.

[6]For example, the property of non-projectivity (that can be possessed by edge) is implemented as a function in a faster way than would be possible in an ontology, by exploiting data structures on descendants and counting elements.

Table 3.3: Difference between ontology-based validator and Udapi in each dataset, in percentage points. Values marked with $\star$ mean that the difference is significant by a level of 0.001. Significance is measured by paired Student's t-tests.

Dataset	ΔPrecision	ΔRecall	$\Delta F1$
UD English EWT	09.89%*	−34.26%*	−22.52%*
UD French GSD	12.34%*	−44.32%*	−41.71%*
UD German GSD	06.27%*	−10.48%*	−11.81%*
UD Portuguese Bosque	−06.45%*	04.65%*	06.29%*
UD Spanish AnCora	−00.14%	−04.02%*	−03.14%*

By considering this table, we notice interesting patterns. For English EWT, French GSD, and German GSD there is a substantial loss in precision, but with a (huge, for French GSD and English EWT) increase in recall. For Portuguese Bosque, on the contrary, precision rose considerably, to the very high value of 97%, but recall suffered a comparable loss. Finally, for Spanish AnCora, UDPipe showed moderate gains in recall, but an insignificant gain in precision.

An interesting feature of Udapi is that each test is associated with an explanation string, allowing to mark related tokens and having an indicative of which tests failed for a sentence. This allows us to inspect which verifications may be missing currently in our ontology, and more generally to compare tests realized with our validation.

For the automatically tagged sentences in the English EWT scenario, the five tests which failed in the most sentences are, from most frequent to less frequent: 1) a finite verb lacking a value for its mood feature, such as imperative or subjunctive (3448 sentences); 2) a pronoun lacking a value for its pronominal type, such as personal, relative or demonstrative (2848 sentences); 3) a word having more than one object child (379 sentences); 4) a word having more than one subject child (184 sentences); 5) a `cc` (coordinating conjunction) dependency relation not having a `CCONJ` (coordinating conjunction) part-of-speech (162 sentences).[7]

This is perhaps surprising. Although these tests constitute most violations, the first (1) and second (2) ones are not related to the syntactic representation, only to morphosyntactic features. This means that these errors are already occurring in the original test sets. While this could point out to a possible cause to the precision drop in English EWT, it is not expected that recall should increase. If this validation is unrelated to the second syntactic annotation (in this case, by parsing), then it would be expected for it to cause false positives, with at most a random increase in recall. However, wrong predicted analyses in our data are not unbiased, as they depend on the data used for learning (with an overlap in folds, due to cross validation) and on the inductive bias of the training model. This may reveal characteristics of our learned scenario.

[7]It should be stressed that multiple tests may fail for the same sentence.

Tests 3 and 4 have a direct corresponding in our ontology. As for test 5, we noticed that by a mistake it is currently missing in our ontology. Still, it is remarkable that adding the most occurring relevant restriction could not increase the recall by more than 1% (one percentage point).

Similar remarks are valid for French GSD. In this case, the most frequent (in sentences) occurring errors we: 1) a numeral lacking a numeral type morphosyntactic feature (6059 sentences); 2) a pronoun lacking a pronominal type morphosyntactic feature (3381 sentences); 3) a word with `cc` (coordinating conjunction) dependency type without an adequate part-of-speech (375 sentences); 4) a word with `det` (determinant) dependency type without an adequate part-of-speech (261 sentences); 5) a word with more than one object child (255 sentences)

By increases in recall, Udapi showed superior results in terms of F1-score with respect to finding predicted analyses which are not exact matches with respect to the original analysis. Nevertheless, there are two reasons for being careful about this result. While covering morphosyntactic features is desirable for finding errors in tests, in this case it seems that this restriction was decisive for the result *even though the exactly same errors occurred in the original analyses*, as they were no re-classified for morphosyntactic features. This is not informative with respect to our original goals. In addition, the main interest in having a validation tool is not maximizing F1 score or accuracy (as in usual classification tasks), but maximizing F1 subject to a sufficiently high precision. For instance, consider a human annotator provided with a validator. On running the validation, if precision is not sufficiently high, it becomes more likely that human will have to read an entire analyzed sentence in order to find out that the analysis is correct, and thus he should not modify it. By having high precision, specially pinpointing the error, manually correcting is more straightforward. In any case, specially after the level of impact measured, it is still necessary to incorporate these constraints on the ontology, even if only in a separate module.

Chapter 4

Treebank Development and Maintenance

Developing treebanks is essential for doing NLP, either for evaluating methods, or for training machine learning tools. Unfortunately, treebanks are typically hard to make, because they require manual annotation of syntactic information. Training annotators and guaranteeing consistency is hard, and annotation requires non-trivial decision problems (WALLIS, 2003). Furthermore, many treebanks come with their own specifications. Thus it is frequently necessary to convert them in order to use in other tasks, which causes errors.

Not only low-resource languages have few treebanks available, but even high-resource ones for text in specific domains, such as legal text, oil & gas, mining and biomedical. Manually annotating such texts requires expertise in both syntax and the domain, which makes the task mostly unfeasible without more robust methods. This is a problem as merely using tools trained out-of-domain cause a significant loss in performance (JIANG *et al.*, 2015; ZHANG *et al.*, 2018). Besides, for downstream semantic applications the quality of such analysis is essential. Thus, the problem of developing and maintaining annotated corpora is important for the success of practical applications. This is a topic of active research, as shown by a research handbook on the area (IDE and PUSTEJOVSKY, 2017), which is interesting to contrast to one of more than 10 years before (ABEILLÉ, 2003).

In this chapter, we will survey methodologies and tools for the development and maintenance of corpora, as well as illustrating the usage of our validator in an interactive environment for literate programming, along with other tools.

4.1 Querying

A very important method for interacting with a treebank is by *querying*, that is, using a search tool. When a user is interested in a specific word or expression, a query can be made in order to find examples of it on the treebank, seeing sentences in which it occurred and how they were annotated.

Usually query engines offer features such as searching not only by word forms (strings in text), but also by regular expressions, as well as searching by annotation content, such as labels and relations. These features may be offered in such a way that complex queries involving any of them can be made.

Examples of querying systems are in Figures 4.1, 4.2 and 4.3.

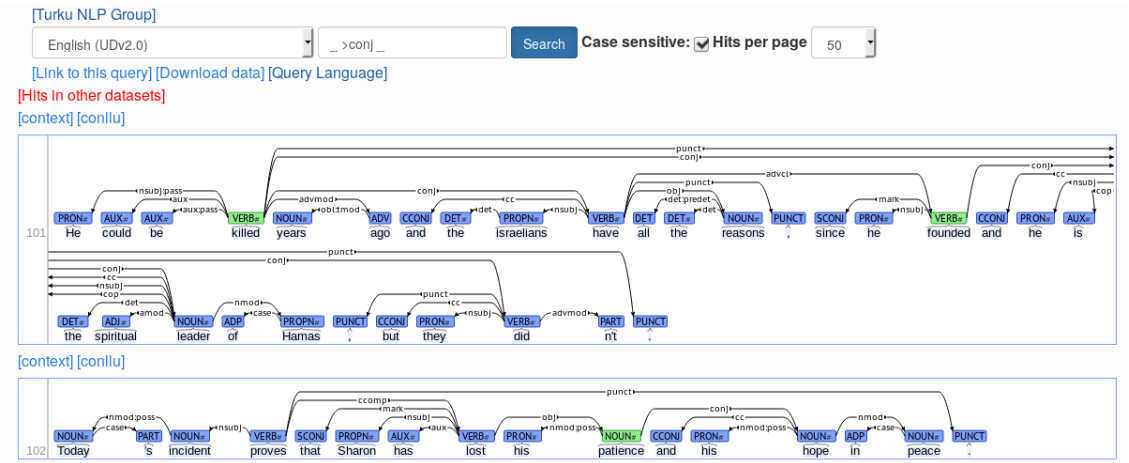

Figure 4.1: Example of query for right-headed `conj` relation in the SETS DEP_SEARCH tool (LUOTOLAHTI *et al.*, 2017).

Queries offer an interactive way of exploring a treebank. As queries are precise instructions for requiring a given information, they are transparent, that is, a user is able to understand exactly what is being asked and what is being returned, given that the query language itself is easy enough to read.

Specific linguistic phenomena and guidelines criteria may be tested by writing a query and considering any hit as a (possible) error. This requires the user to formulate manually each query he may be interested in. WALLIS (2003) proposed a methodology of "transverse correction", using queries for finding specific phenomena and having annotators correct one grammatical construction at a time, improving consistency.

There are many search tools available for UD, such as SETS DEP_SEARCH (LUOTOLAHTI *et al.*, 2017), Grew (GUILLAUME *et al.*, 2012) and Tree Query (STEPÁNEK and PAJAS, 2010).

There is currently a predefined sets of queries available for verifying the contents of UD analyses using specifications from the documentation. This set is a list of

Figure 4.2: Example of query for `conj` relation between words with different part-of-speech tag in the Grew tool (GUILLAUME *et al.*, 2012).

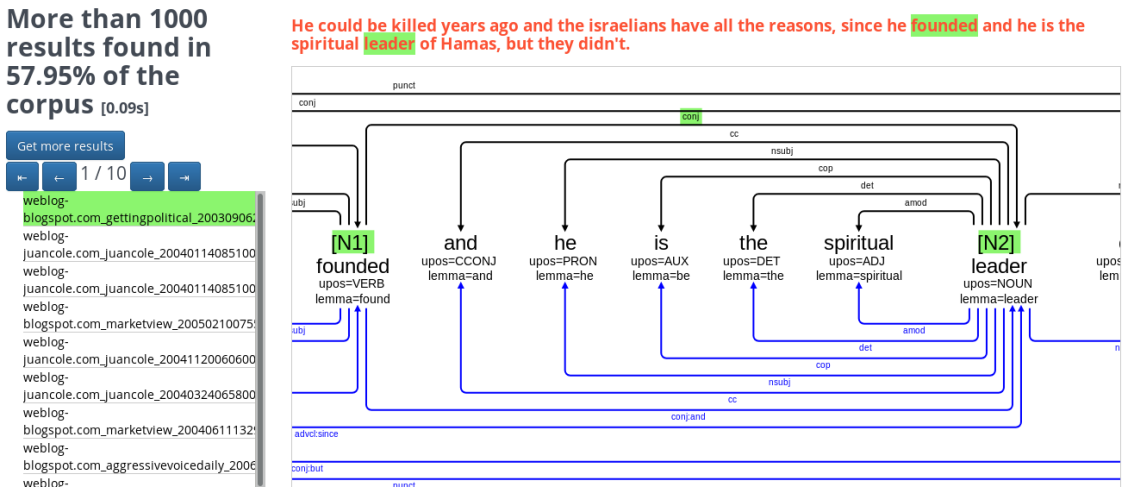

Figure 4.3: Result for the query in Figure 4.2.

queries automatically ran on the official UD datasets. Each test has a corresponding search expression and hits for each treebank are shown. Some of these tests contain a disclaimer that it is only a debugging test and having a hit does not imply that the data is invalid. It is called "syntactic validations" and it is provided by the UD project.[1]

4.2 Consistency verification

Consistency verification is extremely important for treebanks, as they are usually noisy. Noise can be generated both from inconsistent human annotation, bad guidelines, errors in data conversion and by errors of automatic parsers. Thus, in order to produce reliable data, consistency should be evaluated.

Furthermore, consistency verification can be used as a method for producing new reliable treebanks. For instance, by having an automatic parser and a reliable validation tool, one could simply remove the reject analyses, keeping the accepted ones, forming a corpus out of those sentences. If annotators are available, rejected analysis could be hand-corrected. This is already an improvement over selecting every sentence and manually validating them one by one. In the absence of annotators, using multiple methods for verifying consistency is likely an easy way to produce at least a "silver standard" dataset.

We briefly present some strategies for consistency checking in treebanks. We refer the reader to the survey of DICKINSON (2015) for a more detailed discussion.

Treebank consistency verification can be distinguished in two classes. One of them is *external consistency*, that is, consistency is verified with respect to some resource external to the treebank itself. The other class if *internal consistency*, that is, consistency with respect the treebank itself.

4.2.1 External consistency

A typical example of external consistency check is querying, as we just described. For instance, UD's syntactic validation page and the validation module of Udapi (see Section 3.1.3) are examples of external consistency, as they are based on an external notion of integrity, that is, which annotations are acceptable. Another example is the work of DE SMEDT *et al.* (2015), which uses a query tool in order to search for problematic constructions involving multiword expressions.

In OLIVA and KVETON (2002), authors explore the notion of *invalid bigrams* (and, more generally, invalid n-grams): that is, part-of-speech bigrams which are forbidden. This means that any annotation with a forbidden bigram is necessarily

[1] Available at `http://universaldependencies.org/svalidation.html`

wrong. PRZEPIÓRKOWSKI and LENART (2012) parse the same text with two independent automatic analysis: shallow (a little more than chunking) and deep (full syntactic trees). They then verify whether both agree in the head of a phrase. Thus, specific patterns in the dataset are considered problematic.

Our validation method based on an ontology is in this group. The ontology is (intended as) a machine-readable extension of the documentation. Thus it does not depend on data, which can possibly follow it or violate it, characterizing this method as external. Of course, the validity of this validation is entirely dependent on the validity of the ontology itself, that is, on whether the ontology indeed reflects the annotation guidelines. Ideally, the ontology should be maintained by the community. As treebank or language-specific maintainers can write specific guidelines (respecting the universal ones), they could as well build extensions of the the general ontology. This would be useful for asserting restrictions specific to their annotation scheme.

4.2.2 Internal consistency

Internal consistency verifies whether "similar" constructions in a dataset are annotated the same way. For part-of-speech tagging, DICKINSON and MEURERS (2003a) propose the *variation nuclei* method. A variation nucleus is a string which occurs multiple times in the dataset with different annotation. The variation nucleus with its context (surrounding strings) is called a *variation n-gram*. DICKINSON and MEURERS (2003a) use heuristics to prune which variation n-grams are more likely to be errors (instead of real ambiguities). (DICKINSON and MEURERS, 2003b) extend the method to phrase structure treebanks. Finally, BOYD *et al.* (2008) adapt the method to dependency treebanks, by considering pairs of words linked by a dependency relation as variation nuclei. They are labeled by the dependency relation and by the direction of the relation (left-headed or right-headed). Word pairs which on some sentences occur with a direct dependency relation are also compared to sentences where the same words occur but without a direct linking, by using a special `NIL` label. As in the other cases, heuristics on the context are used in order to filter which variation n-grams are more likely to be errors. This method was already tried for three UD treebanks, using not only word forms but also experimenting with lemmas instead (DE MARNEFFE *et al.*, 2017a). It was found that this increases the recall of the method. For English and French, there is only a small loss of precision, but for Finnish, a morphologically-rich language, the loss is huge (from 72% to 19%, as reported by the authors).

In (DICKINSON and MEURERS, 2005), errors in phrase structure syntactic annotation are found by checking the hypothesis that expressions are usually *endocentric*. This means that, given a phrase, which in turn is composed of subphrases,

the label of the subphrases should determine the label of phrase. Thus, in a phrase structure treebank, having many similarly labeled subphrases which combine into different classes should be seen as an error.

One could think about using querying for internal consistency checks. A user could write queries in order to find syntactic phenomena which she expects to be annotated in the same way. However, this would require reading results and comparing them, instead of being (semi-)automatic.

4.3 The Semantic Web approach

There is a line of research focusing on Semantic Web methods for natural language processing. One of their main goals is making *linguistic linked data*, that is, offering linguistic data that is:

1. interlinked, preferably being available on the web;

2. structurally and conceptually interoperable, improving data reusability. This avoids losing information from datasets which are no longer used ("legacy") (CHIARCOS *et al.*, 2013; IDE and PUSTEJOVSKY, 2010).

This line of research tries to develop and maintain annotated data (including treebanks and lexicosemantic resources) by creating interoperability layers between datasets, using Semantic Web technologies, such as RDF and OWL. For instance, for publishing lexical semantic resources, the interchange model *lemon* was created (MCCRAE *et al.*, 2012). On the task of finding errors in data, Semantic Web methods such as OWL, RDFS and SPARQL allow the specification of restrictions and richer queries, as was explored in the work of (KONTOKOSTAS *et al.*, 2014). Although using this technological stack may have costs in performance, its main benefits are reusability and flexibility, and they are not intended as a replacement for high-performance methods (HELLMANN *et al.*, 2013, p. 15; HELLMANN, 2015, p. 22).

A project aiming to cover many aspects of linguistic annotation is Ontologies of Linguistic Annotation (OLiA) (CHIARCOS and SUKHAREVA, 2015; CHIARCOS *et al.*, 2016). OLiA has a "reference model", which is an ontology which aims to be a "interlingua" of different linguistic annotation. It has general annotation concepts, covering mostly morphological and syntactic phenomena, but with extensions to discourse annotation. For connecting a data annotation formalism to OLiA, it is necessary to build a "linking model": an ontology which connects the specification of the data formalism to OLiA concepts. OLiA has already been used for NLP applications, such as morphosyntactic annotation for low-resource languages by

bootstrapping from taggers of similar high-resource languages (SUKHAREVA and CHIARCOS, 2016), as well as improving classification recall by integrating annotation from different tools and annotation styles (CHIARCOS, 2010). OLiA is also used in the NLP Interchange Format (NIF) (HELLMANN *et al.*, 2013), a project to integrate independent NLP applications. This is an alternative to centralized workflows such as UIMA (FERRUCCI and LALLY, 2004) or GATE (CUNNINGHAM *et al.*, 2002).

An important application of an ontology such as OLiA is increasing interoperability in querying corpora. For instance, imagine someone has many treebanks in different annotation formats, such as different tags and relations, but in a OWL representation. If there is a linking model from this representation to OLiA, it is possible to query for OLiA reference model concepts. For instance, one could query for `olia:CommonNoun` instead of querying for the part-of-speech tag `NOUN` in UD or for the Penn Treebank tags `NN` (singular or mass noun) and `NNS` (plural noun). This would allow a single query for data represented in different tags.

Our ontology can be useful for this kind of query enrichment, but when trying to find implicit information from UD datasets. For instance, we define the concept of a `NonverbalClauseHead`, a word which is the head of a nonverbal clause. This is not a concept annotated in CoNLL-U UD files, but it can be inferred from this annotation, using some background knowledge on UD. Our ontology includes this kind of background knowledge. Refinements to the ontology would allow finding more information, such as subtrees headed by specific kinds of word. Of course, this is restricted by the knowledge representation of UD annotation. For instance, as we discussed in Section 1.3, there is ambiguity between words which should be seen as modifying the head of a coordination or as modifying the entire coordination.

OLiA does not use many disjointness axioms, as their goal is including information from as many data sources as possible, in many languages and representations (CHIARCOS *et al.*, 2016, p. 66). Thus, constraints are not a focus of this ontology, as opposed to our work. Our ontology builds on the top ontology (`systems`) of OLiA, but is not currently linked to the OLiA reference model by a linking model.

4.4 Literate programming for treebanks: a Jupyter notebook approach

In this section we present an approach for interacting with treebanks by the usage of "notebooks". In computing, a "notebook" is an environment for writing prose and code, as well as visualizing results, with the goal of improving communication, reproducibility and interacting with source code. In a notebook, code is written in units

called "cells", and the output of each cell is presented after it (KERY *et al.*, 2018; KLUYVER *et al.*, 2016; MILLMAN and PÉREZ, 2014). Notebooks usually offer formatted output ranging from formatted text (such as Markdown, which includes headings, bold font, italics, hyperlinks, among other features) to equations, graphics or interactive controls (KLUYVER *et al.*, 2016, p. 88). More well-known notebook tools are Mathematica notebook interface (WOLFRAM RESEARCH, INC.) and Jupyter Notebook(KLUYVER *et al.*, 2016), the latter with at least 2 million users estimated in 2015. Notebooks are particularly used in scientific computing and data science for making exploratory analyses and writing scientific reports (KERY *et al.*, 2018; MILLMAN and PÉREZ, 2014).

A current problem with development and maintenance of corpora is that there are no *de facto* standard interfaces for it, so many projects develop their own solutions, which in turn are not reusable for different formalisms and standards. A proper interface should include functionalities such as visualization, querying and editing linguistic annotation. This is useful for easier navigation not only by a technical user, but especially for a less computer-trained user (which can be the case of domain experts and linguists). As the development of interfaces can be very costly, an hypothesis to be explored is whether notebooks can be used as an effective tool for interacting with corpora.

Here, we present a notebook in Jupyter with the goal of trying this hypothesis, highlighting the integration with the UD validation tool presented in Section 2.4. We will only briefly present how could such a notebook look like, but this is at experimental stage. This methodology is flexible enough for integrating other auxiliary services, which could be useful for a real framework for the maintenance of corpora.

4.4.1 System architecture

Jupyter notebook consists of a front-end web server which interacts with a *kernel*, which is an interpreter of a specified language. Initially, the Jupyter project was called *IPython notebook*, as the Python kernel was the only language available. Currently three kernels are officially supported by the Jupyter Project[2], but there are over 100 community-maintained kernels, most of them unique with respect to programming language. We use here the kernel for the language *Common Lisp* (PESCHANSKI, 2015).

We developed this project with *Docker*, a platform for developing and running applications in isolated environment called containers, separating concerns about infrastructure (MERKEL, 2014). We also used *Docker compose*, which is a tool for creating and running applications using multiple coordinated Docker containers.

[2]Python, Julia and R.

For each component an Docker image was developed, guaranteeing that each can be run in many platforms, as well as providing modularity.

Our Jupyter for treebanks consists of simply using a Common Lisp library for treebanks. In our case, mostly for CoNLL-U formatted treebanks, by the CL-CONLLU library (MUNIZ *et al.*, 2017). In order to guarantee modularity and scalability, we used a REST web service architecture for integrating the UD validation tool with the notebook. An advantage of this approach is that adapting to distributed frameworks should be easier.

In more details, there is a function for invoking the UD validation tool from Jupyter Notebook by a HTTP POST request which is made to the container responsible for running it. This POST request contains the CoNLL-U format of the sentence. In turn, this container sends another HTTP POST request the RDF converter container. Each container has its own HTTP server in order to handle adequately the requests. For the UD validator and the RDF converter, we use the Hunchentoot Common Lisp web server (WEITZ, 2009).

An illustration of the architecture is in figure 4.4.

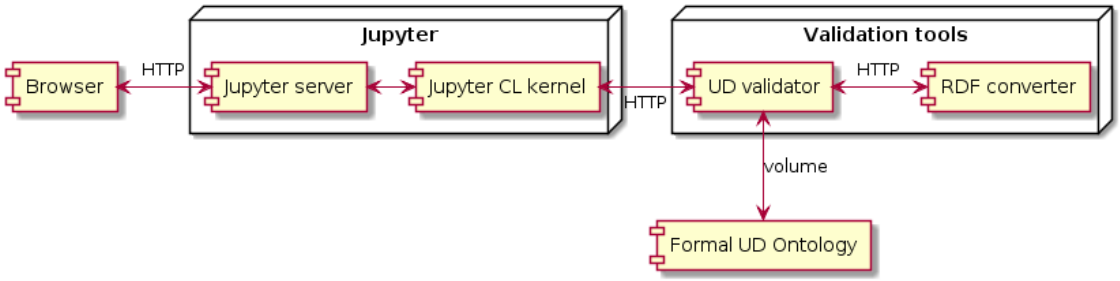

Figure 4.4: Architecture of Jupyter for treebanks, including the validation service.

4.4.2 Example usage

We illustrate some common use cases in treebank development in Figures 4.5, 4.6 and 4.7.

Most features that we present are offered by the CL-CONLLU library. Our contribution is developing the web server architecture for the validation service and integrating with the usage of the library, besides developing the (rudimentary, at the moment) package directly used in Jupyter for treebanks, as well as the example notebook. In the future, a textual interface easier for non-programmers could be developed.

Printing conllu tabular format

```
In [2]: (write-conllu-to-stream *sents*)

# sent_id = test-invalid
# text = The US troops fired into the hostile crowd, killing 4.
1       The     the     DET     DT      Definite=Def|PronType=Art       3       det     _       _
2       US      US      PROPN   NNP     Number=Sing     4       nsubj   _       _
3       troops  troops  NOUN    NNS     Number=Plur     4       nsubj   _       _
4       fired   fire    VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        0       root    _       _
5       into    into    ADP     IN      _       8       case    _       _
6       the     the     DET     DT      Definite=Def|PronType=Art       8       det     _       _
7       hostile hostile ADJ     JJ      Degree=Pos      8       amod    _       _
8       crowd   crowd   NOUN    NN      Number=Sing     4       obl     _       SpaceAfter=No
9       ,       ,       PUNCT   ,       _       4       punct   _       _
10      killing kill    VERB    VBG     VerbForm=Ger    4       advcl   _       _
11      4       4       NUM     CD      NumType=Card    10      obj     _       SpaceAfter=No
12      .       .       PUNCT   .       _       4       punct   _       _

# sent_id = test
# text = The US troops fired into the hostile crowd, killing 4 men.
1       The     the     DET     DT      Definite=Def|PronType=Art       3       det     _       _
2       US      US      PROPN   NNP     Number=Sing     3       compound        _       _
3       troops  troops  NOUN    NNS     Number=Plur     4       nsubj   _       _
4       fired   fire    VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        0       root    _       _
5       into    into    ADP     IN      _       8       case    _       _
6       the     the     DET     DT      Definite=Def|PronType=Art       8       det     _       _
7       hostile hostile ADJ     JJ      Degree=Pos      8       amod    _       _
8       crowd   crowd   NOUN    NN      Number=Sing     4       obl     _       SpaceAfter=No
9       ,       ,       PUNCT   ,       _       4       punct   _       _
10      killing kill    VERB    VBG     VerbForm=Ger    4       advcl   _       _
11      4       4       NUM     CD      NumType=Card    12      nummod  _       _
12      men     man     NOUN    NNS     Number=Sing     10      obj     _       SpaceAfter=No
13      .       .       PUNCT   .       _       4       punct   _       _
```

Figure 4.5: Example of printing analysis of two sentences in the CoNLL-U format.

Drawing sentences

```
In [3]: (conllu.draw:tree-sentence (second *sents*))

                ┌─ The det
                ├─ US compound
             ┌─ troops nsubj
          ┌─ fired root
          │  ├─ into case
          │  ├─ the det
          │  ├─ hostile amod
          │ ┌─ crowd obl
          │ ├─ , punct
          │ ┌─ killing advcl
          │ │  ┌─ 4 nummod
          │ │ ┌─ men obj
          │ │ └─ . punct

Out[3]: NIL
```

Figure 4.6: Example of printing a drawing of the tree of the first sentence (MUNIZ *et al.*, 2017).

The first sentence is invalid, because the verb has two subjects. Therefore, it returns the found explanation.

```
In [4]: (verify (first *sents*))
```

```
Out[4]: " punct SubClassOf DependencyRelation
         isSourceOf EquivalentTo inverse (hasSource)
         Functional: hasUniversalPartOfSpeech
         Word SubClassOf isSourceOf max 1 (csubj or nsubj)
         test-invalid-12.edge Type punct
         DependencyRelation SubClassOf hasSource exactly 1 Word
         test-invalid-2.edge hasSource test-invalid-4
         test-invalid-2.edge Type nsubj
         test-invalid-3.edge Type nsubj
         Relation SubClassOf hasTarget exactly 1 Thing
         test-invalid-3.edge hasTarget test-invalid-3
         PROPN_CLASS EquivalentTo {PROPN}
         test-invalid-2 hasUniversalPartOfSpeech PROPN
         OpenClass DisjointUnionOf ADJ_CLASS, ADV_CLASS, INTJ_CLASS, NOUN_CLASS, PROPN_CLASS, VERB_CLASS
         test-invalid-4 hasUniversalPartOfSpeech VERB
         Relation EquivalentTo (hasSource min 1 Thing) or (hasTarget min 1 Thing)
         test-invalid-2 nextWord test-invalid-3
         test-invalid-12.edge hasSource test-invalid-4
         test-invalid-3 nextWord test-invalid-4
         test-invalid-2.edge hasTarget test-invalid-2
         Functional: nextWord
         test-invalid-3.edge hasSource test-invalid-4
         VERB Type VERB_CLASS
         Relation SubClassOf hasSource exactly 1 Thing
        "
```

The second sentence is valid, there is no inconsistency in it. Therefore, the result is `NIL`.

```
In [5]: (verify (second *sents*))
```

```
Out[5]: NIL
```

Figure 4.7: Example of validation output of the two sentences.

4.5 Other approaches

Another line of work studies the impact of syntactic representation on *learnability*, that is, on the results of trainable parsers (SCHWARTZ *et al.*, 2012; SILVEIRA and MANNING, 2015; WISNIEWSKI and LACROIX, 2017). That is, they consider whether specific ways of annotating syntax improves or deteriorates learning results. For instance, WISNIEWSKI and LACROIX (2017) find out that many transformations on UD's syntactic structure which were expected to improve parsing did not produce better results with a transition-based parser.

RUMSHISKY and STUBBS (2017, section 4) discuss some machine learning techniques in order to leverage limited quantities of annotated data. It presents briefly some semi-supervised learning techniques, that is, methods for cases when there is small amount of labeled data and another set (usually bigger) of unlabeled data. One family of methods discussed is active learning, where the algorithm queries the user for the label of some specific instances, usually ones with must uncertainty, more controversial between multiple automatic classifiers (taggers or parsers), or which would maximally modify the model. Another class of present methods is co-training, where multiple classifiers are trained on the same data, run on the unlabeled data and the output of one is used for re-training the other.

For a different development context, (FLICKINGER *et al.*, 2017) present an approach centered on a *grammar*. While the grammar is built manually and requires much human work, it is argued that it is also reusable, the corpus is built from human choice between possible readings of the sentences by the grammar. This annotation is done by the method of *discriminants*, features which divide the possible readings in partitions (CARTER, 1997). This allows an annotator to, instead of reading syntactic analysis and understanding then in order to choose the right one, simply answer specific questions, narrowing the correct interpretation for the sentence in consideration. In this case, a treebank and a grammar may have complementary developments, where one serves as a resource for improving the other.

4.6 Example practical application

We will briefly comment a real-world scenario in which the validator was used.[3] For evaluating information extraction techniques in the oil & gas domain, a (proprietary) corpus annotated with entities, relations and co-references was available to the researchers. This corpus was sampled randomly from 1298 geological reports in English, made available by the United States Geological Survey, Geological Survey of Canada, and British Geological Survey. From this, 155 text passages were selected according to relevance to petroleum systems, and annotated with entities, relations and co-references internal to the same document. Notice that the resulting dataset (hereinafter oil & gas corpus, O&GC) does not include syntactic annotation. The project goal is to develop a information extraction tool for the oil & gas domain, using syntactic knowledge in order to reduce the dependency on direct semantic annotation on data. While direct semantic annotation does not require the same linguistic knowledge as syntactic annotation, many annotation costs are still high, such as time for a human annotator, possible inconsistencies between annotators and the impossibility of using crowd-sourcing methods, due to the technical subject of the text.

Of course, the problem is that there are no readily available treebanks for the domain of oil & gas. Developing a new treebank by manual annotation shares many of the drawbacks of manually annotating semantics directly. Besides, syntax arguably requires more linguistic expertise, which is unlikely to be hold by annotators with competence in the domain. Thus, for using syntactic methods, there are many advantages in working with known frameworks, leveraging existing resources, even if out-of-domain.

Thus, from a sample of 3 documents out of the 155, three parsing methods were

[3]This application was done during the author's internship period at IBM Research Brazil with the co-supervisor.

compared:

1. the head-driven phrase structure grammar, implemented in the English Resource Grammar (ERG) (COPESTAKE and FLICKINGER, 2000);

2. the English Slot Grammar (ESG) (MCCORD, 1990);

3. the dependency parser UDPipe trained on UD English EWT 2.0, composed mostly of texts from web-blogs, emails, reviews, newsgroups and question-answers (SILVEIRA *et al.*, 2014).

Formal grammars are subject to the robustness problem, as it is possible for them to not find an acceptable adequate analysis for a sentence. On this situation, modern grammars usually do not simply refuse to produce an analysis. Two approaches are used. The first one is relaxing constraints on the grammar, making it *overgenerate*, that is, generating more than would be initially expected. The downside of this approach is that the problem of lack of robustness now becomes lack of disambiguation, which should then be solved. Another one is producing a *partial* analyses returning incomplete syntactic information instead of returning false information or none at all (NIVRE, 2006, p. 22). Even then, it is possible to have a rough measurement of recall by how many sentences were fully parsed by the grammar.

On the other hand, many data-driven methods such as UDPipe follow an approach of *robust disambiguation*, that is, for every text received by the parser, exactly one analysis should be returned, regardless of how correct it is. This guarantees that the parser will not suffer from severe breaks by unseen words or constructions, but try to gracefully approximate according to learned examples.

For at least a rough (but quick) comparison between the behavior of a set of parsers in a corpus, comparing two grammars is easy at least with respect to recall, that is, ratio of fully-parsed sentences (regardless of precision) to the total of sentences. As by design many data-driven methods always produce at least an analysis, this is not a fair comparison. On the other hand, evaluating precision (the correctness of analysis) is more labor-intensive.

Fortunately, by using a validator, the scenarios become somewhat closer. Absurd analysis are detected, and thus there is a filter for guaranteeing at least minimum quality. Thus, it is possible to consider the recall of a robust disambiguation parser as the ratio of errorless (that is, non detected as invalid) parsed sentences to the total.

By using this approach on the three documents subsample of the O&GC, we find out that, out of 203 sentences, ESG fully parses 85% out of them, ERG, 63%. Finally, for UDPipe, 60% of the sentences were not found invalid by the ontology. Using the `ud-theory-and-svalidation` (see Section 2.3.2), only 43% are considered

valid. B inspection we have found important false positives, which confirms results from Chapter 3 of a considerable high false positive rate. However, these results also suggest a higher false negative rate. That is, it is possible that even more sentences are wrong. This rough analysis suggests that there may be a case for well-developed grammars in comparison to data-driven parses; at least that grammars are not trivially worse than data-driven robust disambiguation methods. Of course, any serious conclusion on this issue can only be settled by more controlled scenarios. A validator was useful in this case for producing estimates for an otherwise incommensurable use case in practical decision-making in industry.

Chapter 5

Conclusion

5.1 Discussion

In this work, we presented an OWL-DL ontology for the Universal Dependencies (UD) standard. This ontology covers the universal guidelines. As a consequence, it is valid for every UD treebank, regardless of language.[1] We presented its characteristics and argued for its utility in different contexts. One possible application is using the ontology to create a validation method for finding errors in treebanks, regardless of the data origin: manual annotation, automatic annotation, automatic conversion from other formats, or a combination of these methods. This includes post-processing parser output, although the time cost of consistency checking makes it likely unattractive for an end-user. A different application is enriching treebank querying, although we did not explore this here in detail.

We evaluated our validation method in a task of classifying automatically produced syntactic analyses as correct or incorrect, as a proxy to the task of classifying analyses as valid or invalid. We showed that for many languages it finds incorrect sentences with high precision, but not high recall. This is expected, as the methods tries to capture only syntactic invalidities, not to classify any syntactic analysis as correct or incorrect. By inspecting false positives, we find excessive restrictions in our ontology (for instance, due to ellipsis, a harder to capture phenomenon), real errors in treebanks, and imprecisions on the guidelines. This suggests that inspecting treebanks with our ontology indeed provides a list of sentences with high probability of being wrong. Besides, building and using an ontology may reveal problems in the natural language documentation, raising imprecisions, simplified cases and inconsistencies in the guidelines.

[1] Of course, specific treebanks may end up violating universal guidelines on purpose. This may happen either with a specific guideline, or simply in the data, without any documentation of this annotation decision. However, from the point of view of the UD project, this would be an error. Besides, violating the universal guidelines is not the same as making additional specifications, which is precisely the goal of having specific guidelines.

An advantage of using an ontology for this task is that errors are found by logical reasoning. This implies that it is theoretically possible to generate some explanation of this process, such as by tracing the proof. However, in practice implemented methods in available libraries use less detailed methods for explanation, such as Hitting Set Tree methods, which only show which axioms entail the conclusion. Such an explanation can be confusing in practice, even for users experienced in OWL, and it is thus necessary to use better methods (HORRIDGE, 2011, chapter 12). In this work, we have found confusing explanations, but were still able to understand it due to limited number of axioms in the ontology. Unfortunately, an end user is probably less used to OWL or formal logic, so better explanations should be given in order to deploy the system in practice.

We have also contextualized our work with other methods for maintaining and improving treebanks. Querying is a central concept in interacting with treebanks and can be improved by ontologies, but there are other reliable methods as well, with different strong points. This suggests that multiple methods should be used in a project, in order to improve the quality of the datasets. Furthermore, our work is related to the Semantic Web community, which is actively interested in developing ontologies for NLP. Some works have already successfully used ontologies for interoperability of tools and datasets, and for finding errors in data.

5.2 Future Work

Creating large treebanks is usually motivated by having examples for evaluating systems and training machine-learning-based methods. Examples encode information that is implicit, and many times not entirely clear in the absence of data. On the other hand, syntactic formalisms may be over-inclusive, in ways that not every possible way of annotating data is meaningful or acceptable on linguistic grounds. This points out that both data-driven and declarative methods have uses in generating, manipulating and using linguistic data. While here we explored an ontology, a declarative specification, it is still an open problem to combine this ontology with the plentiful of data existing in the UD project.

One approach for this combination would be using symbolic machine learning methods for revising or extending the specification from data. In practice this would depend on the existence of sufficiently reliable data, but, as we have seen, there are methods for improving datasets. As an ontology can be used to validate data, this suggests a positive feedback loop, where an ontology could filter reliable data, which would be used to extend its quality, improving its ability to find more reliable data.

Another extension is using a probabilistic ontology, where axioms could be weighted in order to represent degrees of certainty on the constraints. Instead of a

binary valid or invalid validation, this would allow ranking sentences according to a continuous metric of invalidity. This could prove useful for a treebank maintainer for prioritizing which sentences to correct first.

Other representations for a formal specification could be explored. While OWL-DL (or, more precisely, its underlying description logic $\mathcal{SROIQ}$) was indeed sufficiently expressive for representing UD constraints, its worst-case complexity is high, and consistency checking for long sentences (e.g. of at least 100 words) can take a long time to complete. While the usual sentence is much shorter than this, for specific corpora this may become problematic. Thus, discovering whether the same restrictions could be implemented in a logical language with reduced complexity could prove to be useful. A possible solution to this problem is not using consistency tests for validation, which require theorem-proving and are thus expensive. An alternative which could be investigated is using *model checking*, that is, considering the representation of the sentence as a model and verifying if the ontology is satisfied in the model (HALPERN and VARDI, 1991). This conforms to our task and has the advantage that it can be computationally cheaper. Unfortunately, there are no out-of-the-shelf model checkers for description logics available, much less for OWL2-DL.

A different approach to testing conformance to a specification, instead of satisfiability or calculating probabilities, is using *inconsistency measures*. There are some logic-based approaches for measuring inconsistency in knowledge bases, increasing the granularity, instead of a binary classification in consistent or inconsnstence. An example of this line of research is the work of HUNTER and KONIECZNY (2008).

As we have discussed before, improving explanations of reasoning is an important line of work to using an ontology in real applications. One method to do so would be using better justification-based explanations (HORRIDGE *et al.*, 2008), which provide more concise explanations. Another possibility is using deductive calculi (such as natural deduction or sequent calculus methods) for generating proof trees (RADEMAKER, 2010).

Finally, we believe that it could be greatly useful to maintain an official UD ontology, by the same community effort used to maintain natural language guidelines. The ontology and the natural language documentation could be more tightly integrated, allowing easy comparison between the two, removing ambiguity in the documentation (by a formal model), and automating testing of guidelines sentence examples and treebanks.

Appendix A

Ontology

Here we present our ontology, the one used in our experiment. Thus, according to our classification in Section 2.3.2, here we have `system`, `ud-annotation-model`, `ud-structure`, and `ud-theory`.

```
Prefix: : <http://www.semanticweb.org/gppassos/ontologies
    /2018/2/merged-ud-theory#>
Prefix: p: <#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: ud-theory: <http://www.semanticweb.org/gppassos/
    ontologies/2017/10/ud-theory#>
Prefix: olia_system: <http://purl.org/olia/system.owl#>
Prefix: ud-annotation-model: <http://www.semanticweb.org/
    gppassos/ontologies/2017/7/ud-annotation-model#>



Ontology: <http://www.semanticweb.org/gppassos/ontologies
    /2018/2/merged-ud-theory>



Annotations:
     rdfs:comment "Annotation model for Universal
        Dependencies 2.0.",
```

```
    owl:versionInfo "TODO: LinguisticAnnotation disjoint",
    owl:versionInfo "
      2008-01-13 created
      2010-04-06 removed deprecated Category (equiv
         UnitOfAnnotation) category
          2010-04-14 added AnnotationProcess (cf. DCR
             process directory)
          2011-07-15 replaced base url by purl
          2011-07-27 added hasTagMatching with full support
             for XSLT-style regular expressions
          2013-06-27 added ISOcat reference for
             LinguisticAnnotation

      Christian Chiarcos, chiarcos@uni-potsdam.de
    ",
    rdfs:comment "OLiA core concepts for linguistic
       annotations."

AnnotationProperty: ud-annotation-model:redundantWith

    Annotations:
        rdfs:comment "In order to annotate detected
           redundancies."

    Annotations:
        rdfs:comment "In order to annotate detected
           redundancies."


AnnotationProperty: owl:versionInfo


AnnotationProperty: rdfs:isDefinedBy


AnnotationProperty: ud-theory:verify

    Annotations:
```

```
        rdfs:comment "Marks properties and annotation that
            should ve revised or otherwise verified in some
            way."

    Annotations:
        rdfs:comment "Marks properties and annotation that
            should ve revised or otherwise verified in some
            way."


AnnotationProperty: ud-annotation-model:worldAssumption

    Annotations:
        rdfs:comment "Informally: Can this axiom be used
            with open world assumption or is it necessary to
            use closed world assumption in order to work as
            expected?"

    Annotations:
        rdfs:comment "Informally: Can this axiom be used
            with open world assumption or is it necessary to
            use closed world assumption in order to work as
            expected?"


AnnotationProperty: owl:qualifiedCardinality


AnnotationProperty: rdfs:label


AnnotationProperty: rdfs:comment


Datatype: rdf:PlainLiteral


Datatype: xsd:string
```

```
Datatype: xsd:nonNegativeInteger


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:dislocated>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:cc>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:clf>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:conj>

     SubPropertyOf:
          ud-annotation-model:isDepOf

     DisjointWith:

               Annotations: rdfs:comment "\"A conjunct is the
                  relation between two elements connected by a
                  coordinating conjunction, such as and, or,
                  etc. We treat conjunctions asymmetrically:
                  The head of the relation is the first
                  conjunct and all the other conjuncts depend
```

```
                on it via the conj relation.\"

( http://universaldependencies.org/u/dep/conj.html )"
        <http://persistence.uni-leipzig.org/nlp2rdf/
            ontologies/nif-core#nextWordTrans>


ObjectProperty: ud-annotation-model:prevWordTrans

    InverseOf:
        <http://persistence.uni-leipzig.org/nlp2rdf/
            ontologies/nif-core#nextWordTrans>


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:flat>

    SubPropertyOf:
        ud-annotation-model:isDepOf

    DisjointWith:

            Annotations: rdfs:comment "\"Structures analyzed
                with fixed and flat are headless by
                definition and are consistently annotated by
                attaching all non-first elements to the first
                and only allowing outgoing dependents from
                the first element.\"

( http://universaldependencies.org/u/overview/specific-
    syntax.html#multiword-expressions )"
        <http://persistence.uni-leipzig.org/nlp2rdf/
            ontologies/nif-core#nextWordTrans>


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:expl>

    SubPropertyOf:
```

ud-annotation-model:isDepOf

ObjectProperty: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#nextWordTrans>

Annotations:
rdfs:comment "\"W1 nextWordTrans W2\" means that W1 comes before (in the sentence) than W2, that is, W1 is to the left of W2."

DisjointWith:

Annotations: rdfs:comment "\"An appos is a nominal phrase that follows the head of another nominal phrase and stands in a co-reference or other equivalence relation to it.\"

(http://universaldependencies.org/u/overview/nominal-syntax.html)

\"An appositional modifier of a noun is a nominal immediately following the first noun that serves to define, modify, name, or describe that noun. It includes parenthesized examples, as well as defining abbreviations in one of these structures.\"

(http://universaldependencies.org/u/dep/appos.html)"
<http://www.semanticweb.org/gppassos/ontologies/2017/7/ud-annotation-model#isDepOf:appos>,

Annotations: rdfs:comment "\"Structures analyzed with fixed and flat are headless by definition and are consistently annotated by attaching all non-first elements to the first and only allowing outgoing dependents from the first element.\"

```
( http://universaldependencies.org/u/overview/specific-
   syntax.html#multiword-expressions )"
        <http://www.semanticweb.org/gppassos/ontologies
           /2017/7/ud-annotation-model#isDepOf:fixed>,

            Annotations: rdfs:comment "\"A conjunct is the
               relation between two elements connected by a
               coordinating conjunction, such as and, or,
               etc. We treat conjunctions asymmetrically:
               The head of the relation is the first
               conjunct and all the other conjuncts depend
               on it via the conj relation.\"

( http://universaldependencies.org/u/dep/conj.html )"
        <http://www.semanticweb.org/gppassos/ontologies
           /2017/7/ud-annotation-model#isDepOf:conj>,

            Annotations: rdfs:comment "\"Structures analyzed
                with fixed and flat are headless by
               definition and are consistently annotated by
               attaching all non-first elements to the first
                and only allowing outgoing dependents from
               the first element.\"

( http://universaldependencies.org/u/overview/specific-
   syntax.html#multiword-expressions )"
        <http://www.semanticweb.org/gppassos/ontologies
           /2017/7/ud-annotation-model#isDepOf:flat>

    InverseOf:
        ud-annotation-model:prevWordTrans


ObjectProperty: ud-annotation-model:nextWord

    Domain:
        ud-annotation-model:WordUnit

    Range:
```

```
        ud-annotation-model:WordUnit


ObjectProperty: ud-annotation-model:inSentence

    Characteristics:
        Functional

    InverseOf:
        ud-annotation-model:hasWord


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:nmod>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:isTargetOf

    EquivalentTo:
         inverse (olia_system:hasTarget)


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:discourse>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:iobj>

    SubPropertyOf:
        ud-annotation-model:isDepOf
```

```
ObjectProperty: ud-annotation-model:wordAfter

    Domain:
        ud-annotation-model:WordUnit

    Range:
        ud-annotation-model:WordUnit

    InverseOf:
        ud-annotation-model:wordBefore


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:mark>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:root>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: olia_system:hasTarget

    Annotations:
        rdfs:comment "A Relation is a directed edge between
            a source and a target concept."^^xsd:string

    Domain:
        olia_system:Relation

    Range:
        olia_system:UnitOfAnnotation
```

```
ObjectProperty:  inverse (olia_system:hasTarget)

    EquivalentTo:
        ud-annotation-model:isTargetOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:nummod>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:case>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:orphan>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:appos>

    SubPropertyOf:
        ud-annotation-model:isDepOf

    DisjointWith:

            Annotations: rdfs:comment "\"An appos is a
                nominal phrase that follows the head of
                another nominal phrase and stands in a co-
                reference or other equivalence relation to it
```

```
                .\"

( http://universaldependencies.org/u/overview/nominal-syntax
   .html )

\"An appositional modifier of a noun is a nominal
   immediately following the first noun that serves to
   define, modify, name, or describe that noun. It includes
   parenthesized examples, as well as defining abbreviations
    in one of these structures.\"

( http://universaldependencies.org/u/dep/appos.html )"
        <http://persistence.uni-leipzig.org/nlp2rdf/
           ontologies/nif-core#nextWordTrans>


ObjectProperty: ud-annotation-model:hasWord

    Domain:
        ud-annotation-model:SentenceUnit

    Range:
        ud-annotation-model:WordUnit

    InverseOf:
        ud-annotation-model:inSentence


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:amod>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:punct>

    SubPropertyOf:
```

```
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:dep>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:isSourceOf

    EquivalentTo:
         inverse (olia_system:hasSource)


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:vocative>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:wordBefore

    Annotations:
        rdfs:comment "\"W1 wordBefore W2\" means that W1
           comes before (in the sentence) than W2, that is,
           W1 is to the left of W2."

    InverseOf:
        ud-annotation-model:wordAfter


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:nsubj>

    SubPropertyOf:
        ud-annotation-model:isDepOf
```

```
ObjectProperty: <http://persistence.uni-leipzig.org/nlp2rdf/
    ontologies/nif-core#nextWord>

    SubPropertyOf:
        <http://persistence.uni-leipzig.org/nlp2rdf/
            ontologies/nif-core#nextWordTrans>

    Characteristics:
        Functional

    InverseOf:
        ud-annotation-model:prevWord


ObjectProperty: olia_system:hasFeature

    Annotations:
        rdfs:comment "A UnitOfAnnotation or a Relation can
            carry Features that express annotations attached
            to them. By convention, (tags that represent)
            Features can be linked with Feature individuals,
            as well. Because of this reflexivity, a predicate
             like hasDegree(positive) allows to retrieve the
            individual positive as well. (This is necessary
            if positive represents a tag on its own, rather
            than being a property of a complex tag.)"^^xsd:
            string

    Domain:
        olia_system:LinguisticAnnotation

    Range:
        olia_system:Feature

    InverseOf:
         inverse (olia_system:hasFeature)
```

```
ObjectProperty:  inverse (olia_system:hasFeature)

    InverseOf:
        olia_system:hasFeature


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:ccomp>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:obj>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:det>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:obl>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:prevWord

    SubPropertyOf:
        ud-annotation-model:prevWordTrans
```

```
    InverseOf:
        <http://persistence.uni-leipzig.org/nlp2rdf/
            ontologies/nif-core#nextWord>


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:aux>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:hasUniversalPartOfSpeech

    SubPropertyOf:
        olia_system:hasFeature

    Characteristics:
        Functional

    Domain:
        ud-annotation-model:WordUnit

    Range:
        ud-annotation-model:UniversalPartOfSpeech


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:advmod>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:acl>

    SubPropertyOf:
```

```
        ud-annotation-model:isDepOf


ObjectProperty: olia_system:hasSource

    Annotations:
        rdfs:comment "A Relation is a directed edge between
           a source and a target concept."^^xsd:string

    Domain:
        olia_system:Relation

    Range:
        olia_system:UnitOfAnnotation


ObjectProperty:  inverse (olia_system:hasSource)

    EquivalentTo:
        ud-annotation-model:isSourceOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:parataxis>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:fixed>

    SubPropertyOf:
        ud-annotation-model:isDepOf

    DisjointWith:

            Annotations: rdfs:comment "\"Structures analyzed
                with fixed and flat are headless by
```

```
                definition and are consistently annotated by
                attaching all non-first elements to the first
                 and only allowing outgoing dependents from
                the first element.\"

( http://universaldependencies.org/u/overview/specific-
   syntax.html#multiword-expressions )"
        <http://persistence.uni-leipzig.org/nlp2rdf/
           ontologies/nif-core#nextWordTrans>


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:list>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:advcl>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:reparandum>

    SubPropertyOf:
        ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
   ontologies/2017/7/ud-annotation-model#isDepOf:compound>

    SubPropertyOf:
        ud-annotation-model:isDepOf
```

```
ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:csubj>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:cop>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:goeswith>

     SubPropertyOf:
          ud-annotation-model:isDepOf


ObjectProperty: ud-annotation-model:isDepOf

     Domain:
          ud-annotation-model:WordUnit

     Range:
          ud-annotation-model:WordUnit


ObjectProperty: <http://www.semanticweb.org/gppassos/
    ontologies/2017/7/ud-annotation-model#isDepOf:xcomp>

     SubPropertyOf:
          ud-annotation-model:isDepOf


DataProperty: olia_system:hasTier
```

```
    Annotations:
        rdfs:comment "Assigns a linguistic annotation a
           string representation of the annotation layer (\"
           tier\", \"level\") where it is to be found, e.g.,
            \"pos\" for Part of Speech annotation, \"gloss\"
            for linguistic glosses, etc."^^xsd:string

    Domain:
        olia_system:LinguisticAnnotation


DataProperty: olia_system:hasTagMatching

    Annotations:
        rdfs:comment "hasTagMatching allows to provide
           regular expressions as those used in XSLT and
           XPath, see http://www.w3.org/TR/xquery-operators
           /#func-matches"^^xsd:string,
        rdfs:comment "hasTagMatching is a subproperty of
           hasTag, so that results can be retrieved if the
           regular expression match is requested, but an
           exact value with reserved characters is defined
        "

    SubPropertyOf:
        olia_system:hasTag


DataProperty: olia_system:hasTagEndingWith

    Annotations:
        rdfs:comment "As opposed to hasTag proper, the
           string representation defines the final
           subsequence of a concrete annotation.
The respective linguistic annotation then applies to every
   element whose annotation (tag) ends with this substring
   ."^^xsd:string

    SubPropertyOf:
```

```
        olia_system:hasTag


DataProperty: olia_system:hasTagContaining

    Annotations:
        rdfs:comment "As opposed to hasTag proper, the
           string representation defines a substring of a
           concrete annotation.
The respective linguistic annotation then applies to every
   element whose annotation (tag) contains this substring
   ."^^xsd:string

    SubPropertyOf:
        olia_system:hasTag


DataProperty: olia_system:hasTagStartingWith

    Annotations:
        rdfs:comment "As opposed to hasTag proper, the
           string representation specifies only the
           beginning of a concrete annotation.
The respective linguistic annotation then applies to every
   element whose annotation (tag) startsWith this substring
   ."^^xsd:string

    SubPropertyOf:
        olia_system:hasTag


DataProperty: olia_system:hasTag

    Annotations:
        rdfs:comment "Assigns a Linguistic Annotation a
           String representation, e.g., a particular Part of
            Speech tag, the respective abbreviation of the
           grammatical cases used in an annotation scheme,
           etc.
```

```
implicit semantics: hasTag is to be used if the tag is equal
    to the string value, otherwise use hasTagContaining,
   hasTagStartingWith, hasTagEndingWith"^^xsd:string

    Domain:
        olia_system:LinguisticAnnotation


Class: ud-annotation-model:Definite

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:EvidentNfh

    EquivalentTo:
        {ud-annotation-model:evidentNfh}

    SubClassOf:
        ud-annotation-model:Evident

    DisjointWith:
        ud-annotation-model:EvidentFh


Class: ud-annotation-model:discourse

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:Ellipsis

    Annotations:
        rdfs:comment "\"The UD approach to ellipsis can be
           summarized as follows:
```

```
1. If the elided element has no overt dependents, we do
   nothing.
2. If the elided element has overt dependents, we promote
   one of these to take the role of the head.
3. If the elided element is a predicate and the promoted
   element a core argument, we use the orphan relation when
   attaching other non-functional dependents to the promoted
    head.\"

(http://universaldependencies.org/u/overview/specific-syntax
   .html#ellipsis)"

    SubClassOf:
        ud-annotation-model:Concept


Class: ud-annotation-model:CCONJ_CLASS

    EquivalentTo:
        {ud-annotation-model:CCONJ}

    SubClassOf:
        ud-annotation-model:ClosedClass


Class: ud-annotation-model:ComplexClause

    SubClassOf:
        ud-annotation-model:Clause


Class: olia_system:AnnotationProcess

    Annotations:
        owl:versionInfo "DCR annotation and editing
           operations ignored, e.g., add first vowel  http
           ://www.isocat.org/datcat/DC-2199
                ",
```

```
        owl:versionInfo "categories for annotation and
           editing operations added to account for the \"
           Processes\" profile in the DCR"


Class: ud-annotation-model:VerbFormInf

    EquivalentTo:
        {ud-annotation-model:verbFormInf}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:NumType

    SubClassOf:

            Annotations: rdfs:comment "\"From the syntactic
               point of view, some numtypes behave like
               adjectives and some behave like adverbs. We
               tag them ADJ and ADV respectively. Thus the
               NumType feature applies to several different
               parts of speech:

- NUM: cardinal numerals
- DET: quantifiers
- ADJ: definite adjectival, e.g. ordinal numerals
- ADV: adverbial (e.g. ordinal and multiplicative)
   numerals, both definite and pronominal\"

    (http://universaldependencies.org/u/feat/NumType.html)"
         inverse (olia_system:hasFeature) only (ud-
            annotation-model:hasUniversalPartOfSpeech some
            ((ud-annotation-model:ADJ_CLASS or ud-annotation
               -model:ADV_CLASS or ud-annotation-model:
               DET_CLASS or ud-annotation-model:NUM_CLASS)))
                ,
        ud-annotation-model:MorphologicalFeature
```

```
Class: ud-annotation-model:clf

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:aux

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:MoodImp

    EquivalentTo:
        {ud-annotation-model:moodImp}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:Animacy

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:Degree

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:NumberPauc

    EquivalentTo:
        {ud-annotation-model:numberPauc}
```

```
    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:VoiceAntip

    EquivalentTo:
        {ud-annotation-model:voiceAntip}

    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:OtherUniversalPartOfSpeech

    Annotations:
        rdfs:label "Other"

    SubClassOf:
        ud-annotation-model:UniversalPartOfSpeech

    DisjointUnionOf:
        ud-annotation-model:PUNCT_CLASS, ud-annotation-model
            :SYM_CLASS, ud-annotation-model:X_CLASS


Class: <http://www.semanticweb.org/gppassos/ontologies
    /2017/7/ud-annotation-model#nsubj:pass>

    SubClassOf:
        ud-annotation-model:nsubj


Class: ud-annotation-model:VerbFormSup

    EquivalentTo:
        {ud-annotation-model:verbFormSup}
```

```
    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:Reflex

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:SYM_CLASS

    EquivalentTo:
        {ud-annotation-model:SYM}

    SubClassOf:
        ud-annotation-model:OtherUniversalPartOfSpeech


Class: ud-annotation-model:UniversalPartOfSpeech

    SubClassOf:
        ud-annotation-model:PartOfSpeech,
        olia_system:hasTier value "UPOSTAG"

    DisjointUnionOf:
        ud-annotation-model:ClosedClass, ud-annotation-model
            :OpenClass, ud-annotation-model:
            OtherUniversalPartOfSpeech


Class: ud-annotation-model:ModifierWords

    Annotations:
        rdfs:comment "Perhaps this should be \"
            ModifierExpression\"?

(Example: \"very quickly\" in \"John talked very quickly\",
   where we have
```

```
advmod(quickly, very)
advmod(talked, quickly))"

    SubClassOf:
        ud-annotation-model:LinguisticStructure


Class: ud-annotation-model:Person4

    EquivalentTo:
        {ud-annotation-model:person4}

    SubClassOf:
        ud-annotation-model:Person


Class: ud-annotation-model:MoodDes

    EquivalentTo:
        {ud-annotation-model:moodDes}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:PronTypePrs

    EquivalentTo:
        {ud-annotation-model:pronTypePrs}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:Foreign

    SubClassOf:
        ud-annotation-model:MorphologicalFeature
```

```
Class: ud-annotation-model:iobj

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DET_CLASS

    EquivalentTo:
        {ud-annotation-model:DET}

    SubClassOf:
        ud-annotation-model:ClosedClass


Class: ud-annotation-model:cc

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:MorphologicalFeature

    Annotations:
        rdfs:comment "\"Features are additional pieces of
           information about the word, its part of speech
           and morphosyntactic properties. Every feature has
            the form Name=Value and every word can have any
           number of features, separated by the vertical bar
           , as in Gender=Masc|Number=Sing.

       We provide an inventory of features that are attested in
            multiple corpora and it is thus desirable that they
          are encoded in a uniform way. The list is certainly
          not exhaustive and later versions of the standard may
            include new features or values found in new
          languages, corpora or tagsets.\"
```

```
    (http://universaldependencies.org/u/overview/morphology.
       html)"

    SubClassOf:
        olia_system:Feature,
        olia_system:hasTier value "FEATS"

    DisjointUnionOf:
        ud-annotation-model:Abbr, ud-annotation-model:
           Animacy, ud-annotation-model:Aspect, ud-
           annotation-model:Case, ud-annotation-model:
           Definite, ud-annotation-model:Degree, ud-
           annotation-model:Evident, ud-annotation-model:
           Foreign, ud-annotation-model:Gender, ud-
           annotation-model:Mood, ud-annotation-model:
           NumType, ud-annotation-model:Number, ud-
           annotation-model:Person, ud-annotation-model:
           Polarity, ud-annotation-model:Polite, ud-
           annotation-model:Poss, ud-annotation-model:
           PronType, ud-annotation-model:Reflex, ud-
           annotation-model:Tense, ud-annotation-model:
           VerbForm, ud-annotation-model:Voice


Class: ud-annotation-model:punct

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:Polarity

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:ClosedClass

    SubClassOf:
```

```
        ud-annotation-model:UniversalPartOfSpeech

    DisjointUnionOf:
        ud-annotation-model:ADP_CLASS, ud-annotation-model:
            AUX_CLASS, ud-annotation-model:CCONJ_CLASS, ud-
            annotation-model:DET_CLASS, ud-annotation-model:
            NUM_CLASS, ud-annotation-model:PART_CLASS, ud-
            annotation-model:PRON_CLASS, ud-annotation-model:
            SCONJ_CLASS


Class: ud-annotation-model:Person0

    EquivalentTo:
        {ud-annotation-model:person0}

    SubClassOf:
        ud-annotation-model:Person


Class: ud-annotation-model:Person1

    EquivalentTo:
        {ud-annotation-model:person1}

    SubClassOf:
        ud-annotation-model:Person


Class: ud-annotation-model:Person2

    EquivalentTo:
        {ud-annotation-model:person2}

    SubClassOf:
        ud-annotation-model:Person


Class: ud-annotation-model:Person3
```

```
    EquivalentTo:
        {ud-annotation-model:person3}

    SubClassOf:
        ud-annotation-model:Person


Class: owl:Thing


Class: <http://www.semanticweb.org/gppassos/ontologies
    /2017/7/ud-annotation-model#aux:pass>

    SubClassOf:
        ud-annotation-model:aux


Class: ud-annotation-model:DefiniteCons

    EquivalentTo:
        {ud-annotation-model:definiteCons}

    SubClassOf:
        ud-annotation-model:Definite


Class: ud-annotation-model:EvidentFh

    EquivalentTo:
        {ud-annotation-model:evidentFh}

    SubClassOf:
        ud-annotation-model:Evident

    DisjointWith:
        ud-annotation-model:EvidentNfh
```

```
Class: ud-annotation-model:Number

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:VerbFormVnoun

    EquivalentTo:
        {ud-annotation-model:verbFormVnoun}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:ForeignYes

    EquivalentTo:
        {ud-annotation-model:foreignYes}

    SubClassOf:
        ud-annotation-model:Foreign


Class: ud-annotation-model:NumberInv

    EquivalentTo:
        {ud-annotation-model:numberInv}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:NonverbalClause

    Annotations:
        rdfs:comment "\"A nonverbal predicate (nominal or
           adjective) takes a single argument with the nsubj
            relation. The core argument relation is the same
```

```
            regardless of whether there is an overt copula
           linking the predicate to the subject or not.\"

(http://universaldependencies.org/u/overview/simple-syntax.
   html#nonverbal-clauses)"

    EquivalentTo:
        ud-annotation-model:Clause
         and (not (ud-annotation-model:VerbalClause))

    SubClassOf:
        ud-annotation-model:Clause


Class: ud-annotation-model:VerbForm

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:PartOfSpeech

    Annotations:
        rdfs:comment "\"The list of universal POS tags is a
           fixed list containing 17 tags.
    It is possible that some tags will not be used in some
       languages. However, the list cannot be extended to
       cover language-specific extensions. Instead, more
       fine-grained classification of words can be achieved
       via the use of features (see below).

    Also, note that the CoNLL-U format allows an additional
       POSTAG, taken from a language-specific (or corpus-
       specific) tagset. Such language-specific POSTAGs have
        their own data column and are not mixed with the
       universal POS tags.

    The universal POS tags consist of uppercase English
       letters [A-Z] only. Just one tag per word is expected
```

```
        , and it should not be empty. (Use the X tag instead
        of underscore if no other tag is appropriate.)\"
    (http://universaldependencies.org/u/overview/morphology.
        html#part-of-speech-tags)"

    EquivalentTo:
        (ud-annotation-model:LanguageSpecificPartOfSpeech or
            ud-annotation-model:UniversalPartOfSpeech)

    SubClassOf:
        olia_system:Feature


Class: ud-annotation-model:nummod

    SubClassOf:

            Annotations: rdfs:comment "\"Marked as nummod
                but not NUM

DEBUGGING TEST. NONZERO HITS DOES NOT MEAN THE DATA IS
    INVALID. If a word is marked as a numeric modifier, it
    should be marked as a numeral (POS).

Search expression: !NUM <nummod _\"

( http://universaldependencies.org/svalidation.html#marked-
    as-nummod-but-not-num )"
        olia_system:hasTarget some
            (ud-annotation-model:Word
             and (ud-annotation-model:
                 hasUniversalPartOfSpeech some ud-annotation-
                 model:NUM_CLASS)),
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:PronTypeRcp

    EquivalentTo:
```

```
        {ud-annotation-model:pronTypeRcp}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:mark

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DefiniteCom

    EquivalentTo:
        {ud-annotation-model:definiteCom}

    SubClassOf:
        ud-annotation-model:Definite


Class: ud-annotation-model:PoliteElev

    EquivalentTo:
        {ud-annotation-model:politeElev}

    SubClassOf:
        ud-annotation-model:Polite


Class: ud-annotation-model:NUM_CLASS

    EquivalentTo:
        {ud-annotation-model:NUM}

    SubClassOf:
        ud-annotation-model:ClosedClass
```

```
Class: ud-annotation-model:PoliteForm

    EquivalentTo:
        {ud-annotation-model:politeForm}

    SubClassOf:
        ud-annotation-model:Polite


Class: ud-annotation-model:PronTypeExc

    EquivalentTo:
        {ud-annotation-model:pronTypeExc}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:WordUnit

    Annotations:
        rdfs:comment "\"The UD annotation is based on a
           lexicalist view of syntax, which means that
           dependency relations hold between words. Hence,
           morphological features are encoded as properties
           of words and there is no attempt at segmenting
           words into morphemes. However, it is important to
            note that the basic units of annotation are
           syntactic words (not phonological or orthographic
            words), which means that we systematically want
           to split off clitics, as in Spanish dámelo = da
           me lo, and undo contractions, as in French au = à
            le. We refer to such cases as multiword tokens
           because a single orthographic token corresponds
           to multiple (syntactic) words. In exceptional
           cases, it may be necessary to go in the other
           direction, and combine several orthographic
           tokens into a single syntactic word. Starting
           from v2 of the UD guidelines, such multitoken
```

```
            words are allowed for a restricted class of
            phenomena, such as numerical expressions like 20
            000 and abbreviations like e. g., as long as
            these phenomena are approved and clearly
            specified in the language-specific documentation.
             Note, however, that this technique should not be
             generalized to multiword expressions like in
            spite of and by and large (let alone to more
            flexible multiword expressions like compounds or
            particle verbs), which should instead be
            annotated using special dependency relations.\"
    ( http://universaldependencies.org/u/overview/
       tokenization.html )"

    SubClassOf:
        olia_system:UnitOfAnnotation,
        ud-annotation-model:isTargetOf exactly 1 ud-
           annotation-model:DependencyRelation,
        ud-annotation-model:inSentence exactly 1 ud-
           annotation-model:SentenceUnit


Class: ud-annotation-model:ADJ_CLASS

    EquivalentTo:
        {ud-annotation-model:ADJ}

    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:PUNCT_CLASS

    EquivalentTo:
        {ud-annotation-model:PUNCT}

    SubClassOf:
        ud-annotation-model:OtherUniversalPartOfSpeech
```

```
Class: ud-annotation-model:AspectHab

    EquivalentTo:
        {ud-annotation-model:aspectHab}

    SubClassOf:
        ud-annotation-model:Aspect


Class: ud-annotation-model:MoodOpt

    EquivalentTo:
        {ud-annotation-model:moodOpt}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:PronTypeDem

    EquivalentTo:
        {ud-annotation-model:pronTypeDem}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:TensePqp

    EquivalentTo:
        {ud-annotation-model:tensePqp}

    SubClassOf:
        ud-annotation-model:Tense


Class: ud-annotation-model:PRON_CLASS
```

```
    EquivalentTo:
        {ud-annotation-model:PRON}

    SubClassOf:
        ud-annotation-model:ClosedClass


Class: ud-annotation-model:Clause

    SubClassOf:
        ud-annotation-model:LinguisticStructure

    DisjointUnionOf:
        ud-annotation-model:ComplexClause, ud-annotation-
            model:SimpleClause

    DisjointUnionOf:
        ud-annotation-model:NonverbalClause, ud-annotation-
            model:VerbalClause


Class: ud-annotation-model:NOUN_CLASS

    EquivalentTo:
        {ud-annotation-model:NOUN}

    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:VerbFormConv

    EquivalentTo:
        {ud-annotation-model:verbFormConv}

    SubClassOf:
        ud-annotation-model:VerbForm
```

```
Class: ud-annotation-model:amod

    Annotations:
        rdfs:comment "\"An amod is an adjective modifying
           the head of a nominal phrase.\"

    ( http://universaldependencies.org/u/overview/nominal-
       syntax.html#modifier-dependents )"

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:PronTypeRel

    EquivalentTo:
        {ud-annotation-model:pronTypeRel}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:csubj

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:CaseErg

    EquivalentTo:
        {ud-annotation-model:caseErg}

    SubClassOf:
        ud-annotation-model:Case


Class: ud-annotation-model:dep
```

```
    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:TenseImp

    EquivalentTo:
        {ud-annotation-model:tenseImp}

    SubClassOf:
        ud-annotation-model:Tense


Class: ud-annotation-model:det

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:MoodPot

    EquivalentTo:
        {ud-annotation-model:moodPot}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-theory:NonverbalClauseHead

    Annotations:
        rdfs:comment "\"A nonverbal predicate (nominal or
           adjective) takes a single argument with the nsubj
            relation. The core argument relation is the same
            regardless of whether there is an overt copula
           linking the predicate to the subject or not.\"

(http://universaldependencies.org/u/overview/simple-syntax.
   html#nonverbal-clauses)"
```

```
    EquivalentTo:
        ud-annotation-model:ClauseHead
         and (ud-annotation-model:hasUniversalPartOfSpeech
            some (not (ud-annotation-model:VERB_CLASS)))

    SubClassOf:
        ud-annotation-model:isSourceOf exactly 1 ud-
           annotation-model:nsubj,
        ud-annotation-model:ClauseHead,

            Annotations: rdfs:comment "\"A nonverbal
               predicate (nominal or adjective) takes a
               single argument with the nsubj relation. The
               core argument relation is the same regardless
                of whether there is an overt copula linking
               the predicate to the subject or not.\"

(http://universaldependencies.org/u/overview/simple-syntax.
   html#nonverbal-clauses)"
        <http://org.semanticweb.owlapi/error#Error3>


Class: ud-annotation-model:Voice

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:VoiceRcp

    EquivalentTo:
        {ud-annotation-model:voiceRcp}

    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:cop
```

```
    SubClassOf:

            Annotations: rdfs:comment "\"A cop (copula) is
               the relation of a function word used to link
               a subject to a nonverbal predicate. It is
               often a verb but nonverbal copulas are also
               frequent in the world's languages. \"

( http://universaldependencies.org/u/dep/cop.html )"
        olia_system:hasSource some ud-theory:
           NonverbalClauseHead,
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:AspectProsp

    EquivalentTo:
        {ud-annotation-model:aspectProsp}

    SubClassOf:
        ud-annotation-model:Aspect


Class: ud-annotation-model:PronType

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:PronTypeNeg

    EquivalentTo:
        {ud-annotation-model:pronTypeNeg}

    SubClassOf:
        ud-annotation-model:PronType
```

```
Class: ud-annotation-model:flat

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:IntransitiveVerbalClause

    SubClassOf:
        ud-annotation-model:VerbalClause


Class: ud-annotation-model:OpenClass

    SubClassOf:
        ud-annotation-model:UniversalPartOfSpeech

    DisjointUnionOf:
        ud-annotation-model:ADJ_CLASS, ud-annotation-model:
            ADV_CLASS, ud-annotation-model:INTJ_CLASS, ud-
            annotation-model:NOUN_CLASS, ud-annotation-model:
            PROPN_CLASS, ud-annotation-model:VERB_CLASS


Class: ud-annotation-model:NumTypeFrac

    EquivalentTo:
        {ud-annotation-model:numTypeFrac}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:VoiceCau

    EquivalentTo:
        {ud-annotation-model:voiceCau}

    SubClassOf:
```

```
        ud-annotation-model:Voice


Class: ud-annotation-model:CaseAcc

    EquivalentTo:
        {ud-annotation-model:caseAcc}

    SubClassOf:
        ud-annotation-model:Case


Class: ud-annotation-model:AspectIter

    EquivalentTo:
        {ud-annotation-model:aspectIter}

    SubClassOf:
        ud-annotation-model:Aspect


Class: olia_system:Feature

    Annotations:
        rdfs:comment "UnitsOfAnnotation and Relations can be
            described in a more detailed way by the features
            that are attached to them, e.g., case, number,
           or aspect. Features are, however, not subject to
           further annotations (by means of hasFeature),
           they are thus disjoint from Relations and
           UnitsOfAnnotation."^^xsd:string

    SubClassOf:
        olia_system:LinguisticAnnotation


Class: ud-annotation-model:CoordinatedFunctionWords

    Annotations:
```

```
        rdfs:comment "\"Head coordination is a syntactic
           process that can apply to almost any word
           category, including function words like
           conjunctions and prepositions. In such cases, the
            standard analysis of coordination is used and
           function words have dependents.\"

(http://universaldependencies.org/u/overview/syntax.html#
   coordinated-function-words)"

    SubClassOf:
        ud-annotation-model:Coordination


Class: ud-annotation-model:DefiniteDef

    EquivalentTo:
        {ud-annotation-model:definiteDef}

    SubClassOf:
        ud-annotation-model:Definite


Class: ud-annotation-model:MoodPrp

    EquivalentTo:
        {ud-annotation-model:moodPrp}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:AUX_CLASS

    EquivalentTo:
        {ud-annotation-model:AUX}

    SubClassOf:
        ud-annotation-model:ClosedClass
```

```
Class: ud-annotation-model:CaseNom

    EquivalentTo:
        {ud-annotation-model:caseNom}

    SubClassOf:
        ud-annotation-model:Case


Class: ud-annotation-model:MultiwordExpression

    SubClassOf:
        ud-annotation-model:Concept


Class: ud-annotation-model:dislocated

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:Gender

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:DependencySubtree

    Annotations:
        rdfs:comment "Consists of WordUnit individuals.",
        rdfs:comment "Given a WordUnit individual, the set
           whose members are this individual and its
           descendents in the DependencyRelation."

    SubClassOf:
        ud-annotation-model:Concept
```

```
Class: ud-annotation-model:obl

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:Case

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:vocative

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:CaseAbs

    EquivalentTo:
        {ud-annotation-model:caseAbs}

    SubClassOf:
        ud-annotation-model:Case


Class: ud-annotation-model:obj

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:SimpleClause

    SubClassOf:
        ud-annotation-model:Clause
```

```
Class: ud-annotation-model:VERB_CLASS

    EquivalentTo:
        {ud-annotation-model:VERB}

    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:AnimacyInan

    EquivalentTo:
        {ud-annotation-model:animacyInan}

    SubClassOf:
        ud-annotation-model:Animacy


Class: ud-annotation-model:NumberDual

    EquivalentTo:
        {ud-annotation-model:numberDual}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:PronTypeTot

    EquivalentTo:
        {ud-annotation-model:pronTypeTot}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:MoodCnd
```

```
    EquivalentTo:
        {ud-annotation-model:moodCnd}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:NumTypeOrd

    EquivalentTo:
        {ud-annotation-model:numTypeOrd}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:TensePast

    EquivalentTo:
        {ud-annotation-model:tensePast}

    SubClassOf:
        ud-annotation-model:Tense


Class: ud-annotation-model:TensePres

    EquivalentTo:
        {ud-annotation-model:tensePres}

    SubClassOf:
        ud-annotation-model:Tense


Class: ud-annotation-model:PROPN_CLASS

    EquivalentTo:
        {ud-annotation-model:PROPN}
```

```
    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:PronTypeArt

    EquivalentTo:
        {ud-annotation-model:pronTypeArt}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:SentenceUnit

    SubClassOf:
        olia_system:UnitOfAnnotation,
        ud-annotation-model:hasWord exactly 1 (ud-annotation
            -model:isTargetOf some ud-annotation-model:root)


Class: ud-annotation-model:NumberGrpl

    EquivalentTo:
        {ud-annotation-model:numberGrpl}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:root

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:ccomp
```

```
    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DegreeSup

    EquivalentTo:
        {ud-annotation-model:degreeSup}

    SubClassOf:
        ud-annotation-model:Degree


Class: ud-annotation-model:NominalHead

    SubClassOf:
        ud-annotation-model:Word,

            Annotations: ud-theory:verify "\"Ellipsis\"
               should not be exactly a word, so this
               modelling should be revised.",
                         rdfs:comment "\"A nominal head does
                             not take any core arguments but
                             may be associated with
                            different types of modifiers:\"

( http://universaldependencies.org/u/overview/nominal-syntax
   .html )"
        (ud-annotation-model:Ellipsis or (not (ud-annotation
           -model:isSourceOf some (olia_system:hasTarget
           some ud-annotation-model:CoreArgument))))


Class: <http://www.semanticweb.org/gppassos/ontologies
   /2017/7/ud-annotation-model#flat:name>

    SubClassOf:
        ud-annotation-model:flat
```

```
Class: ud-annotation-model:Evident

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:MoodInd

    EquivalentTo:
        {ud-annotation-model:moodInd}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:NumTypeRange

    EquivalentTo:
        {ud-annotation-model:numTypeRange}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:NumberGrpa

    EquivalentTo:
        {ud-annotation-model:numberGrpa}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:CoreArgument

    SubClassOf:
        ud-annotation-model:ContentWord
```

```
    DisjointWith:
        ud-annotation-model:ObliqueModifier


Class: ud-annotation-model:MoodJus

    EquivalentTo:
        {ud-annotation-model:moodJus}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:MultiWordFunctionExpression

    Annotations:
        rdfs:comment "\"The word forms that make up a fixed
            function-word multiword expression (MWE) are
            connected using the special dependency relation
            fixed. By convention, the first word is always
            taken as the head, so when the multiword
            expression is a functional element, the initial
            word form will then superficially look like a
            function word with dependents.\"

(http://universaldependencies.org/u/overview/syntax.html#
   multiword-function-words)"

    SubClassOf:
        ud-annotation-model:MultiwordExpression


Class: ud-annotation-model:PART_CLASS

    EquivalentTo:
        {ud-annotation-model:PART}

    SubClassOf:
        ud-annotation-model:ClosedClass
```

```
Class: ud-annotation-model:PronTypeInd

    EquivalentTo:
        {ud-annotation-model:pronTypeInd}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:AspectImp

    EquivalentTo:
        {ud-annotation-model:aspectImp}

    SubClassOf:
        ud-annotation-model:Aspect


Class: ud-annotation-model:PronTypeEmp

    EquivalentTo:
        {ud-annotation-model:pronTypeEmp}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:acl

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:VoiceInv

    EquivalentTo:
        {ud-annotation-model:voiceInv}
```

```
    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:MoodNec

    EquivalentTo:
        {ud-annotation-model:moodNec}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:VerbFormFin

    EquivalentTo:
        {ud-annotation-model:verbFormFin}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:PronTypeInt

    EquivalentTo:
        {ud-annotation-model:pronTypeInt}

    SubClassOf:
        ud-annotation-model:PronType


Class: ud-annotation-model:NumTypeMult

    EquivalentTo:
        {ud-annotation-model:numTypeMult}

    SubClassOf:
        ud-annotation-model:NumType
```

```
Class: ud-annotation-model:DegreeCmp

    EquivalentTo:
        {ud-annotation-model:degreeCmp}

    SubClassOf:
        ud-annotation-model:Degree


Class: ud-annotation-model:Abbr

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:TransitiveVerbalClause

    SubClassOf:
        ud-annotation-model:VerbalClause


Class: ud-annotation-model:fixed

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:NumberPlur

    EquivalentTo:
        {ud-annotation-model:numberPlur}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:appos
```

```
    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:reparandum

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:PoliteHumb

    EquivalentTo:
        {ud-annotation-model:politeHumb}

    SubClassOf:
        ud-annotation-model:Polite


Class: ud-annotation-model:ADV_CLASS

    EquivalentTo:
        {ud-annotation-model:ADV}

    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:GenderNeut

    EquivalentTo:
        {ud-annotation-model:genderNeut}

    SubClassOf:
        ud-annotation-model:Gender


Class: ud-annotation-model:ContentWord
```

```
    SubClassOf:
        ud-annotation-model:Word


Class: ud-annotation-model:NumTypeSets

    EquivalentTo:
        {ud-annotation-model:numTypeSets}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:AnimacyNhum

    EquivalentTo:
        {ud-annotation-model:animacyNhum}

    SubClassOf:
        ud-annotation-model:Animacy


Class: ud-annotation-model:Sentence

    SubClassOf:
        ud-annotation-model:Concept


Class: ud-annotation-model:Lemma

    Annotations:
        rdfs:comment "\"The LEMMA field should contain the
           canonical or base form of the word, such as the
           form typically found in dictionaries.

    If the lemma is not available, an underscore (''_'') can
        be used to indicate its absence.
```

```
    The LEMMA field should not be used to encode features or
        other similar properties of the word (use FEATS and
       MISC instead; see format).\"
    (http://universaldependencies.org/u/overview/morphology.
       html#lemmas)"

    SubClassOf:
        olia_system:hasTier value "LEMMA",
        olia_system:Feature


Class: ud-annotation-model:orphan

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:SCONJ_CLASS

    EquivalentTo:
        {ud-annotation-model:SCONJ}

    SubClassOf:
        ud-annotation-model:ClosedClass


Class: ud-annotation-model:VerbFormPart

    EquivalentTo:
        {ud-annotation-model:verbFormPart}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:advmod

    SubClassOf:
```

```
            Annotations: rdfs:comment "\"An adverbial
               modifier of a word is a (non-clausal) adverb
               or adverbial phrase that serves to modify a
               predicate or a modifier word.\"

http://universaldependencies.org/u/dep/advmod.html#advmod-
   adverbial-modifier

However, there are some problems, such as function word
   modifiers, as in the example in the documentation:
  \"not every linguist\"
      det(linguist,every)
      advmod(every,not)

In this case, \"every\" is a function word, not a (content)
   modifier word.

http://universaldependencies.org/u/overview/syntax.html#
   function-word-modifiers

In this case, this axiom could be weakened by transforming
   it into:
  advmod SubClassOf hasSource some (Clause Head of
     ModifierWord of FunctionWord)"
        olia_system:hasSource some
            ((ud-annotation-model:ClauseHead or ud-
               annotation-model:ModifierWord)),
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:goeswith

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:MoodQot

    EquivalentTo:
```

```
        {ud-annotation-model:moodQot}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:LanguageSpecificPartOfSpeech

    SubClassOf:
        olia_system:hasTier value "XPOSTAG",
        ud-annotation-model:PartOfSpeech


Class: ud-annotation-model:PoliteInfm

    EquivalentTo:
        {ud-annotation-model:politeInfm}

    SubClassOf:
        ud-annotation-model:Polite


Class: ud-annotation-model:LinguisticStructure

    SubClassOf:
        ud-annotation-model:Concept


Class: ud-annotation-model:VerbalClause

    SubClassOf:
        ud-annotation-model:Clause


Class: ud-annotation-model:VoiceDir

    EquivalentTo:
        {ud-annotation-model:voiceDir}
```

```
    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:LayeredFeature

    Annotations:
        rdfs:comment "\"In some languages, some features are
            marked more than once on the same word. We say
           that there are several layers of the feature. The
            exact meaning of individual layers is language-
           dependent.

    For example, possessive adjectives, determiners and
       pronouns may have two different values of Gender and
       two of Number. One of the values is determined by
       agreement with the modified (possessed) noun. This is
        parallel to other (non-possessive) adjectives and
       determiners that agree in gender and number with the
       nouns they modify. The other value is determined
       lexically because it is a property of the possessor
       .\"
    (http://universaldependencies.org/u/overview/morphology.
       html#layered-features)"

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:GenderFem

    EquivalentTo:
        {ud-annotation-model:genderFem}

    SubClassOf:
        ud-annotation-model:Gender


Class: ud-annotation-model:Person
```

```
    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:INTJ_CLASS

    EquivalentTo:
        {ud-annotation-model:INTJ}

    SubClassOf:
        ud-annotation-model:OpenClass


Class: ud-annotation-model:Aspect

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:AbbrYes

    EquivalentTo:
        {ud-annotation-model:abbrYes}

    SubClassOf:
        ud-annotation-model:Abbr


Class: ud-annotation-model:AnimacyAnim

    EquivalentTo:
        {ud-annotation-model:animacyAnim}

    SubClassOf:
        ud-annotation-model:Animacy


Class: ud-annotation-model:VoiceAct
```

```
    EquivalentTo:
        {ud-annotation-model:voiceAct}

    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:expl

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:MoodAdm

    EquivalentTo:
        {ud-annotation-model:moodAdm}

    SubClassOf:
        ud-annotation-model:Mood


Class: ud-annotation-model:Mood

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:NumberSing

    EquivalentTo:
        {ud-annotation-model:numberSing}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:GenderMasc
```

```
    EquivalentTo:
        {ud-annotation-model:genderMasc}

    SubClassOf:
        ud-annotation-model:Gender


Class: ud-annotation-model:NumTypeCard

    EquivalentTo:
        {ud-annotation-model:numTypeCard}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:NumberTri

    EquivalentTo:
        {ud-annotation-model:numberTri}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:AspectPerf

    EquivalentTo:
        {ud-annotation-model:aspectPerf}

    SubClassOf:
        ud-annotation-model:Aspect


Class: <http://org.semanticweb.owlapi/error#Error3>


Class: ud-annotation-model:Concept
```

```
Class: ud-annotation-model:Tense

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:NominalPhrase

    SubClassOf:
        ud-annotation-model:LinguisticStructure


Class: ud-annotation-model:case

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DependencyRelation

    SubClassOf:
        olia_system:hasSource exactly 1 ud-annotation-model:
            Word,
        olia_system:hasTarget exactly 1 ud-annotation-model:
            Word,
        olia_system:Relation


Class: ud-annotation-model:parataxis

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DegreePos

    EquivalentTo:
```

```
        {ud-annotation-model:degreePos}

    SubClassOf:
        ud-annotation-model:Degree


Class: ud-annotation-model:NumTypeDist

    EquivalentTo:
        {ud-annotation-model:numTypeDist}

    SubClassOf:
        ud-annotation-model:NumType


Class: ud-annotation-model:conj

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:AnimacyHum

    EquivalentTo:
        {ud-annotation-model:animacyHum}

    SubClassOf:
        ud-annotation-model:Animacy


Class: ud-annotation-model:NumberCount

    EquivalentTo:
        {ud-annotation-model:numberCount}

    SubClassOf:
        ud-annotation-model:Number
```

```
Class: ud-annotation-model:FunctionWord

    SubClassOf:
        ud-annotation-model:Word,

            Annotations: ud-theory:verify "\"Ellipsis\"
               should not be exactly a word, so this
               modelling should be revised.",
                        rdfs:comment "\"Nevertheless, there
                            are four important exceptions
                           to the rule that function words
                           do not take dependents:

    Multiword function words
    Coordinated function words
    Function word modifiers
    Promotion by head elision\"

( http://universaldependencies.org/u/overview/syntax.html )"
        (ud-annotation-model:Ellipsis or (ud-annotation-
           model:isSourceOf only
            ((ud-annotation-model:advcl or ud-annotation-
               model:advmod or ud-annotation-model:amod or
               ud-annotation-model:cc or ud-annotation-model
               :conj or ud-annotation-model:fixed))))


Class: ud-annotation-model:VoiceMid

    EquivalentTo:
        {ud-annotation-model:voiceMid}

    SubClassOf:
        ud-annotation-model:Voice


Class: ud-annotation-model:Word

    EquivalentTo:
```

```
        (ud-annotation-model:ClauseHead or ud-annotation-
            model:FunctionWord or ud-annotation-model:
            ModifierWord or ud-annotation-model:NominalHead)

    SubClassOf:

        Annotations: rdfs:comment "This is never
            explicitly stated in the documentation.

However, it is stated that:
\"The object of a verb is the second most core argument of a
    verb after the subject. Typically, it is the noun phrase
    that denotes the entity acted upon or which undergoes a
   change of state or motion (the proto-patient).\" ( http
   ://universaldependencies.org/u/dep/obj.html )

This suggests that the object is unique. Furthermore:

\"A clausal complement of a verb or adjective is a dependent
    clause which is a core argument. That is, it functions
   like an object of the verb, or adjective.\" ( http://
   universaldependencies.org/u/dep/ccomp.html )

and
\"In general, if there is just one object, it should be
   labeled obj, regardless of the morphological case or
   semantic role that it bears. If there are two or more
   objects, one of them should be obj and the others should
   be iobj. In such cases it is necessary to decide what is
   the most directly affected object (patient).\" ( http://
   universaldependencies.org/u/dep/obj.html )

These suggest that there should not be more than one direct
   object, and that ccomp is an object.

Besides, this is in the svalidations as a warning."
        ud-annotation-model:isSourceOf max 1 ((ud-annotation
            -model:csubj or ud-annotation-model:nsubj)),
```

```
            Annotations: rdfs:comment "This is never
               explicitly stated on the documentation.
               However, some possible annotation guidelines
               are rejected as they would create words with
               two subjects (for instance, see discussion on
                copula in equational constructions involving
                full clauses: http://universaldependencies.
               org/u/overview/complex-syntax.html#clausal-
               complements-objects )

Besides, in some descriptions a subject is referred to as \"
   the\" subject, what implies in uniqueness:
\"A nominal subject (nsubj) is a nominal which is the
   syntactic subject and the proto-agent of a clause.\"

This is also in a svalidation as a warning."
        ud-annotation-model:isSourceOf max 1 ((ud-annotation
           -model:ccomp or ud-annotation-model:obj)),
        olia_system:hasFeature exactly 1 ud-annotation-model
           :UniversalPartOfSpeech,
        ud-annotation-model:Concept

    DisjointUnionOf:
        ud-annotation-model:ContentWord, ud-annotation-model
           :FunctionWord


Class: olia_system:UnitOfAnnotation

    Annotations:
        rdfs:comment "A UnitOfAnnotation is a structural
           entity that can be annotated, e.g., a word, token
           , constituent, or another types of markable.
Word classes, etc., are then modelled as indirect children
   of UnitOfAnnotation. The division between Features and
   classes of UnitsOfAnnotation follows conventional
   standards.
    "^^xsd:string
```

```
    SubClassOf:
        olia_system:LinguisticAnnotation


Class: ud-annotation-model:compound

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:Polite

    SubClassOf:
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:ReflexYes

    EquivalentTo:
        {ud-annotation-model:reflexYes}

    SubClassOf:
        ud-annotation-model:Reflex


Class: olia_system:LinguisticAnnotation

    Annotations:
        rdfs:comment "label: Text attached to an element",
        rdfs:comment "Linguistic annotations pertain to
           either structural entities (words, tokens,
           constituents, sentences => UnitOfAnnotation),
           relations between these (dependency, dominance,
           coreference, etc. => Relation), or grammatical
           features attached to these (case, gender, number,
            agreement, tense, mood, aspect, ... => Feature)
           ."^^xsd:string,
        rdfs:isDefinedBy "http://www.isocat.org/datcat/DC
           -1857"
```

```
Class: ud-annotation-model:nmod

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:ObliqueModifier

    SubClassOf:
        ud-annotation-model:ContentWord

    DisjointWith:
        ud-annotation-model:CoreArgument


Class: ud-annotation-model:advcl

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:PossYes

    EquivalentTo:
        {ud-annotation-model:possYes}

    SubClassOf:
        ud-annotation-model:Poss


Class: ud-annotation-model:ClauseHead

    SubClassOf:
        ud-annotation-model:Word


Class: ud-annotation-model:DegreeEqu
```

```
    EquivalentTo:
        {ud-annotation-model:degreeEqu}

    SubClassOf:
        ud-annotation-model:Degree


Class: ud-annotation-model:ModifierWord

    SubClassOf:
        ud-annotation-model:Word


Class: ud-annotation-model:DefiniteSpec

    EquivalentTo:
        {ud-annotation-model:definiteSpec}

    SubClassOf:
        ud-annotation-model:Definite


Class: ud-annotation-model:TenseFut

    EquivalentTo:
        {ud-annotation-model:tenseFut}

    SubClassOf:
        ud-annotation-model:Tense


Class: ud-annotation-model:xcomp

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:DegreeAbs
```

```
    EquivalentTo:
        {ud-annotation-model:degreeAbs}

    SubClassOf:
        ud-annotation-model:Degree


Class: ud-annotation-model:NumberPtan

    EquivalentTo:
        {ud-annotation-model:numberPtan}

    SubClassOf:
        ud-annotation-model:Number


Class: ud-annotation-model:VerbFormGdv

    EquivalentTo:
        {ud-annotation-model:verbFormGdv}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:X_CLASS

    EquivalentTo:
        {ud-annotation-model:X}

    SubClassOf:
        ud-annotation-model:OtherUniversalPartOfSpeech


Class: ud-annotation-model:DefiniteInd

    EquivalentTo:
        {ud-annotation-model:definiteInd}
```

```
    SubClassOf:
        ud-annotation-model:Definite


Class: ud-annotation-model:Coordination

    SubClassOf:
        ud-annotation-model:Concept


Class: ud-theory:VerbalClauseHead

    EquivalentTo:
        ud-annotation-model:ClauseHead
         and (ud-annotation-model:hasUniversalPartOfSpeech
            some ud-annotation-model:VERB_CLASS)

    SubClassOf:
        ud-annotation-model:ClauseHead


Class: ud-annotation-model:Poss

    SubClassOf:
         inverse (olia_system:hasFeature) only
            ((ud-annotation-model:ADJ_CLASS or ud-annotation
               -model:DET_CLASS or (ud-annotation-model:
               hasUniversalPartOfSpeech some ud-annotation-
               model:PRON_CLASS))),
        ud-annotation-model:MorphologicalFeature


Class: ud-annotation-model:MoodSub

    EquivalentTo:
        {ud-annotation-model:moodSub}

    SubClassOf:
```

```
        ud-annotation-model:Mood


Class: ud-annotation-model:list

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: <http://www.semanticweb.org/gppassos/ontologies
    /2017/7/ud-annotation-model#acl:relcl>

    SubClassOf:
        ud-annotation-model:acl


Class: ud-annotation-model:PolarityNeg

    EquivalentTo:
        {ud-annotation-model:polarityNeg}

    SubClassOf:
        ud-annotation-model:Polarity

    DisjointWith:
        ud-annotation-model:PolarityPos


Class: ud-annotation-model:VoicePass

    EquivalentTo:
        {ud-annotation-model:voicePass}

    SubClassOf:
        ud-annotation-model:Voice


Class: olia_system:Relation
```

```
    Annotations:
        rdfs:comment "Between instances of two Categories, a
            Relation can be established that links these
           together, e.g., a dominance relation (constituent
            X is grammatical subject of sentence Y), a
           dependency relation (token X is a modifier of
           token Y), a discourse relation (discourse unit X
           is in contrast to discourse unit Y), an anaphoric
            relation (referring expression X is coreferent
           with referring expressing Y), an alignment
           relation (word X expresses the same meaning as
           word Y).
Note that Relation and UnitOfAnnotation are not disjoint,
    because in some approaches, establishing a Relation
    between two concepts entails that a structural entity is
    formed, consisting of Relation and the Categories
    connected by the Relation, e.g., in Rhetorical Structure
    Theory (Mann and Thompson 1987)."^^xsd:string

    EquivalentTo:
        ((olia_system:hasSource min 1 owl:Thing) or (
           olia_system:hasTarget min 1 owl:Thing))

    SubClassOf:
        olia_system:LinguisticAnnotation,
        olia_system:hasTarget exactly 1 owl:Thing,
        olia_system:hasSource exactly 1 owl:Thing


Class: ud-annotation-model:VerbFormGer

    EquivalentTo:
        {ud-annotation-model:verbFormGer}

    SubClassOf:
        ud-annotation-model:VerbForm


Class: ud-annotation-model:PolarityPos
```

```
    EquivalentTo:
        {ud-annotation-model:polarityPos}

    SubClassOf:
        ud-annotation-model:Polarity

    DisjointWith:
        ud-annotation-model:PolarityNeg


Class: ud-annotation-model:GenderCom

    EquivalentTo:
        {ud-annotation-model:genderCom}

    SubClassOf:
        ud-annotation-model:Gender


Class: ud-annotation-model:nsubj

    SubClassOf:
        ud-annotation-model:DependencyRelation


Class: ud-annotation-model:AspectProg

    EquivalentTo:
        {ud-annotation-model:aspectProg}

    SubClassOf:
        ud-annotation-model:Aspect


Class: ud-annotation-model:ADP_CLASS

    EquivalentTo:
        {ud-annotation-model:ADP}
```

```
    SubClassOf:
        ud-annotation-model:ClosedClass


Class: ud-annotation-model:NumberColl

    EquivalentTo:
        {ud-annotation-model:numberColl}

    SubClassOf:
        ud-annotation-model:Number


Individual: ud-annotation-model:verbFormGer


Individual: ud-annotation-model:degreeAbs


Individual: ud-annotation-model:animacyAnim


Individual: ud-annotation-model:numTypeFrac


Individual: ud-annotation-model:voiceMid


Individual: ud-annotation-model:pronTypeExc


Individual: ud-annotation-model:pronTypeArt


Individual: ud-annotation-model:pronTypeRel


Individual: ud-annotation-model:SYM
```

```
    Types:
        ud-annotation-model:SYM_CLASS


Individual: ud-annotation-model:polarityPos


Individual: ud-annotation-model:verbFormGdv


Individual: ud-annotation-model:voiceRcp


Individual: ud-annotation-model:PROPN

    Types:
        ud-annotation-model:PROPN_CLASS


Individual: ud-annotation-model:numberTri


Individual: ud-annotation-model:degreeSup


Individual: ud-annotation-model:AUX

    Types:
        ud-annotation-model:AUX_CLASS


Individual: ud-annotation-model:voiceDir


Individual: ud-annotation-model:numTypeRange


Individual: ud-annotation-model:tensePqp
```

```
Individual: ud-annotation-model:CCONJ

    Types:
        ud-annotation-model:CCONJ_CLASS


Individual: ud-annotation-model:verbFormInf


Individual: ud-annotation-model:ADP

    Types:
        ud-annotation-model:ADP_CLASS


Individual: ud-annotation-model:numberCount


Individual: ud-annotation-model:person1


Individual: ud-annotation-model:person0


Individual: ud-annotation-model:moodPrp


Individual: ud-annotation-model:ADV

    Types:
        ud-annotation-model:ADV_CLASS


Individual: ud-annotation-model:ADJ

    Types:
        ud-annotation-model:ADJ_CLASS
```

Individual: ud-annotation-model:numTypeCard

Individual: ud-annotation-model:person3

Individual: ud-annotation-model:person2

Individual: ud-annotation-model:degreePos

Individual: ud-annotation-model:person4

Individual: ud-annotation-model:voiceAct

Individual: ud-annotation-model:numberInv

Individual: ud-annotation-model:voicePass

Individual: ud-annotation-model:pronTypeRcp

Individual: ud-annotation-model:evidentFh

Individual: ud-annotation-model:numTypeSets

Individual: ud-annotation-model:numberPlur

Individual: ud-annotation-model:degreeCmp

Individual: ud-annotation-model:aspectPerf

Individual: ud-annotation-model:pronTypeEmp

Individual: ud-annotation-model:definiteDef

Individual: ud-annotation-model:possYes

Individual: ud-annotation-model:definiteCom

Individual: ud-annotation-model:SCONJ

Types:
ud-annotation-model:SCONJ_CLASS

Individual: ud-annotation-model:politeForm

Individual: ud-annotation-model:definiteSpec

Individual: ud-annotation-model:definiteCons

Individual: ud-annotation-model:animacyHum

Individual: ud-annotation-model:numberPtan

Individual: ud-annotation-model:PUNCT

```
    Types:
        ud-annotation-model:PUNCT_CLASS

Individual: ud-annotation-model:tensePres

Individual: ud-annotation-model:animacyInan

Individual: ud-annotation-model:verbFormFin

Individual: ud-annotation-model:moodInd

Individual: ud-annotation-model:aspectProsp

Individual: ud-annotation-model:genderMasc

Individual: ud-annotation-model:pronTypeDem

Individual: ud-annotation-model:politeInfm

Individual: ud-annotation-model:verbFormVnoun

Individual: ud-annotation-model:pronTypeInt

Individual: ud-annotation-model:abbrYes

Individual: ud-annotation-model:moodSub
```

```
Individual: ud-annotation-model:VERB

    Types:
        ud-annotation-model:VERB_CLASS


Individual: ud-annotation-model:aspectProg


Individual: ud-annotation-model:genderNeut


Individual: ud-annotation-model:moodImp


Individual: ud-annotation-model:moodPot


Individual: ud-annotation-model:genderCom


Individual: ud-annotation-model:moodAdm


Individual: ud-annotation-model:moodQot


Individual: ud-annotation-model:pronTypeInd


Individual: ud-annotation-model:verbFormConv


Individual: ud-annotation-model:verbFormSup


Individual: ud-annotation-model:definiteInd
```

```
Individual: ud-annotation-model:numTypeDist

Individual: ud-annotation-model:numberPauc

Individual: ud-annotation-model:INTJ

    Types:
        ud-annotation-model:INTJ_CLASS

Individual: ud-annotation-model:evidentNfh

Individual: ud-annotation-model:pronTypePrs

Individual: ud-annotation-model:NUM

    Types:
        ud-annotation-model:NUM_CLASS

Individual: ud-annotation-model:moodJus

Individual: ud-annotation-model:voiceAntip

Individual: ud-annotation-model:reflexYes

Individual: ud-annotation-model:polarityNeg

Individual: ud-annotation-model:politeHumb
```

```
Individual: ud-annotation-model:voiceCau

Individual: ud-annotation-model:numTypeMult

Individual: ud-annotation-model:PRON

    Types:
        ud-annotation-model:PRON_CLASS

Individual: ud-annotation-model:PART

    Types:
        ud-annotation-model:PART_CLASS

Individual: ud-annotation-model:foreignYes

Individual: ud-annotation-model:tensePast

Individual: ud-annotation-model:numberColl

Individual: ud-annotation-model:genderFem

Individual: ud-annotation-model:animacyNhum

Individual: ud-annotation-model:caseAcc

Individual: ud-annotation-model:pronTypeTot
```

Individual: ud-annotation-model:moodDes

Individual: ud-annotation-model:moodNec

Individual: ud-annotation-model:caseErg

Individual: ud-annotation-model:pronTypeNeg

Individual: ud-annotation-model:numTypeOrd

Individual: ud-annotation-model:caseAbs

Individual: ud-annotation-model:numberSing

Individual: ud-annotation-model:moodCnd

Individual: ud-annotation-model:numberGrpl

Individual: ud-annotation-model:X

    Types:
        ud-annotation-model:X_CLASS

Individual: ud-annotation-model:degreeEqu

Individual: ud-annotation-model:verbFormPart

```
Individual: ud-annotation-model:moodOpt

Individual: ud-annotation-model:tenseFut

Individual: ud-annotation-model:aspectImp

Individual: ud-annotation-model:numberGrpa

Individual: ud-annotation-model:caseNom

Individual: ud-annotation-model:aspectIter

Individual: ud-annotation-model:politeElev

Individual: ud-annotation-model:tenseImp

Individual: ud-annotation-model:numberDual

Individual: ud-annotation-model:voiceInv

Individual: ud-annotation-model:aspectHab

Individual: ud-annotation-model:DET

    Types:
        ud-annotation-model:DET_CLASS
```

```
Individual: ud-annotation-model:NOUN

    Types:
        ud-annotation-model:NOUN_CLASS


Individual: _:genid2908

    Annotations:
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger


Individual: _:genid2484

    Annotations:
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger


DisjointClasses:
    ud-annotation-model:Abbr, ud-annotation-model:Animacy, ud-
        annotation-model:Aspect, ud-annotation-model:Case, ud-
        annotation-model:Definite, ud-annotation-model:Degree,
        ud-annotation-model:Evident, ud-annotation-model:
        Foreign, ud-annotation-model:Gender, ud-annotation-
        model:Mood, ud-annotation-model:NumType, ud-annotation-
        model:Number, ud-annotation-model:Person, ud-annotation
        -model:Polarity, ud-annotation-model:Polite, ud-
        annotation-model:Poss, ud-annotation-model:PronType, ud
        -annotation-model:Reflex, ud-annotation-model:Tense, ud
        -annotation-model:VerbForm, ud-annotation-model:Voice

DisjointClasses:
    ud-annotation-model:VerbFormConv, ud-annotation-model:
        VerbFormFin, ud-annotation-model:VerbFormGdv, ud-
        annotation-model:VerbFormGer, ud-annotation-model:
        VerbFormInf, ud-annotation-model:VerbFormPart, ud-
        annotation-model:VerbFormSup, ud-annotation-model:
```

```
        VerbFormVnoun

DisjointClasses:
    ud-annotation-model:AspectHab,ud-annotation-model:
        AspectImp,ud-annotation-model:AspectIter,ud-
        annotation-model:AspectPerf,ud-annotation-model:
        AspectProg,ud-annotation-model:AspectProsp

DisjointClasses:
    ud-annotation-model:TenseFut,ud-annotation-model:
        TenseImp,ud-annotation-model:TensePast,ud-annotation-
        model:TensePqp,ud-annotation-model:TensePres

DisjointClasses:
    ud-annotation-model:DegreeAbs,ud-annotation-model:
        DegreeCmp,ud-annotation-model:DegreeEqu,ud-annotation
        -model:DegreePos,ud-annotation-model:DegreeSup

DisjointClasses:
    ud-annotation-model:PronTypeArt,ud-annotation-model:
        PronTypeDem,ud-annotation-model:PronTypeEmp,ud-
        annotation-model:PronTypeExc,ud-annotation-model:
        PronTypeInd,ud-annotation-model:PronTypeInt,ud-
        annotation-model:PronTypeNeg,ud-annotation-model:
        PronTypePrs,ud-annotation-model:PronTypeRcp,ud-
        annotation-model:PronTypeRel,ud-annotation-model:
        PronTypeTot

DisjointClasses:
    ud-annotation-model:PoliteElev,ud-annotation-model:
        PoliteForm,ud-annotation-model:PoliteHumb,ud-
        annotation-model:PoliteInfm

DisjointClasses:
    ud-annotation-model:NumTypeCard,ud-annotation-model:
        NumTypeDist,ud-annotation-model:NumTypeFrac,ud-
        annotation-model:NumTypeMult,ud-annotation-model:
        NumTypeOrd,ud-annotation-model:NumTypeRange,ud-
        annotation-model:NumTypeSets
```

```
DisjointClasses:
    ud-annotation-model:PUNCT_CLASS,ud-annotation-model:
        SYM_CLASS,ud-annotation-model:X_CLASS

DisjointClasses:
    ud-annotation-model:GenderCom,ud-annotation-model:
        GenderFem,ud-annotation-model:GenderMasc,ud-
        annotation-model:GenderNeut

DisjointClasses:
    ud-annotation-model:acl,ud-annotation-model:advcl,ud-
        annotation-model:advmod,ud-annotation-model:amod,ud-
        annotation-model:appos,ud-annotation-model:aux,ud-
        annotation-model:case,ud-annotation-model:cc,ud-
        annotation-model:ccomp,ud-annotation-model:clf,ud-
        annotation-model:compound,ud-annotation-model:conj,ud
        -annotation-model:cop,ud-annotation-model:csubj,ud-
        annotation-model:dep,ud-annotation-model:det,ud-
        annotation-model:discourse,ud-annotation-model:
        dislocated,ud-annotation-model:expl,ud-annotation-
        model:fixed,ud-annotation-model:flat,ud-annotation-
        model:goeswith,ud-annotation-model:iobj,ud-annotation
        -model:list,ud-annotation-model:mark,ud-annotation-
        model:nmod,ud-annotation-model:nsubj,ud-annotation-
        model:nummod,ud-annotation-model:obj,ud-annotation-
        model:obl,ud-annotation-model:orphan,ud-annotation-
        model:parataxis,ud-annotation-model:punct,ud-
        annotation-model:reparandum,ud-annotation-model:root,
        ud-annotation-model:vocative,ud-annotation-model:
        xcomp

DisjointClasses:
    ud-annotation-model:CaseAbs,ud-annotation-model:CaseAcc,
        ud-annotation-model:CaseErg,ud-annotation-model:
        CaseNom

DisjointClasses:
```

ud-annotation-model:ADP_CLASS, ud-annotation-model:
AUX_CLASS, ud-annotation-model:CCONJ_CLASS, ud-
annotation-model:DET_CLASS, ud-annotation-model:
NUM_CLASS, ud-annotation-model:PART_CLASS, ud-
annotation-model:PRON_CLASS, ud-annotation-model:
SCONJ_CLASS

DisjointClasses:
ud-annotation-model:AnimacyAnim, ud-annotation-model:
AnimacyHum, ud-annotation-model:AnimacyInan, ud-
annotation-model:AnimacyNhum

DisjointClasses:
ud-annotation-model:DefiniteCom, ud-annotation-model:
DefiniteCons, ud-annotation-model:DefiniteDef, ud-
annotation-model:DefiniteInd, ud-annotation-model:
DefiniteSpec

DisjointClasses:
ud-annotation-model:MoodAdm, ud-annotation-model:MoodCnd,
ud-annotation-model:MoodDes, ud-annotation-model:
MoodImp, ud-annotation-model:MoodInd, ud-annotation-
model:MoodJus, ud-annotation-model:MoodNec, ud-
annotation-model:MoodOpt, ud-annotation-model:MoodPot,
ud-annotation-model:MoodPrp, ud-annotation-model:
MoodQot, ud-annotation-model:MoodSub

DisjointClasses:
ud-annotation-model:VoiceAct, ud-annotation-model:
VoiceAntip, ud-annotation-model:VoiceCau, ud-annotation
-model:VoiceDir, ud-annotation-model:VoiceInv, ud-
annotation-model:VoiceMid, ud-annotation-model:
VoicePass, ud-annotation-model:VoiceRcp

DisjointClasses:
ud-annotation-model:ADJ_CLASS, ud-annotation-model:
ADV_CLASS, ud-annotation-model:INTJ_CLASS, ud-
annotation-model:NOUN_CLASS, ud-annotation-model:
PROPN_CLASS, ud-annotation-model:VERB_CLASS

```
DisjointClasses:
    ud-annotation-model:NumberColl,ud-annotation-model:
        NumberCount,ud-annotation-model:NumberDual,ud-
        annotation-model:NumberGrpa,ud-annotation-model:
        NumberGrpl,ud-annotation-model:NumberInv,ud-
        annotation-model:NumberPauc,ud-annotation-model:
        NumberPlur,ud-annotation-model:NumberPtan,ud-
        annotation-model:NumberSing,ud-annotation-model:
        NumberTri

DisjointClasses:
    ud-annotation-model:ClosedClass,ud-annotation-model:
        OpenClass,ud-annotation-model:
        OtherUniversalPartOfSpeech

DisjointClasses:
    ud-annotation-model:Person0,ud-annotation-model:Person1,
        ud-annotation-model:Person2,ud-annotation-model:
        Person3,ud-annotation-model:Person4
```

Bibliography

2013, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*. The Association for Computer Linguistics. ISBN: 978-1-937284-51-0. Available at: <http://aclweb.org/anthology/P/P13/>. Cited on page 159.

2014, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. The Association for Computer Linguistics. ISBN: 978-1-937284-72-5. Available at: <http://aclweb.org/anthology/P/P14/>. Cited on page 164.

2003, *EACL 2003, 10th Conference of the European Chapter of the Association for Computational Linguistics, April 12-17, 2003, Agro Hotel, Budapest, Hungary*. The Association for Computer Linguistics. Available at: <http://aclweb.org/anthology/E/E03/>. Cited on page 154.

2000, *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece*. European Language Resources Association. Available at: <http://www.lrec-conf.org/proceedings/lrec2000/>. Cited on page 153.

2008, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*. European Language Resources Association. Available at: <http://www.lrec-conf.org/proceedings/lrec2008/>. Cited on page 164.

2010, *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*. The Association for Computational Linguistics. ISBN: 978-1-932432-65-7. Cited on page 161.

2005, *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. AUAI Press. ISBN: 0-9749039-1-4. Cited on page 164.

2016, *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*. The Association for Computer Linguistics. Available at: <http://aclweb.org/anthology/W/W16/>. Cited on page 151.

ABEILLÉ, A. (Ed.), 2003, *Treebanks: Building and Using Parsed Corpora*. Text, Speech and Language Technology. Springer Netherlands. ISBN: 9789401002011. Cited on pages 37 and 163.

ABEND, O., RAPPOPORT, A., 2017, "The State of the Art in Semantic Representation". In: BARZILAY and KAN (2017), pp. 77–89. ISBN: 978-1-945626-75-3. Cited on pages 4 and 5.

ALANI, H., KAGAL, L., FOKOUE, A., et al. (Eds.), 2013, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, v. 8219, *Lecture Notes in Computer Science*. Springer. ISBN: 978-3-642-41337-7. Available at: <https://doi.org/10.1007/978-3-642-41338-4>. Cited on page 156.

ALLEN, J. F., FIKES, R., SANDEWALL, E. (Eds.), 1991, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991*. Morgan Kaufmann. ISBN: 1-55860-165-1. Cited on page 155.

ARTZI, Y., ZETTLEMOYER, L., 2013, "Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions", *TACL*, v. 1, pp. 49–62. Cited on page 5.

BARZILAY, R., KAN, M. (Eds.), 2017, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics. ISBN: 978-1-945626-75-3. Available at: <http://aclanthology.info/volumes/proceedings-of-the-55th-annual-meeting-of-the-association-for-computationa
Cited on page 150.

BELTAGY, I., ERK, K., MOONEY, R., 2014, "Semantic Parsing using Distributional Semantics and Probabilistic Logic", *Proceedings of the ACL 2014 Workshop on Semantic Parsing*. Cited on page 4.

BENDER, E. M., FLICKINGER, D., OEPEN, S., et al., 2015, "Layers of Interpretation: On Grammar and Compositionality". In: PURVER *et al.* (2015), pp. 239–249. ISBN: 978-1-941643-33-4. Cited on pages 4 and 5.

BOJAR, O., CHATTERJEE, R., FEDERMANN, C., et al., 2016, "Findings of the 2016 Conference on Machine Translation". In: *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany* DBL (2016), pp. 131–198. Cited on page 5.

BOYD, A., DICKINSON, M., MEURERS, W. D., 2008, "On Detecting Errors in Dependency Treebanks", *Research on Language and Computation*, v. 6, n. 2 (Oct), pp. 113–137. ISSN: 1572-8706. Cited on page 41.

BRACHMAN, R. J., LEVESQUE, H. J., 2004, *Knowledge Representation and Reasoning*. Elsevier. ISBN: 978-1-55860-932-7. Cited on page 15.

CALZOLARI, N., CHOUKRI, K., MAEGAARD, B., et al. (Eds.), 2010, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association. ISBN: 2-9517408-6-7. Available at: <`http://www.lrec-conf.org/proceedings/lrec2010/index.html`>. Cited on page 163.

CALZOLARI, N., CHOUKRI, K., DECLERCK, T., et al. (Eds.), 2012, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*. European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7. Cited on page 160.

CALZOLARI, N., CHOUKRI, K., DECLERCK, T., et al. (Eds.), 2014, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*. European Language Resources Association (ELRA). Available at: <`http://www.lrec-conf.org/lrec2014`>. Cited on page 153.

CALZOLARI, N., CHOUKRI, K., DECLERCK, T., et al. (Eds.), 2016, *Proceedings of the Tenth International Conference on Language Resources*

and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016. European Language Resources Association (ELRA). Available at: <http://www.lrec-conf.org/lrec2016>. Cited on pages 160, 162, and 163.

CAROLL, J., HERMAN, I., PATEL-SCHNEIDER, P. F. (Eds.), 2012, *OWL 2 Web Ontology Language RDF-Based Semantics*. W3C Recommendation. Available at https://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/. Cited on page 17.

CARTER, D., 1997, "The TreeBanker. A tool for supervised training of parsed corpora". In: *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pp. pp. 9–15, Madrid, Spain. Cited on page 48.

CHIARCOS, C., 2010, "Towards Robust Multi-Tool Tagging. An OWL/DL-Based Approach". In: HAJIC *et al.* (2010), pp. 659–670. ISBN: 978-1-932432-66-4. Cited on page 43.

CHIARCOS, C., FÄTH, C., 2017, "CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way". In: GRACIA *et al.* (2017), pp. 74–88. ISBN: 978-3-319-59887-1. Cited on page 24.

CHIARCOS, C., SUKHAREVA, M., 2015, "OLiA - Ontologies of Linguistic Annotation", *Semantic Web*, v. 6, n. 4, pp. 379–386. Cited on pages 19 and 42.

CHIARCOS, C., MCCRAE, J., CIMIANO, P., et al., 2013, "Towards Open Data for Linguistics: Linguistic Linked Data". In: OLTRAMARI, A., VOSSEN, P., QIN, L., et al. (Eds.), *New Trends of Research in Ontologies and Lexical Resources: Ideas, Projects, Systems*, Springer Berlin Heidelberg, pp. 7–25, Berlin, Heidelberg. ISBN: 978-3-642-31782-8. Cited on page 42.

CHIARCOS, C., FÄTH, C., SUKHAREVA, M., 2016, "Developing and using the ontologies of linguistic annotation (2006-2016)". In: *LDL 2016 5th Workshop on Linked Data in Linguistics: Managing, Building and Using Linked Language Resources*, p. 63. Cited on pages 42 and 43.

CLARK, A., FOX, C., LAPPIN, S. (Eds.), 2010, *The Handbook Of Computational Linguistics And Natural Language Processing*. Chichester, United Kingdom, Wiley-Blackwell. Cited on pages 152, 160, and 161.

CLARK, S., 2010, "Statistical Parsing". In: CLARK *et al.* (2010), p. 333–363. Cited on page 5.

COPESTAKE, A. A., FLICKINGER, D., 2000, "An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG". In: *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece* DBL (2000). Cited on page 49.

CORRO, L. D., GEMULLA, R., 2013, "ClausIE: clause-based open information extraction". In: SCHWABE *et al.* (2013), pp. 355–366. ISBN: 978-1-4503-2035-1. Cited on page 4.

CROFT, W., 2002, *Typology and Universals*. Cambridge University Press. ISBN: 9780511840579. Cited on page 2.

CUNNINGHAM, H., MAYNARD, D., BONTCHEVA, K., et al., 2002, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications". In: *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Cited on page 43.

DE MARNEFFE, M., DOZAT, T., SILVEIRA, N., et al., 2014, "Universal Stanford dependencies: A cross-linguistic typology". In: CALZOLARI *et al.* (2014), pp. 4585–4592. Cited on page 7.

DE MARNEFFE, M., GRIONI, M., KANERVA, J., et al., 2017a, "Assessing the Annotation Consistency of the Universal Dependencies Corpora". In: MONTEMAGNI and NIVRE (2017), pp. 108–115. ISBN: 978-91-7685-467-9. Cited on page 41.

DE MARNEFFE, M., NIVRE, J., SCHUSTER, S. (Eds.), 2017b, *Proceedings of the NoDaLiDa Workshop on Universal Dependencies, UDW@NoDaLiDa 2017, Gothenburg, Sweden, May 22, 2017*, b. Association for Computational Linguistics. ISBN: 978-91-7685-501-0. Available at: <https://aclanthology.info/volumes/proceedings-of-the-nodalida-2017-workshop-on-universal-dependencies-udw-20 Cited on pages 161 and 164.

DE SMEDT, K., ROSÉN, V., MEURER, P., 2015, "Studying Consistency in UD Treebanks with INESS-Search". In: DICKINSON *et al.* (2015). Cited on page 40.

DICKINSON, M., 2015, "Detection of Annotation Errors in Corpora", *Language and Linguistics Compass*, v. 9, n. 3, pp. 119–138. Cited on page 40.

DICKINSON, M., MEURERS, D., 2003a, "Detecting Errors in Part-of-Speech Annotation". In: *EACL 2003, 10th Conference of the European Chapter of the Association for Computational Linguistics, April 12-17, 2003, Agro Hotel, Budapest, Hungary* DBL (2003), pp. 107–114. Cited on page 41.

DICKINSON, M., MEURERS, W. D., 2003b, "Detecting Inconsistencies in Treebanks". In: *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, pp. 45–56, Växjö, Sweden, b. Cited on page 41.

DICKINSON, M., MEURERS, W. D., 2005, "Prune diseased branches to get healthy trees! How to find erroneous local trees in a treebank and why it matters". In: *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pp. pp. 41–52, Barcelona, Spain. Cited on page 41.

DICKINSON, M., HINRICHS, E., PATEJUK, A., et al. (Eds.), 2015, *Proceedings of the Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT14)*, Warsaw, Poland. Cited on page 153.

DOHERTY, P., MYLOPOULOS, J., WELTY, C. A. (Eds.), 2006, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. AAAI Press. ISBN: 978-1-57735-271-6. Cited on page 156.

FERRUCCI, D., LALLY, A., 2004, "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment", *Natural Language Engineering*, v. 10, n. 3-4 (Sept.), pp. 327–348. ISSN: 1351-3249. Cited on page 43.

FLICKINGER, D., OEPEN, S., BENDER, E. M., 2017, "Sustainable Development and Refinement of Complex Linguistic Annotations at Scale". In: IDE and PUSTEJOVSKY (2017), p. 353–377. ISBN: 9789402408812. Cited on page 48.

GETOOR, L., TASKAR, B. (Eds.), 2007, *Introduction to Statistical Relational Learning*. Adaptive Computation and Machine Learning series. Cambridge, Massachusetts, MIT Press. Cited on page 16.

GLIMM, B., HORROCKS, I., MOTIK, B., et al., 2014, "HermiT: An OWL 2 Reasoner", *Journal of Automated Reasoning*, v. 53, n. 3 (Oct), pp. 245–269. ISSN: 1573-0670. Cited on page 24.

GRACIA, J., BOND, F., MCCRAE, J. P., et al. (Eds.), 2017, *Language, Data, and Knowledge - First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings*, v. 10318, *Lecture Notes in Computer Science*. Springer. ISBN: 978-3-319-59887-1. Available at: <https://doi.org/10.1007/978-3-319-59888-8>. Cited on page 152.

GRISHMAN, R., 2010, "Information Extraction", *The Handbook of Computational Linguistics and Natural Language Processing*, (Jun), pp. 515–530. Cited on page 4.

GUILLAUME, B., BONFANTE, G., MASSON, P., et al., 2012, "Grew : un outil de réécriture de graphes pour le TAL (Grew: a Graph Rewriting Tool for NLP) [in French]". In: *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 5: Software Demonstrations*, pp. 1–2. ATALA/AFCP. Cited on pages xi, 38, and 39.

HAJIC, J., ZEMAN, D. (Eds.), 2017, *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*. Association for Computational Linguistics. ISBN: 978-1-945626-70-8. Available at: <http://aclanthology.info/volumes/proceedings-of-the-conll-2017-shared-task-multilingual-parsing-from-raw-te
Cited on page 164.

HAJIC, J., CARBERRY, S., CLARK, S. (Eds.), 2010, *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*. The Association for Computer Linguistics. ISBN: 978-1-932432-66-4. Cited on page 152.

HAJICOVÁ, E., NIVRE, J. (Eds.), 2015, *Proceedings of the Third International Conference on Dependency Linguistics, DepLing 2015, August 24-26 2015, Uppsala University, Uppsala, Sweden*. Uppsala University, Department of Linguistics and Philology. ISBN: 978-91-637-8965-6. Available at: <http://aclweb.org/anthology/W/W15/>. Cited on page 162.

HALPERN, J. Y., VARDI, M. Y., 1991, "Model Checking vs. Theorem Proving: A Manifesto". In: ALLEN *et al.* (1991), pp. 325–334. ISBN: 1-55860-165-1. Cited on page 53.

HELLMANN, S., 2015, *Integrating Natural Language Processing (NLP) and Language Resources Using Linked Data*. Tese de Doutorado, Universität Leipzig, Göttingen, Germany, jan. Cited on page 42.

HELLMANN, S., LEHMANN, J., AUER, S., et al., 2013, "Integrating NLP Using Linked Data". In: ALANI *et al.* (2013), pp. 98–113. ISBN: 978-3-642-41337-7. Cited on pages 42 and 43.

HITZLER, P., KRÖTZSCH, M., PARSIA, B., et al. (Eds.), 2009, *OWL 2 Web Ontology Language: Primer*. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-primer/`. Cited on page 18.

HITZLER, P., KRÖTZSCH, M., RUDOLPH, S., 2010, *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press. ISBN: 9781420090505. Cited on page 15.

HORRIDGE, M., 2011, *Justification based explanation in ontologies*. Tese de Doutorado, University of Manchester, UK. Available at: <`http://www.manchester.ac.uk/escholar/uk-ac-man-scw:131699`>. Cited on pages 25 and 52.

HORRIDGE, M., BECHHOFER, S., 2011, "The OWL API: A Java API for OWL ontologies", *Semantic Web*, v. 2, n. 1, pp. 11–21. Cited on page 24.

HORRIDGE, M., PARSIA, B., SATTLER, U., 2008, "Laconic and Precise Justifications in OWL". In: SHETH *et al.* (2008), pp. 323–338. ISBN: 978-3-540-88563-4. Cited on pages 24, 25, 28, and 53.

HORROCKS, I., KUTZ, O., SATTLER, U., 2006, "The Even More Irresistible SROIQ". In: DOHERTY *et al.* (2006), pp. 57–67. ISBN: 978-1-57735-271-6. Cited on pages 18 and 22.

HUNTER, A., KONIECZNY, S., 2008, "Measuring Inconsistency through Minimal Inconsistent Sets". In: *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pp. 358–366, Sydney, Australia. Cited on page 53.

IDE, N., PUSTEJOVSKY, J. (Eds.), 2017, *Handbook of Linguistic Annotation*. Dordrecht, Netherlands, Springer Science+Business Media. ISBN: 9789402408812. Cited on pages 37, 154, and 162.

IDE, N., PUSTEJOVSKY, J., 2010, "What Does Interoperability Mean, anyway? Toward an Operational Definition of Interoperability". In: *Proceedings of the Second International Conference on Global Interoperability for Language Resources (ICGL 2010)*, Hong Kong, China. Cited on page 42.

IDE, N., XIA, F. (Eds.), 2012, *Proceedings of the Sixth Linguistic Annotation Workshop, LAW@ACL 2012, July 12-13, 2012, Jeju Island, Republic of*

Korea. The Association for Computer Linguistics. ISBN: 978-1-937284-32-9. Cited on page 161.

IVANOVA, A., 2015, *Bilexical Dependencies as an Intermedium for Data-Driven and HPSG-Based Parsing*. Tese de Doutorado, Faculty of Mathematics and Natural Sciences, University of Oslo, nov. Cited on pages 6 and 7.

JEŽEK, E., 2016, *The Lexicon: An Introduction*. Oxford University Press. Cited on page 8.

JIANG, M., HUANG, Y., FAN, J.-W., et al., 2015, "Parsing clinical text: how good are the state-of-the-art parsers?" *BMC Medical Informatics and Decision Making*, v. 15, n. S1 (May). ISSN: 1472-6947. Cited on page 37.

JOHNSON, M., SCHUSTER, M., LE, Q., et al., 2017, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation", *Transactions of the Association for Computational Linguistics*, v. 5, pp. 339–351. ISSN: 2307-387X. Cited on page 5.

KALOULI, A.-L., CROUCH, R., 2018, "GKR: the Graphical Knowledge Representation for semantic parsing". In: *Proceedings of the Workshop on Computational Semantics beyond Events and Roles*, pp. 27–37. Association for Computational Linguistics. Cited on page 14.

KANAYAMA, H., TAKEDA, K., 2017, "Multilingualization of Question Answering Using Universal Dependencies". In: *Proceedings of Open Knowledge Base and Question Answering Workshop at SIGIR2017*, Tokyo, Japan, aug. Cited on page 13.

KAY, M., BOITET, C. (Eds.), 2012, *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. Indian Institute of Technology Bombay. Available at: <`http://aclweb.org/anthology/C/C12/`>. Cited on page 162.

KERY, M. B., RADENSKY, M., ARYA, M., et al., 2018, "The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool", *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. Cited on page 44.

KLUYVER, T., RAGAN-KELLEY, B., PÉREZ, F., et al., 2016, "Jupyter Notebooks - a publishing format for reproducible computational workflows". In: LOIZIDES and SCHMIDT (2016), pp. 87–90. ISBN: 978-1-61499-648-4. Cited on page 44.

KONTOKOSTAS, D., BRÜMMER, M., HELLMANN, S., et al., 2014, "NLP Data Cleansing Based on Linguistic Ontology Constraints", *The Semantic Web: Trends and Challenges*, p. 224–239. ISSN: 1611-3349. Cited on page 42.

KÜBLER, S., MCDONALD, R. T., NIVRE, J., 2009, *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers. Cited on page 6.

LOIZIDES, F., SCHMIDT, B. (Eds.), 2016, *Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany, June 7-9, 2016*. IOS Press. ISBN: 978-1-61499-648-4. Available at: <`http://ebooks.iospress.nl/ISBN/978-1-61499-648-4`>. Cited on page 157.

LUOTOLAHTI, J., KANERVA, J., GINTER, F., 2017, "Dep_search: Efficient Search Tool for Large Dependency Parsebanks". In: *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*, Gothenburg, Sweden. Linköping University Electronic Press. Cited on pages xi and 38.

MARCUS, M. P., SANTORINI, B., MARCINKIEWICZ, M. A., 1993, "Building a Large Annotated Corpus of English: The Penn Treebank", *Computational Linguistics*, v. 19, n. 2, pp. 313–330. Cited on pages 1 and 6.

MÀRQUEZ, L., CALLISON-BURCH, C., SU, J., et al. (Eds.), 2015, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. The Association for Computational Linguistics. ISBN: 978-1-941643-32-7. Available at: <`http://aclweb.org/anthology/D/D15/`>. Cited on page 162.

MAUSAM, SCHMITZ, M., SODERLAND, S., et al., 2012, "Open Language Learning for Information Extraction". In: TSUJII *et al.* (2012), pp. 523–534. ISBN: 978-1-937284-43-5. Cited on page 4.

MCCORD, M. C., 1990, "Slot Grammar", *Lecture Notes in Computer Science*, p. 118–145. ISSN: 1611-3349. Cited on page 49.

MCCRAE, J., MONTIEL-PONSODA, E., CIMIANO, P., 2012, "Integrating WordNet and Wiktionary with lemon". In: CHIARCOS, C., NORDHOFF, S., HELLMANN, S. (Eds.), *Linked Data in Linguistics: Representing and Connecting Language Data and Language Metadata*, Springer Berlin Heidelberg, pp. 25–34, Berlin, Heidelberg. ISBN: 978-3-642-28249-2. Cited on page 42.

MCDONALD, R. T., NIVRE, J., QUIRMBACH-BRUNDAGE, Y., et al., 2013, "Universal Dependency Annotation for Multilingual Parsing". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers* DBL (2013), pp. 92–97. ISBN: 978-1-937284-51-0. Cited on page 7.

MERKEL, D., 2014, "Docker: Lightweight Linux Containers for Consistent Development and Deployment", *Linux Journal*, v. 2014, n. 239 (Mar.). ISSN: 1075-3583. Cited on page 44.

MILLMAN, K. J., PÉREZ, F., 2014, "Developing Open-Source Scientific Practice". In: STODDEN *et al.* (2014), p. 448. Cited on page 44.

MONTEMAGNI, S., NIVRE, J. (Eds.), 2017, *Proceedings of the Fourth International Conference on Dependency Linguistics, Depling 2017, Pisa, Italy, September 18-20, 2017*. Linköping University Electronic Press. ISBN: 978-91-7685-467-9. Available at: <`https://aclanthology.info/volumes/proceedings-of-the-fourth-international-conference-on-dependency-linguisti` Cited on page 153.

MOTIK, B., PATEL-SCHNEIDER, P. F., GRAU, B. C. (Eds.), 2012a, *OWL 2 Web Ontology Language Direct Semantics*. W3C Recommendation. Available at `https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/`. Cited on page 17.

MOTIK, B., PATEL-SCHNEIDER, P. F., PARSIA, B. (Eds.), 2012b, *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. W3C Recommendation. Available at `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/`. Cited on page 22.

MUGGLETON, S., RAEDT, L. D., POOLE, D., et al., 2012, "ILP turns 20 - Biography and future challenges", *Machine Learning*, v. 86, n. 1, pp. 3–23. Cited on page 16.

MUNIZ, H., CHALUB, F., RADEMAKER, A., 2017, "CL-CONLLU: dependências universais em Common Lisp". In: *V Workshop de Iniciação Científica em Tecnologia da Informação e da Linguagem Humana (TILic)*, Uberlândia, MG, Brazil. https://sites.google.com/view/tilic2017/. Cited on pages xii, 24, 45, and 46.

NIVRE, J., 2006, *Inductive Dependency Parsing*, v. 34, *Text, speech and language technology*. Springer. ISBN: 978-1-4020-4888-3. Cited on pages 5, 6, and 49.

NIVRE, J., 2015, "Towards a Universal Grammar for Natural Language Processing", *Lecture Notes in Computer Science*, p. 3–16. ISSN: 1611-3349. Cited on page 18.

NIVRE, J., DE MARNEFFE, M., GINTER, F., et al., 2016, "Universal Dependencies v1: A Multilingual Treebank Collection". In: CALZOLARI *et al.* (2016). Cited on pages 7, 8, and 12.

NIVRE, J., AGIĆ, Ž., AHRENBERG, L., et al., 2017. "Universal Dependencies 2.1". Available at: <`http://hdl.handle.net/11234/1-2515`>. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. Cited on page 7.

NIVRE, J., ABRAMS, M., AGIĆ, Ž., et al., 2018. "Universal Dependencies 2.2". Available at: <`http://hdl.handle.net/11234/1-2837`>. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. Cited on page 7.

OLIVA, K., KVETON, P., 2002, "Linguistically Motivated Bigrams in Part-of-Speech Tagging of Language Corpora", *Prague Bull. Math. Linguistics*, v. 78, pp. 23–36. Cited on page 40.

OWL WORKING GROUP, W., 2009, *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation. Available at `http://www.w3.org/TR/owl2-overview/`. Cited on page 17.

PALMER, M., XUE, N., 2010, "Linguistic Annotation". In: CLARK *et al.* (2010), p. 238–270. Cited on page 1.

PESCHANSKI, F., 2015. "cl-Jupyter: an enhanced interactive Shell for Common Lisp". oct. Available at: <`https://github.com/fredokun/cl-jupyter/blob/master/about-cl-jupyter.pdf`>. Accessed on August 4, 2018. Cited on page 44.

PETROV, S., DAS, D., MCDONALD, R. T., 2012, "A Universal Part-of-Speech Tagset". In: CALZOLARI *et al.* (2012), pp. 2089–2096. ISBN: 978-2-9517408-7-7. Cited on page 7.

POLLERES, A., D'AMATO, C., ARENAS, M., et al. (Eds.), 2011, *Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures*, v. 6848, *Lecture Notes in Computer Science*. Springer.

ISBN: 978-3-642-23031-8. Available at: <https://doi.org/10.1007/978-3-642-23032-5>. Cited on page 161.

POPEL, M., ZABOKRTSKÝ, Z., VOJTEK, M., 2017, "Udapi: Universal API for Universal Dependencies". In: DE MARNEFFE *et al.* (2017b), pp. 96–101. ISBN: 978-91-7685-501-0. Cited on page 33.

PRZEPIÓRKOWSKI, A., LENART, M., 2012, "Simultaneous error detection at two levels of syntactic annotation". In: IDE and XIA (2012), pp. 118–123. ISBN: 978-1-937284-32-9. Cited on page 41.

PURVER, M., SADRZADEH, M., STONE, M. (Eds.), 2015, *Proceedings of the 11th International Conference on Computational Semantics, IWCS 2015, 15-17 April, 2015, Queen Mary University of London, London, UK*. The Association for Computer Linguistics. ISBN: 978-1-941643-33-4. Available at: <http://aclweb.org/anthology/W/W15/>. Cited on page 151.

RADEMAKER, A., 2010, *A Proof Theory for Description Logics*. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro. Cited on page 53.

RAMBOW, O., 2010, "The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece". In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA* DBL (2010), pp. 337–340. ISBN: 978-1-932432-65-7. Cited on page 6.

REDDY, S., TÄCKSTRÖM, O., COLLINS, M., et al., 2016, "Transforming Dependency Structures to Logical Forms for Semantic Parsing", *TACL*, v. 4, pp. 127–140. Cited on page 4.

REDDY, S., TÄCKSTRÖM, O., PETROV, S., et al., 2017, "Universal Semantic Parsing", *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Cited on pages 13 and 14.

RESNIK, P., LIN, J., 2010, "Evaluation of NLP Systems". In: CLARK *et al.* (2010), p. 271–295. Cited on page 1.

RIGUZZI, F., BELLODI, E., LAMMA, E., et al., 2013, "BUNDLE: A Reasoner for Probabilistic Ontologies". In: *RR*, v. 7994, *Lecture Notes in Computer Science*, pp. 183–197. Springer. Cited on page 24.

RUDOLPH, S., 2011, "Foundations of Description Logics". In: POLLERES *et al.* (2011), pp. 76–136. ISBN: 978-3-642-23031-8. Cited on page 18.

RUMSHISKY, A., STUBBS, A., 2017, "Machine Learning for Higher-Level Linguistic Tasks". In: IDE and PUSTEJOVSKY (2017), p. 333–351. ISBN: 9789402408812. Cited on page 47.

SAG, I. A., WASOW, T., BENDER, E. M., 2003, *Syntactic Theory: A Formal Introduction*. 2 ed. Stanford, CA, CSLI. Cited on pages 2 and 4.

SCHUSTER, S., MANNING, C. D., 2016, "Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks". In: CALZOLARI *et al.* (2016). Cited on page 12.

SCHWABE, D., ALMEIDA, V. A. F., GLASER, H., et al. (Eds.), 2013, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. International World Wide Web Conferences Steering Committee / ACM. ISBN: 978-1-4503-2035-1. Available at: <http://dl.acm.org/citation.cfm?id=2488388>. Cited on page 153.

SCHWARTZ, R., ABEND, O., RAPPOPORT, A., 2012, "Learnability-Based Syntactic Annotation Design". In: KAY and BOITET (2012), pp. 2405–2422. Cited on page 47.

SENNRICH, R., HADDOW, B., 2015, "A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation". In: MÀRQUEZ *et al.* (2015), pp. 2081–2087. ISBN: 978-1-941643-32-7. Cited on page 5.

SHETH, A. P., STAAB, S., DEAN, M., et al. (Eds.), 2008, *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, v. 5318, *Lecture Notes in Computer Science*. Springer. ISBN: 978-3-540-88563-4. Available at: <https://doi.org/10.1007/978-3-540-88564-1>. Cited on page 156.

SILVEIRA, N., MANNING, C. D., 2015, "Does Universal Dependencies need a parsing representation? An investigation of English". In: HAJICOVÁ and NIVRE (2015), pp. 310–319. ISBN: 978-91-637-8965-6. Cited on page 47.

SILVEIRA, N., DOZAT, T., DE MARNEFFE, M.-C., et al., 2014, "A Gold Standard Dependency Corpus for English". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. Cited on page 49.

STEEDMAN, M., 2000, *The Syntactic Process*. MIT Press. Cited on page 2.

STEPÁNEK, J., PAJAS, P., 2010, "Querying Diverse Treebanks in a Uniform Way". In: CALZOLARI *et al.* (2010). ISBN: 2-9517408-6-7. Cited on page 38.

STODDEN, V. C., LEISCH, F., PENG, R. D., 2014, *Implementing Reproducible Research*. CRC Press. Cited on page 159.

STRAKA, M., STRAKOVÁ, J., 2017, "Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe". In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 88–99, Vancouver, Canada, August. Association for Computational Linguistics. Cited on page 29.

SUKHAREVA, M., CHIARCOS, C., 2016, "Combining Ontologies and Neural Networks for Analyzing Historical Language Varieties. A Case Study in Middle Low German". In: CALZOLARI *et al.* (2016). Cited on page 43.

SZABÓ, Z. G., 2017, "Compositionality". In: ZALTA, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy*, summer 2017 ed., Metaphysics Research Lab, Stanford University. Cited on page 4.

TSUJII, J., HENDERSON, J., PASCA, M. (Eds.), 2012, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL*, Jeju Island, Korea, jul. ACL. ISBN: 978-1-937284-43-5. Available at: <`http://www.aclweb.org/anthology/K/K12/#2012_0`>. Cited on page 158.

VAN VALIN, JR, R. D., 2004, *An Introduction to Syntax*. Cambridge University Press. Cited on page 2.

VAN VALIN, JR, R. D., LAPOLLA, R. J., 1997, *Syntax: Structure, Meaning and Function*. Cambridge University Press. Cited on page 2.

WALLIS, S., 2003, "Completing Parsed Corpora". In: ABEILLÉ (2003), p. 61–71. ISBN: 9789401002011. Cited on pages 37 and 38.

WAY, A., 2010, "Machine Translation", *The Handbook of Computational Linguistics and Natural Language Processing*, (Jun), pp. 531–573. Cited on page 4.

WEBBER, B., WEBB, N., 2010, "Question Answering", *The Handbook of Computational Linguistics and Natural Language Processing*, (Jun), pp. 630–654. Cited on page 4.

WEITZ, E., 2009, *Hunchentoot - The Common Lisp web server formerly known as TBNL*. Available at: <https://edicl.github.io/hunchentoot/>. Accessed on August 4, 2018. Cited on page 45.

WHITE, A. S., REISINGER, D., SAKAGUCHI, K., et al., 2016, "Universal Decompositional Semantics on Universal Dependencies", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Cited on pages 4 and 14.

WISNIEWSKI, G., LACROIX, O., 2017, "A Systematic Comparison of Syntactic Representations of Dependency Parsing". In: DE MARNEFFE *et al.* (2017b), pp. 146–152. ISBN: 978-91-7685-501-0. Cited on page 47.

WOLFRAM RESEARCH, INC. "Mathematica, Version 11.3". Champaign, IL, 2018. Cited on page 44.

WU, Y., SCHUSTER, M., CHEN, Z., et al., 2016, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation", *Computing Research Repository/arXiv*, v. abs/1609.08144. Cited on page 5.

YAO, X., DURME, B. V., 2014, "Information Extraction over Structured Data: Question Answering with Freebase". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers* DBL (2014), pp. 956–966. ISBN: 978-1-937284-72-5. Cited on page 4.

ZEMAN, D., 2008, "Reusable Tagset Conversion Using Tagset Drivers". In: *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco* DBL (2008). Cited on page 7.

ZEMAN, D., POPEL, M., STRAKA, M., et al., 2017, "CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies". In: HAJIC and ZEMAN (2017), pp. 1–19. ISBN: 978-1-945626-70-8. Cited on page 7.

ZETTLEMOYER, L. S., COLLINS, M., 2005, "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars". In: *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005* DBL (2005), pp. 658–666. ISBN: 0-9749039-1-4. Cited on page 5.

ZHANG, Y., TIRYAKI, F., JIANG, M., et al., 2018, "Parsing Clinical Text: How Good are the state-of-the-art Deep Learning Based Parsers?" *2018 IEEE International Conference on Healthcare Informatics Workshop (ICHI-W)*, (Jun). Cited on page 37.