Acknowledgements

I am thankful for the financial aid of the foundation of the Department of Theoretical Linguistics, Nyelvészet az Oktatásban és a Kutatásban Alapítvány, which made it possible for me to participate at MSZNY2018 and to present part of this work there.
Also I would like to say thank you to my advisor Gábor Recski, who introduced me to natural language processing, taught me the basics of coding and patiently answered all my questions in the past two years.

Contents

Chapter 1

Introduction

This thesis reimplements and extends the dep_to_4lang functionality using an interpreted regular tree grammar (IRTG), which maps rules of a regular tree grammar (RTG) to pairs of operations over Universal Dependencies (UD) and 4lang graphs, thereby allowing for efficient transformation between the two representations. Our contribution is available on GitHub under an MTI license.[1]

The thesis is structured as follows: Chapter 2 describes the history of dependency parsing and its recent applications, as well as introducing the UD formalism, followed by a manual error analysis of three state-of-art dependency parsers of 2017. In Chapter 3 we give a theoretical background of semantic parsing and a short review of some early and more recent systems as well. Chapter 4 explains interpreted regular tree grammars and the s-graph formalism. Then we present an IRTG which achieves the aforementioned reimplementation and extension of dep_to_4lang. Our system contains several enhancements, such as UD-conformity and the treatment of the UD relation case.

[1] `https://github.com/kornai/4lang/tree/master/exp/alto`

Chapter 2

Dependency parsing

This chapter gives a brief review on dependency parsing. We do not intend to provide a full historical overview or description of any grammatical theories in full detail. We provide a short description of dependency grammars, followed by a review of approaching the task of dependency parsing in Section 2.1. In Section 2.2 we briefly describe the Universal Dependencies project, then in Section 2.3, we provide a manual error analysis of the outputs of three recent dependency parsers which received top scores at the CoNLL 2017 shared task.

We give an introduction on the roots and main features of depenedency grammar, for a detailed overview of the field, see Nivre (2005) Although constituent-based grammars have more prevalence in traditional schools of grammar, the dependency-based approach, which state that syntactic structure consists of lexical elements linked by binary relations, also has a long tradition. Rooted in Panini's grammar of Sanskrit, the first modern work on dependency grammars is the one of Tesnière's (1959).

The notion of dependency means that words (head and dependent) are connected to each other via directed links. The finite verb counts as the root, or the structural center of the sentence, while the relations between heads and dependents determine the structure, as in the example of Figure 2.1. There are many well-known theories of dependency grammars, for summary see (Nivre, 2005, p. 3).

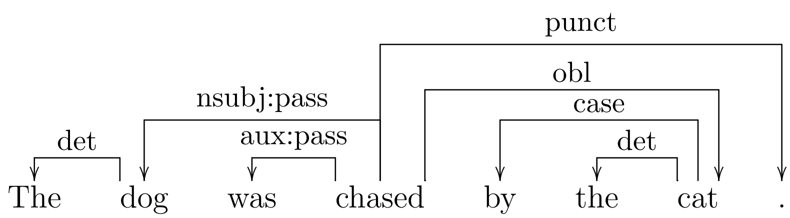


Figure 2.1: Dependency analysis of the sentence 'The dog was chased by the cat.' Source: http://universaldependencies.org/introduction.html

2.1 Approaches to dependency parsing

In the early days of dependency parsing, most of the efforts were focused on the grammar-driven approach. These can be split into two main categories.

The first one is based on the formalization of dependency grammar, Gaifman (1965) is one of the earliest works. It is a formal description of dependency grammar. Gaifman's dependency system contains a finite number of rules for dependency analysis. His three types of rules include 1. rules that list the possible dependents of category X and their relative order (Figure 2.2), 2. rules that give the list of all words belonging to the grammatical categories X, and 3. a rule that gives the list of all categories the occurrence of which may govern a sentence. Beside rules, there are general structure requirements (Gaifman, 1965, p. 306). His relations are dependent-to-head relations, opposed to today's head-to-dependent ones.

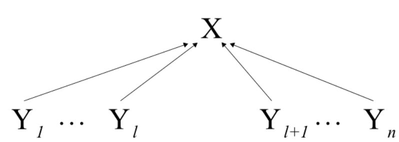


Figure 2.2: A dependency rule. Reproduced from (Gaifman, 1965, p. 306)

The second type of approach sees parsing as constraint-satisfaction, where sentence representations are filtered successively by dropping those that violate constraints until only valid representations remain. Maruyama (1990) is one of the earliest works. He argues that firstly, one needs to construct an initial constraint network using a core grammar, then to remove all local inconsistencies, finally to add new constraints and go back to the previous step if any ambiguity remained. He illustrates this with a PP-attachment example. The sentence *Put the block on the floor on the table in the room* contains many structural ambiguities. For the sake of simplicity, the author treats the symbols V, NP and PP as terminals. The core grammar is constructed to contain the terminals, the dependency labels and the constraints. Constraints define which terminals can modify which terminals and under what circumstances. They also define what label should be assigned to the resulting dependency relations. An example constraint is that if a PP modifies a PP or an NP, its label should be $POSTMOD$. These constraints narrow down the number of the possible parse trees. According to the grammar, the example sentence has 14 different syntactic structures. These are not generated one by one, instead a complex data structure, a constraint network is built. Based on this structure, parse trees can be generated. Figure 2.3 shows a possible dependency structure.

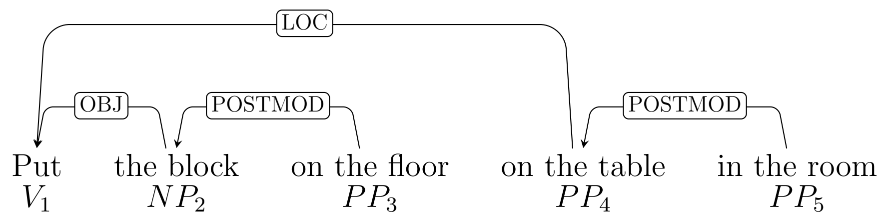


Figure 2.3: A possible dependency structure of the sentence 'Put the block on the floor on the table in the room' (Maruyama, 1990, p. 34). Arrows are drawn from the dependent to the head to emphasize that the information is contained in the role of the modifier.

Today the majority of dependency parsers employ data-driven methods, which involve probabilistic models and evaluation with supervised learning. For an overview of the field, see Jurafsky and Martin (2018b). Eisner (1996) is the most important among the first tries, he provides three different probabilistic models with different weighting schemes which all have part-of-speech tags in addition to word tokens and dependency relations. This algorithm serves as a basis for many modern parsers. McDonald et al. (2005), whose system is highly influenced by Eisner's, applied discriminative estimation methods to probabilistic dependency parsing (Nivre (2005)). Their system achieved a competitive parsing accuracy on both English and Czech data. The two main types of data-driven parsing are shift-reduce parsing and graph-based parsing. Shift-reduce parsing uses a context-free grammar, a stack and a list of tokens. It accepts words one by one, starting at the beginning of a sentence, and tries linking each word as a head or dependent of the previous word, based on a simpler notion of dependency grammar, together with a deterministic parsing strategy. Dependency links form a tree with a unique root and the parser should make a single left-to-right pass through the input string while establishing each link as early in its left-right pass as possible. A function named *oracle* decides which step should be chosen at any time. The basic algorithm of a generic transition-based parser is shown in Figure 2.4.

function DEPENDENCYPARSE(*words*) **returns** dependency tree
state ← [root], [*words*], [ ] ; initial configuration
while *state* **not final**
t ← ORACLE(*state*) ; choose a transition operator to apply
state ← APPLY(*t, state*) ; apply it, creating a new state
return *state*

Figure 2.4: A generic transition-based dependency parser (Jurafsky and Martin, 2018b, p. 9)

The oracle returns a transition operator in each step, based on the current configuration. Then it applies that operator to the current configuration, resulting in a new configuration. After all the words have been consumed and only the ROOT element is left behind, the process ends. State-of-the-art systems use supervised machine learning to map configurations to transition operations, as the system of Chen and Manning (2014).

Another data-driven approach is graph-based parsing. This method searches through all possible trees for a sentence and selects a tree with the highest score. Edge-factored approaches calculate a score from the scores of edges comprising the trees. The scores of edges are derived from training data. First, a fully-connected, weighted, directed graph is generated, where nodes represent words and edges represent head-dependent relations. Another root node is added to the graph with edges leading to all the other nodes. Then for each node, excluding the root, the edge with the highest score leading to it is selected. If this yields a maximum spanning tree, then this is the desired parse tree. If the result contains cycles, another algorithm must be employed to resolve this. Integrations of graph-based and transition-based parsers had also been implemented. Nivre and McDonald (2008)'s system is based on letting one model generating features for the other to learn from, and Zhang and Clark (2008)'s system uses a transition-based decoder for a combined system. Most recent systems are based on neural networks. State-of-the art parsers will be further discussed in Section 2.3. In natural language processing, dependency parsing is widely used, as seen in its downstream applications such as sentiment analysis (Wilson et al. (2005), Wu et al. (2009)) and question answering (Cui et al. (2005)).

2.2 Universal Dependencies

The Universal Dependencies (UD) project[1] (De Marneffe et al. (2014)) is a cross-linguistically consistent annotation system and treebanks for over 60 languages (as of version 2.1, released in November 2017, Nivre et al. (2017); the next version, v2.2 will be released on 15 April 2018). It aims to provide a universal inventory of categories and annotation guidelines while allowing language-specific extensions. UD has evolved from Stanford Dependencies (De Marneffe and Manning (2008)) by merging it with Google universal tags (Petrov et al. (2011)), a revised subset of the Interset feature inventory (Zeman (2008)), and a revised version of the CoNLL-X format (Buchholz and Marsi (2006)). It has two groups of core dependencies: the clausal relations describe syntactic roles concerning the predicate, and the modifier relations categorize the ways words modify their heads (Jurafsky and Martin (2018b)). Table 2.1 presents a selected set of UD's total of 42 relations.

The formalism follows a lexicalist approach for the sake of computational use, but this gave rise to many difficulties which needed to be solved. One example is the treatment of copulas. Copulas are treated as a dependent of a lexical predicate. This analysis had been chosen because many languages lack an overt copula in constructions like in Figure 2.5. Even English lacks the overt copula in raising-to-object or small clause constructions, like in Figure 2.6 (De Marneffe et al. (2014)).

De Marneffe et al. (2014) also argue that although compounds and modifications are strictly separated in a lexicalist approach, compounds cannot be treated uniformly. There are three types of relations for compounds in the UD formalism. `mwe` is used for fixed grammatical expressions (mwe(of, instead)), `name` labels proper names of multiple elements and `compound` is used for the remaining multiword expressions.

Prepositions and other case-marking elements are treated as a dependent of the noun it introduces or is attached to, for the sake of a uniform analysis.

[1] `http://universaldependencies.org/`

Clausal Argument Relations	**Description**
nsubj	Nominal subject
dobj	Direct object
iobj	Indirect object
ccomp	Clausal complement
xcomp	Open clausal complement
Nominal Modifier Relations	**Description**
nmod	Nominal modifier
amod	Adjectival modifier
nummod	Numeric modifier
appos	Appositional modifier
det	Determiner
case	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
conj	Conjunct
cc	Coordinating conjunction

Table 2.1: Selected dependency relations from the UD set ((Jurafsky and Martin, 2018b, p. 3)).

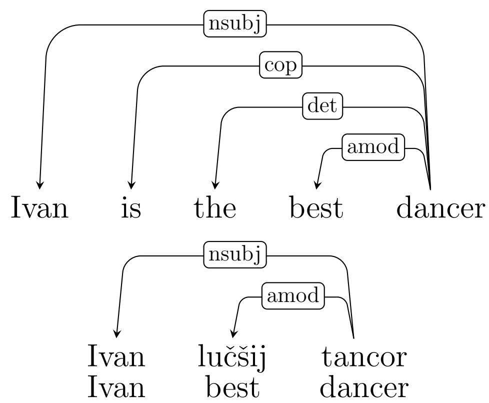


Figure 2.5: Dependency structure of the sentence 'Ivan is the best dancer' in English and Russian. Reproduced from (De Marneffe et al., 2014, p. 4586).

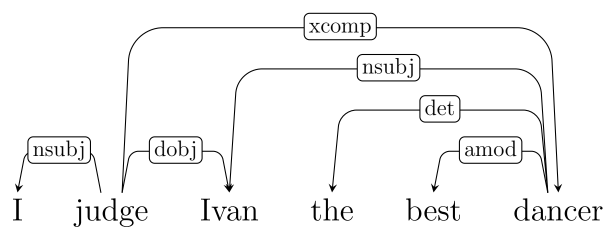


Figure 2.6: Dependency structure of the sentence 'I judge Ivan the best dancer'. Reproduced from (De Marneffe et al., 2014, p. 4587).

In Figure 2.7, the case marker is depending on the object. In cases where case markers are morphemes, such as in Figure 2.8, the morpheme, which is responsible for the case marking, is not separated from the noun as a case dependent. Instead, POS-tags are included in the representation to mark case.

These solutions provide similar treatment to constructions of different languages. For phenomena which appear only in a small subset of languages and generalization is not possible, special language-specific subrelations had been introduced.

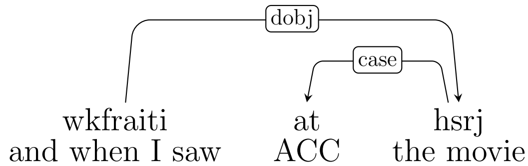
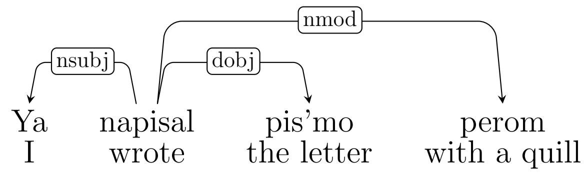


Figure 2.7: Dependency structure of the sentence 'And when I saw the movie' in Hebrew. Reproduced from (De Marneffe et al., 2014, p. 4587).



nsubj(napisal/VERB, Ya/NOUN-NOM)
dobj(napisal/VERB, pis'mo/NOUN-ACC)
nmod(napisal/VERB, perom/NOUN-INSTR)

Figure 2.8: Dependency structure of the sentence 'I wrote the letter with a quill' in Russian. Reproduced from (De Marneffe et al., 2014, p. 4587).

2.3 An error analysis

In this section we shall present a manual error analysis of three state-of-the-art dependency parsers. The results were presented at MSZNY2018 (Ács and Recski (2018)).

2.3.1 Background

As a response to the heightened interest in UD and dependency parsing, the 2017 edition of the Conference on Natural Language Learning (CoNLL) organized a shared task on *"Multilingual Parsing from Raw Text to Universal Dependencies"* Zeman et al. (2017). The training data – based on version 2.0 of the Universal Dependency dataset – consisted of 64 treebanks for 45 languages. Test treebanks contained at least 10,000 words for each language in the training set and an additional 4 surprise languages.

33 research groups submitted solutions to the task, their systems were ranked based on the macro-average of labeled attachment F-scores (LAS) achieved on each language. LAS matches require that a dependency is as-

	Overall			Hungarian		
	LAS	CLAS	UAS	LAS	CLAS	UAS
UnstableParser (Stanford)	76.30	72.57	81.30	77.56	76.08	82.35
C2L2 (Cornell)	75.00	70.90	80.32	76.55	74.36	82.07
IMS (Stuttgart)	74.42	70.18	79.90	73.55	70.87	79.90

Table 2.2: LAS, CLAS and UAS scores of all three parsers

signed to the correct pair of tokens in a sentence and with the correct label. In contrast, unlabeled attachment score (UAS) is more lenient in that it disregards edge labels. A third metric commonly used to evaluate dependency parsers is Content-word Labeled Attachment Score (CLAS), which only considers relation between content words and not function words or punctuation.

In Section 2.4.3 we shall analyze errors made by the top three parsers in the competition. The Stanford Dozat et al. (2017) and C2L2 (Cornell) Shi et al. (2017) teams submitted neural parsers that use LSTMs for representing input sentences; both of these systems leverage character-level representations to handle languages with rich morphologies. The Stuttgart IMS team's solution Björkelund et al. (2017) uses CRFs for POS/morphological tagging and a neural tagger for predicting supertags. Overall scores and scores for Hungarian data achieved by each of these three systems is presented in Table 2.2. Note that the gap between these three systems and the next teams is quite large so that Stanford, C2L2, and IMS are the top three systems based on any of the metrics presented here, and in particular for the Hungarian data.

2.3.2 Dependency parsing of Hungarian

The Hungarian section of the Universal Dependencies dataset has been created using the Szeged Dependency Treebank Vincze et al. (2010), challenges of the conversion process are described in Vincze et al. (2017). A manual error analysis similar to ours has been performed on Hungarian data before: Farkas et al. (2012) inspects 200 sentences from the output of Bohnet's parser Bohnet (2010) trained on the Szeged Dependency Treebank. A meaningful comparison of our analysis and theirs is not possible due to the differences between the two tasks: most error classes are specific to the respective annotation systems.

UnstableParser		C2L2		IMS	
punct	43	punct	46	punct	44
cc	13	cc	17	cc	17
det	11	det	16	det	11
advmod	9	conj	6	advmod	11
amod	7	conj-nmod	5	amod	7
conj	7	cc-advmod	5	amod-conj	6

Table 2.3: Types of erroneous edges

2.3.3 Evaluation

We inspected manually the analyses given by each of the three parsers on the first 50 sentences of the Hungarian test data. We grouped errors both by the types of dependency relations they involved and by the types of errors, i.e. the way in which the parsers misinterpreted the structure of a phrase, a clause, or an entire sentence. The number of erroneous edges in each output is similar in all three outputs: the Stanford data contained 208, C2L2 245, and IMS 261. Table 2.3 lists the top errors by edge type.

As we shall also see when grouping errors by their possible cause, punctuation is the single largest error class for each of the three systems. It has been questioned whether edges in a dependency graph that connect punctuation symbols to some word in the sentence are relevant to dependency structure, in fact the UD community is currently experimenting with the CLAS score as a means to disregard these edges when evaluating dependency parsers (Zeman et al., 2017, p.7). The `cc` relation is also ignored by CLAS scoring: it is responsible for connecting conjuncts such as *és* ('and'), *de* ('but'), etc. to some other word in the sentence.

Error types

We shall now describe the most common classes of errors, based on a close observation of each misinterpreted sentence. Besides punctuation and conjuncts we shall discuss 4 additional problem classes that are each responsible for between 2 and 7% of all observed errors (see Table 2.4 for counts).

	UnstableParser	C2L2	IMS
punct, cc	59	63	61
root	9 (15)	9 (17)	10 (19)
conj	9	9	7
modifier POS	8	5	13
structural ambiguity	6 (8)	6 (8)	5 (7)

Table 2.4: Number of occurrences of each error type (number of edges affected, if different)

Root elements

In nearly a fifth of all sentences observed, parsers assigned the `root` dependency to the wrong word, i.e. they failed to identify the main predicate of the sentence. These errors are worthy of attention not only because of their frequency but because they are usually responsible for several further erroneous edges – if the parser misses the main predicate, it is likely to miss relations of each of its dependents. An example of this phenomenon is shown in Figures 2.9 and 2.10, which show the gold and erroneous dependency analyses of the sentence in (1).

(1) – *Azért* *nem* *lehetett* *olyan* *rossz* *közelről* *élvezni* *a*
– Because not be-CAN-PAST so bad near-DEL enjoy-INF the
nehézsúly *Lewis-Holyfield-csúcsrangadóját!*
heavy-weight Lewis-Holyfield-faceoff-ACC!

It can't have been that bad, enjoying the Lewis-Holyfield faceoff from so close!

Coordination

Another group of errors involves coordinating conjunctions. In UD, conjunctions are treated asymmetrically: one of the coordinated elements is considered the head of the conjunction and others are connected only to this element (via the `conj` relation) but not to any other word in the sentence. Parser errors occur when these non-head elements of a conjunction are also connected to other words. These erroneous relations can be justified, since they reflect dependencies that actually hold between some word and *each*

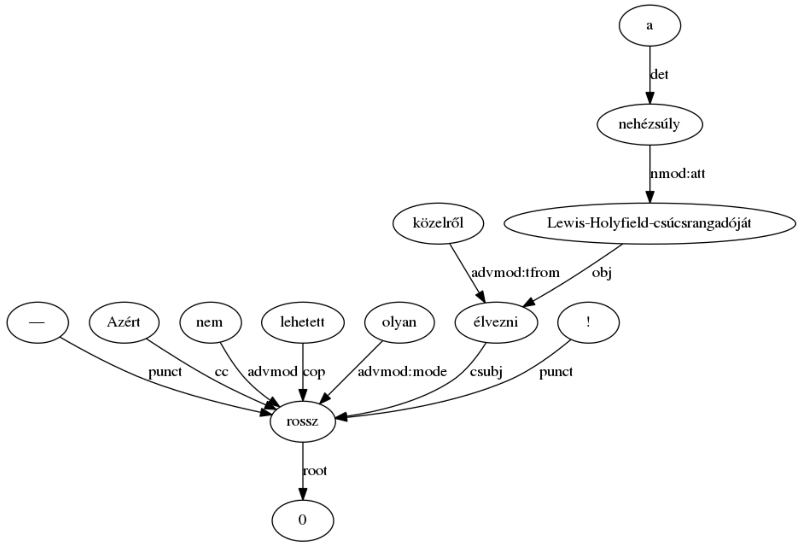


Figure 2.9: Gold analysis of (1)

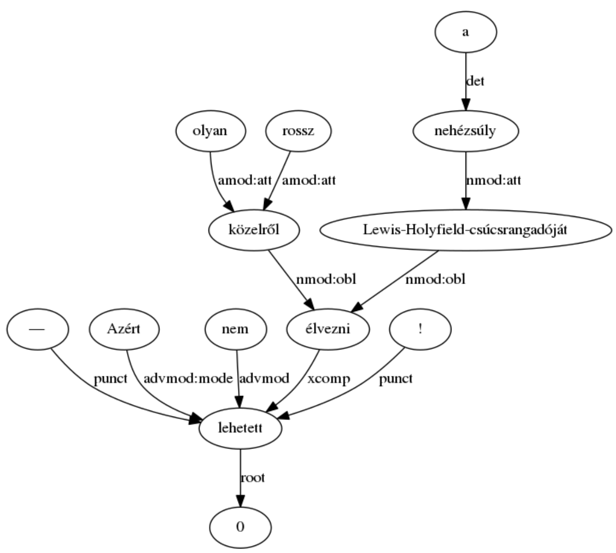


Figure 2.10: Incorrect analysis of (1) by the IMS parser

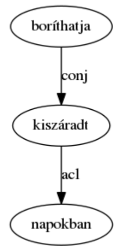


Figure 2.11: Partial analysis of (2)

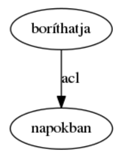


Figure 2.12: Partial analysis of (2) by both the C2L2 and the Stanford parser

element of a coordinated structure – nevertheless this treatment goes against UD conventions. An example is shown in Figure 2.12, a partial analysis of the sentence in (2).

(2) *Ezek is leginkább csak a januári napokban, amikorra*
these too mainly just the January-ATT day-PL-INE, when-SUBL
a fa már kiszáradt és egy csillagszóró is lángba
the tree already dry-out-PAST and a sparkler too flame-INE
boríthatja – mondta az alezredes.
cover-DEF – say-PAST the colonel.

But only in the days of January, when the tree is dry and a sparkler might burn it down – said the colonel.

Modifiers

The UD relations `nmod` and `amod` represent the dependencies between a noun and its nominal or adjectival modifier, respectively. Similarly, the `advmod` relation connects adverbs to predicates or modifiers. A large portion of errors were caused by parsers mixing the above three labels on edges that were otherwise correctly identified, i.e. they connected the modifiers to the right

	az	*Y2K*	*problémát*
	the	Y2K	problem-ACC
gold POS	DET	NOUN	NOUN
gold dependency	det	nmod	
IMS POS	DET	ADJ	NOUN
IMS dependency	det	amod	

Table 2.5: Gold and IMS analyses of a noun phrase

	türelmetlenül	újra	tárcsáz
	impatient-ESS	again	dial
	'dials again impatiently'		
gold POS	ADJ	ADV	VERB
gold dependency	amod	advmod	
IMS POS	ADJ	ADV	VERB
IMS dependency	advmod	advmod	

Table 2.6: Gold and IMS analyses of a noun phrase

word. Since the distinction between `nmod`, `amod`, and `advmod` is based entirely on the part-of-speech (POS) categories of dependents, one may expect that each of these errors are direct results of POS-tagging mistakes. In fact, out of 26 such errors in the three datasets (Stanford: 8, C2L2: 5, IMS: 13), only 14 (4, 2, 8) are in line with the above assumption: the output contains an incorrect POS-tag for the modifier word and the dependency label reflects the same mistake (an example is shown in Table 2.5). In the remaining 12 cases dependency labels were assigned incorrectly despite a correct POS-tag. In 4 cases, however, 2 made by the Stanford system and 2 by IMS, one may argue that the incorrect dependency labels are actually justified, while gold labels are a result of annotators' compliance with gold POS-tags that are linguistically questionable. An example is shown in Table 2.6.

Structural ambiguity

The final error group involves sentences that are structurally ambiguous and whose parses are consistent with a different constituent structure than the one reflected by the gold dependency annotation. The ambiguous phrase of one such sentence is shown in (3), with English paraphrases for both

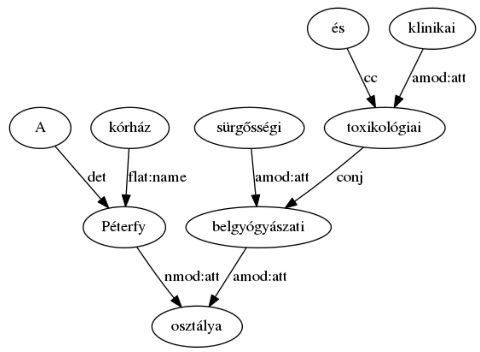


Figure 2.13: Gold analysis of (3).

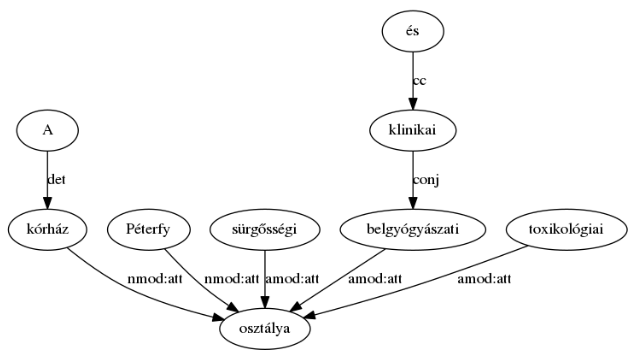


Figure 2.14: UnstableParser's analysis of (3).

possible readings. The two dependency structures are shown in Figure 2.13 and Figure 2.14.

(3) *a* *Péterfy* *kórház* *sürgősségi* *belgyógyászati* *és* *klinikai*
the Péterfy hospital emergency internal-medicine and clinical
toxikológiai *osztálya*
toxicology department-POSS

The department of emergency internal medicine and clinical toxicology
The emergency department of internal medicine and clinical toxicology

2.3.4 Comparison

Farkas et al. (2012) includes an error analysis on the output of Bohnet's parser trained on Hungarian data from the Szeged Dependency Treebank. Their method is similar to ours and involves manual inspection of 200 parser errors on the news section of the Szeged dataset.

2.3.5 Conclusion

We have presented the results of manual error analysis of three dependency parsers on a small sample of Hungarian data. We have identified several error classes that are in some ways technical: those concerning punctuations and conjuncts have little relevance to the dependency structure of content words and underline the necessity of alternative evaluation metrics like CLAS, while those involving coordinating conjunctions introduce edges that may be justifiable and might challenge UD's current treatment of coordination. Modifier relations have brought to light errors in POS-tagging and some possible inconsistencies in the gold standard data. Finally, we have seen examples of structural ambiguity, which remains one of the most challenging problems in syntactic analysis.we have presented the results of manual error analysis of three dependency parsers on a small sample of Hungarian data. We have identified several error classes that are in some ways technical: those concerning punctuations and conjuncts have little relevance to the dependency structure of content words and underline the necessity of alternative evaluation metrics like CLAS, while those involving coordinating conjunctions introduce edges that may be justifiable and might challenge UD's current treatment of coordination. Modifier relations have brought to light errors in POS-tagging and some possible inconsistencies in the gold standard data. Finally, we have seen examples of structural ambiguity, which remains one of the most challenging problems in syntactic analysis.

Chapter 3

Semantic parsing

In this chapter we provide a literature overview of semantic parsing, which means translating language to a formal representation of meaning. To introduce some issues of semantic representation, Section 3.1 begins with a short overview of Katz and Fodor's *The structure of a semantic theory*, then we describe some of the earliest graph-based models. Then in Section 3.2, we continue with describing some earlier and some more contemporary systems as well.

3.1 Theoretical background

Katz and Fodor (1963), in their paper *The structure of a semantic theory*, describe what form a semantic theory should take to accurately show the structure of the language. The main difficulty regarding its modeling is referred to as the projection problem: speakers are able to produce and understand an infinite number of sentences, based on the finite number of rules they know and the also finite number of sentences they have heard. They argue that a semantic theory should accomplish this with the same accuracy, using known elements and rules which combine them. It should detect non-structural ambiguity, semantic relations within the sentence, semantically anomalous sentences and also create paraphrases.

The authors conclude that the meaning must be somewhere else, not entirely in the grammar, because: 1. two sentences can have the same grammar descriptions even if they are different in meaning (The dog bit the man vs. The cat bit the woman), 2. the grammar can be entirely different when their meaning is essentially the same (The dog bit the man vs. The man was bitten by the dog).

In their view a theory cannot deal with discourse information, as it would be required to take all the knowledge of the speakers into account. As discourse is out of the question and grammar is insufficient in itself, they argue that a dictionary is needed. The entries of the dictionary consist of two main parts. One is the grammatical part, which is essentially the part of speech classification, and the other is the semantic part, which contain each of the distinct senses of the lexical item as a given part of speech. The semantic part also consists of two parts: semantic markers, which contain systematic semantic relations and are enclosed in parentheses in Figure 3.1, and the distinguishers, which are enclosed in brackets. The unenclosed element *noun* is the grammatical marker.

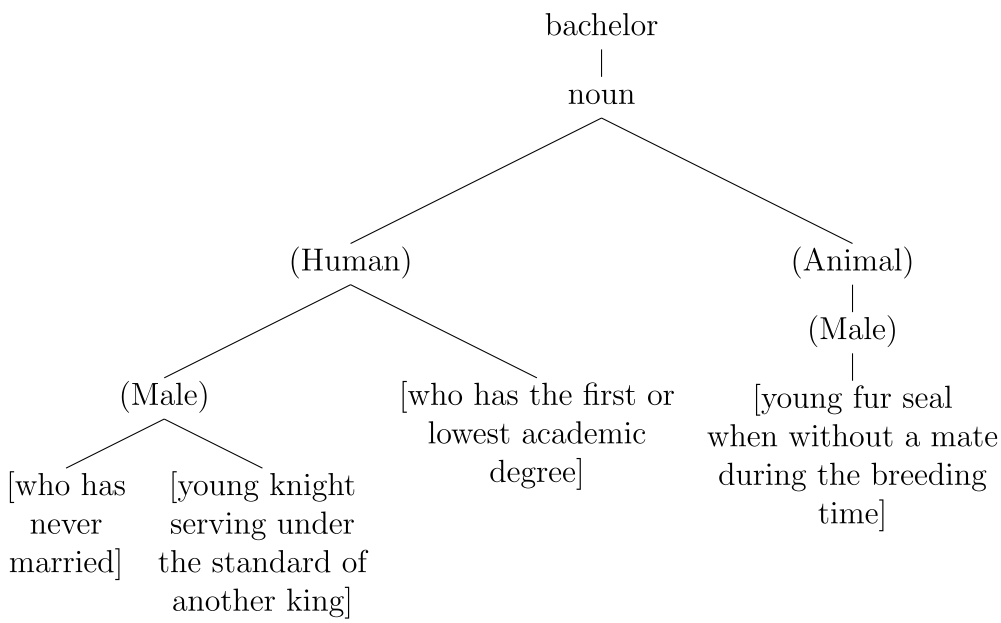


Figure 3.1: The structure of lexical items (Katz and Fodor (1963), page 186)

Speakers can understand senteces if some words are ambiguous in them. This means that the dictionary in itself is also insufficient. The authors state that there must be projection rules, which select the appropriate sense of the given word. These rules require the lexicon to be structured as in Figure 3.1. The projection rules amalgamate sets of paths dominated by a grammatical marker, so it assigns a set of readings to the concatenation of the items until it reaches the highest mark (i.e. the sentence). Word meaning representations also contain constraints and limitations on semantic contents on certain paths of the representation.

Quillian (1969) is among the earliest to propose a graph-based model: word meanings are represented as directed graphs of concepts which should be learned automatically. He assumes that human memory works in a similar way. This non-hierarchical structure consists of type and token nodes, which are organized into planes and appear multiple times in each concept when used in definitions. He discusses his theory through the architecture of a text comprehender program, which will be further discussed in Section 3.2.1.

Woods (1975), unlike Quillian, does not propose a complete semantic network, but argues about some problems about interpreting links in a network to represent knowledge through them, and some possible solutions about the issue. In his view, semantics can be seen in two inherently different ways. A linguist should find that a sentence can mean multiple things while some sentences mean nothing, and they can be translated to formal expressions. On the other hand, a philosopher is looking for the meaning of the formal expression, which is always true or false. Woods argues that these two approaches should be united in the semantic description of a natural language. He states that semantics is the relation between things and the linguistic expressions which denote them. These links connect fact together inside a large representation, which should contain every representation which can be linked to a sentence by a human. There must be an algorithm which retrieves this representation from the original sentence.

The author's reasoning is that a proposition's every paraphrase cannot be reduced to a canonical, standard form because in addition to defining the nodes and the links, their meaning should also be specified. A semantic network should also have attribute-value pairs where the attribute links have an intensional node which is linked to predicates and facts. This way, differences of intension and extension must be explicitly stated. Extension is essentially a function ("what does it mean to be red"), but an intensional representation is also necessary. However, the author admits that there are some problems with this approach. It cannot treat quantifiers properly, also a huge amount of elements must have an explicit meaning if the model aims to represent general knowledge.

In reaction to Woods, Brachman (1977) argued about the nature of concepts and stated that the uniformity of the notation is misleading in previous works. In his paper titled *What's in a concept: structural foundations for* *semantic networks*, he proposes a complicated structure of primitive links which can specify the concept as a set of attribute definitions. In a semantic model, such as Quillian's, nodes represent objects, assertions, events and classes of individuals. Those classes have subclasses, connected via the IS_A relation. The nodes for classes are called concept nodes. They contain the information that Woods referred to as extension. Instance nodes represent both the individuals and their supersets. A concept node and a value is connected via a named link. Such a link names the relationship, and according to Brachman, this is what should be called the concept. Properties should be described by binary relations.

In Schank and Rieger III (1974)'s opinion,it's important to differentiate between the domains of parsing (extraction of information, implicit and explicit) and inference (adding-on probably correct information) to understand natural language. Schank's theory of Conceptual Dependencies (CD) contains six main conceptual categories which describe how dependencies between them should be interpreted. The main categories are real world objects, real world actions (Table 3.1), attributes of objects, attributes of actions, times and locations. The four conceptual cases are OBJECTIVE, RECIPIENT, DIRECTIVE and INSTRUMENTAL.

Conceptualizations are governed by syntactic rules. Tenses are considered to be the link between an object and its state, or modifications of the main link between the actor and the action. CD also contains 14 language-independent inferences. A system which operates based on this theory will be further described in Section 3.2.2.

Sowa (1976)'s Conceptual Structures, which aim to provide a semantic basis for natural languages, consist of concepts and relations between them. Graphs are formed using rules which ensure that graphs can be translated to well-formed logical formulae. In this formalism, the basic primitive is the concept, which is represented by a box with a sort label. As concepts can differ in terms of how general they are, the sort labels are ordered. The other type of nodes is called conceptual relation, represented by a labeled circle. A well-formed graph is shown in Figure 3.2.

Well-formed graphs are like well-formed sentences or logical formulae; they don't have to be true or describe a plausible event. Concept graphs are mapped to predicate calculus. Constants or quantified variables are assigned to each concept by the operator ϕ, as in Figure 3.3.

This structure, as the author argues, is an important step towards a simpler user interface for computer applications.

Name	Description
ATRANS	the transfer of an abstract relationship, such as possession, ownership and control
PTRANS	the transfer of the physical location of an object
MTRANS	the transfer of mental information between animals or within an animal
MBUILD	the construction of new information from old information by an animal
CONC	the conceptualizing or thinking about an idea by an animal
PROPEL	the application of a physical force to an object
SMELL	-
SPEAK	-
LOOK-AT	-
LISTEN-TO	-
MOVE	-
GRASP	-
INGEST	-
EXPEL	-

Table 3.1: Schank's 14 ACTs. Reproduced from (Schank and Rieger III, 1974, p. 17.).

As for more recent graph-based systems, AMR and 4lang will be discussed in Section 3.2.3 and Section 3.2.4.

Although not a graph-based semantic model, we also mention WordNet (Miller (1995)) which is a large lexical database. It contains more than 166, 000 word form and sense pairs. It supports the syntactic categories noun, verb, adjective and adverb. The basic lexical relation is synonymy, as the database uses sets of synonyms, also referred to as synsets, to represent word senses. Other relations are antonymy, hyponymy, meronymy, troponymy and entailment, as in Table 3.2. Version 3.0 contains 117,798 nouns, 11,529 verbs, 22,479 adjectives and 4,481 adverbs. The average verb has 2,16 senses, and the average noun has 1,23 senses. (Jurafsky and Martin (2018a). Version 3.1 is currently available only for online search[1]. Figure 3.4 presents the lemma entry for the noun and verb *fox*.

WordNet had been used for various tasks, for example detecting and

[1] `http://wordnetweb.princeton.edu/perl/webwn`

NOUN RELATIONS		
Relation	**Definition**	**Example**
Hypernym	From concepts to superordinate	$\text{breakfast}^1 \rightarrow \text{meal}^1$
Hyponym	From concepts to subtypes	$\text{meal}^1 \rightarrow \text{lunch}^1$
Instance Hypernym	From instances to their concepts	$\text{Austen}^1 \rightarrow \text{author}^1$
Instance Hyponym	From concepts to concept instances	$\text{composer}^1 \rightarrow \text{Bach}^1$
Member Meronym	From groups to their members	$\text{faculty}^2 \rightarrow \text{professor}^1$
Member Holonym	From members to their groups	$\text{copilot}^1 \rightarrow \text{crew}^1$
Part Meronym	From wholes to parts	$\text{table}^2 \rightarrow \text{leg}^3$
Part Holonym	From parts to wholes	$\text{course}^7 \rightarrow \text{meal}^1$
Substance Meronym	From substances to their subparts	$\text{water}^1 \rightarrow \text{oxygen}^1$
Substance Holonym	From parts of substances to wholes	$\text{gin}^1 \rightarrow \text{martini}^1$
Antonym	Semantic opposition between lemmas	$\text{leader}^1 \Leftrightarrow \text{follower}^1$
Derivationally Related Form	Lemmas w/same morphological root	$\text{destruction}^1 \Leftrightarrow \text{destroy}^1$
VERB RELATIONS		
Hypernym	From events to superordinate events	$\text{fly}^9 \rightarrow \text{travel}^5$
Troponym	From events to subordinate event (often via specific manner)	$\text{walk}^1 \rightarrow \text{stroll}^1$
Entails	From events to the events they entail	$\text{snore}^1 \rightarrow \text{sleep}^1$
Antonym	Semantic opposition between lemmas	$\text{increase}^1 \Leftrightarrow \text{decrease}^1$
Derivationally Related Form	Lemmas w/same morphological root	$\text{destroy}^1 \Leftrightarrow \text{destruction}^1$

Table 3.2: Noun and verb relations in WordNet. (Jurafsky and Martin, 2018a, p. 7.)

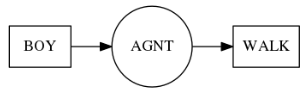
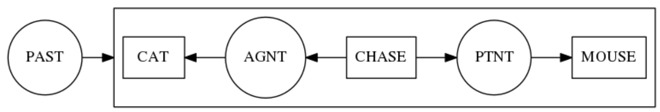


Figure 3.2: The representation of the phrase 'boy walking'. Reproduced from (Sowa, 1976, p. 338.).



$$past((\exists x)(\exists y)(\exists z)(cat(x) \wedge chase(y) \wedge mouse(z) \wedge agnt(y,x) \wedge ptnt(y,z)))$$

Figure 3.3: The representation of the sentence 'A cat chased a mouse'. Reproduced from (Sowa, 1992, p. 80.).

interpreting English puns (Miller et al. (2017)) and measuring word similarity (Camacho-Collados et al. (2017)) in SemEval 2017 shared task.

3.2 Systems

This chapter describes some systems based on the models explained in the previous chapter. In section 3.2.1 we provide an overview of Quillian's teachable language comprehender, followed by Schank's program which performs inference tasks in Section 3.2.2. Then we describe some of the more contemporary systems, such as AMR systems in Section 3.2.3 and 4lang in Section 3.2.4.

3.2.1 Quillian's teachable language comprehender

Quillian (1969), in his paper titled *The teachable language comprehender*, provides a complete model of language understanding through the architecture of his program's memory. The teachable language comprehender (TLC) is able to understand written text, although a quite limited pool of it. TLC requires a human overseer to provide factual information and form tests, which mandate certain features to be present in the input (e.g. word order, word ending, etc). To accomplish this, the human monitor must use a language which is similar to a string manipulation language.

NOUN
fox^{1} - alert carnivorous mammal with pointed muzzle and ears and a bushy tail; most are predators that do not hunt in packs
fox^{2} - a shifty deceptive person; synonyms: dodger,slyboots
fox^{3} - the grey or reddish-brown fur of a fox
fox^{4} - Charles James Fox (English statesman who supported American independence and the French Revolution (1749-1806))
fox^{5} - George Fox (English religious leader who founded the Society of Friends (1624-1691))
fox^{6} - a member of an Algonquian people formerly living west of Lake Michigan along the Fox River
fox^{7} - the Algonquian language of the Fox

VERB
fox^{1} - deceive somebody; synonyms: flim-flam, play a joke on, play tricks, trick, fob, pull a fast one on, play a trick on
fox^{2} - be confusing or perplexing to; cause to be unable to think clearly; synonyms: confuse, throw, befuddle, fuddle, bedevil, confound, discombobulate
fox^{3} - become discolored with, or as if with, mildew spots

Figure 3.4: A portion of the WordNet 3.1 entry for the noun and verb *fox*.

The memory consists of two main parts: the units and the properties. Units represent things that can be represented by a single word, a sentence or a noun phrase in English. Properties specify predications, for example verb phrases, relative clauses or modifiers. The words are located in a dictionary, outside of the memory. The items' more general forms (such as *person* for *client*) are called the supersets of the items. In cases of words which cannot be assigned to a more general concept, the NIL unit serves as their superset. The words in the dictionary are connected to the units which describe its meaning, in the memory, via pointers. The unit's first item must be a pointer to the superset, and it also contains pointers to the describing properties, as in the example of Figure 3.5.

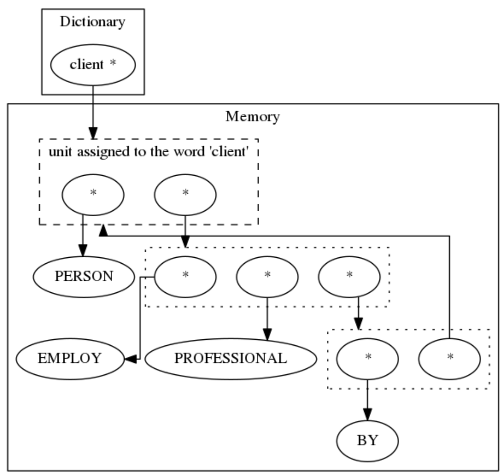


Figure 3.5: A piece of information in the memory. Reproduced from (Quillian, 1969, p. 13.). Stars represent pointers.

With this structure, an infinite number of new units can be made (the existing one becomes the superset, it is linked to the new concept's properties). For example, there's the unit Joe Smith, but if one wants to talk about him when he was three years old, a new unit should be generated, whose superset is the original Joe Smith and its property is his age.

Properties are attribute-value pairs, so prepositions and verbs and their objects can be handled. In this memory structure, the intersections can quickly be found via breadth-first search; it marks the already found units with the concept (activation tagging): first it founds the nearest intersection, later the further ones.

To sum up how text comprehending happens: when it sees a new word, it makes a new, empty unit for it, then looks for the candidates, which are the possible senses of the word, and finally finds the correct properties. There are three problems with this approach: 1. a word can have multiple meanings, 2. how to compile the properties which describe the new concept, 3. references should be also understood. The author solves them by adding a new empty unit for every new word, with a list of pointers to the representations of the possible meanings in the memory. Besides activation tagging, it uses the aforementioned form tests, which contain syntactic information. For example, for the phrase *lawyer's client*, the form tests check whether the word *lawyer* precedes the word *client* or the word *lawyer* has the *'s* ending.

Ambiguity is handled with the help of superset intersections. Three types of superset intersections exist: 1. the intersection of the superset-chain of two properties 2. the intersection of two possible interpretation's superset chains 3. the intersection of superset chains of a property and a possible interpretation. The latter gives relevant information regarding the meaning of the text.

The data and the form tests are generalized in the memory, so if it understands *lawyer's client*, it can also understand *woman's client*. From *lawyer*, it reaches the concept *person* via *professional*, then it can find *woman* through *person*. Form tests work similarly.

TLC can also comprehend more complex sentences. For example, while dealing with the sentence *lawyer's young client*, it cannot find relevant form tests for *lawyer's young*, so it stores the relevant properties (for example EMPLOY), then in the next iteration it finds that *young* is connected to the property AGE, and *client* (PERSON) has the AGE property, so they can be connected. Finally it decides which word is the head with the help of form tests.

3.2.2 Schank's inference making program

Schank and Rieger III (1974)'s system was not designed to excel in text comprehension, rather it aims to be an easily extendable and theoretically accurate program. The theory of Conceptual Dependencies was briefly introduced in Section 3.1.

The propositional information is stored in a list in the memory, in predicate-conceptual slots pairs. The stored proposition is called a bond, it is stored under a so-called superatom. This way propositions resemble simple concepts. Simple concepts are defined by an occurrence set, which is a set of pointers to superatoms. The knowledge about a concept is essentially the occurrence set, pointed to the propositions in the superatom.

Superatoms have other characteristics, such as STRENGTH, MODE, TRUTH, REASONS, OFFSPRING and RECENCY. STRENGTH indicates the credibility of the proposition, MODE contains its truth value, TRUTH is used when the proposition is true at the present time, REASONS are the superatoms used to infer the given proposition and OFFSPRING is the inverse of REASONS. Another function is RECENCY, which is shared with simple concepts, and contains the value of the system clock.

Inferences, which are lambda-functions under predicates, have the same structure. Pattern matching happens in these lambda-functions as the program does the testing. Inferencing is done in breadth-first order.

In the example sentence *John hit Mary*, the conceptualization looks like as in Figure 3.6. After the memory established the referents of each concept, the conceptualization takes a shorter form, where C001 refers to John's hand in the memory and C002 stands for the time of the event. Next the memory divides the conceptualization into subpropositions, as seen in Figure 3.7. The causal relation (number 9) becomes the superatom. Inferences are made based on inference patterns in the memory. For example, John has a movable hand since he propelled his hand. Because John propelling his hand resulted in physical contact with Mary, she must have been hurt.

3.2.3 AMR systems

AMR (Banarescu et al. (2013)) aims to be a simple representation intended to represent any English sentences. AMRs are directed graphs which use PropBank framesets. Proposition Bank is an annotated corpus of semantic roles and focus on the argument structure of the verbs (Palmer et al. (2005)). An example is shown in Figure 3.8.

The same AMR is used for sentences with the same meaning, regardless its syntactic form. They are intended to use for deriving meanings from strings and vice versa.

Nodes represent entities, properties, events or states. Leaves are labeled with concepts, which are English words, PropBank framesets or keywords. Keywords are special entity types, quantities or logical conjunctions. Relations link entities. AMR uses approximately 100 relations (Banarescu et al. (2013)) it can treat general semantic relations, co-reference, questions,

```
((CAUSE ((PROPEL C1: ((ISA_#PERSON)(NAME_ "JOHN"))
C2: ((ISA_#HAND)(PART_ C1))
C1
C3: ((ISA_#PERSON)(NAME_ "MARY"))
))
(PHYSCONT C2 C3))
) (TIME_ C4:)(ISA_#TIME)(BEFORE_ #NOW)))
)

((CAUSE ((8 PROPEL #JOHN #C001 #JOHN #MARY))
((PHYSCONT #C001 #MARY)))
(TIME_ #C002))
```

Figure 3.6: Conceptualization for the sentence "John hit Mary". (Schank and Rieger III, 1974, p. 35.).

modals and negations. Relations also have their inverses.

However, AMR has many limitations. It doesn't contain inflectional morphology and articles, also it doesn't have the universal quantifier. Words like *all* modify their head concept. It does not distinguish between real or hypothetical events, and cannot tell whether the event happens in the past, present or future.

Flanigan et al. (2014)'s JAMR is the first approach for AMR parsing and serves as a baseline for future works.

SemEval shared tasks

2016

Task 8 of SemEval 2016 shared task (May (2016)) required participants to generate graphs for English sentences in the news and forum domain. As training data, the corpus LDC2015E86 was made available, which contains 19,572 sentences and the corresponding 31,830 AMRs. For other resources, see (May, 2016, p. 1064). The evaluation data contained 1,053 previously unseen English sentences and AMR annotations. 11 systems were submitted to the task. The results of the top three systems are shown in Table 3.3.

Brandeis/cemantix.org/RPI (Wang et al. (2016)) and RIGA (Barzdins

1. JOHN PROPELLED SOMETHING
2. A HAND WAS PROPELLED
3. JOHN MOVED SOMETHING
4. A HAND WAS MOVED
5. A HAND IS PART OF JOHN
6. SOMETHING WAS PROPELLED FROM JOHN TO MARY
7. A HAND AND MARY WERE IN PHYSICAL CONTACT
8. JOHN PROPELLED HIS HAND
9. 8 CAUSED 7
10. IT WAS BEFORE "NOW" THAT 1 - 9 OCCURRED

Figure 3.7: Subpropositions for the sentence "John hit Mary". (Schank and Rieger III, 1974, p. 37.).

Frameset **accept.01** “take willingly”
Arg0: Acceptor
Arg1: Thing accepted
Arg2: Accepted-from
Arg3: Attribute

Figure 3.8: The argument structure of the verb *accept* (Palmer et al., 2005, p. 75.).

and Gosko (2016)) are based on Wang et al. (2015)'s CAMR, which was made available to the participants as a strong baseline parser alongside JAMR. CU-NLP (Foland and Martin (2016)) used recurrent neural networks.

The results and the relative lack of interest in participation led to the conclusion that AMR parsing is a challenging task so the organizers decided to conduct another competition in 2017.

2017

The SemEval 2017 shared task (May and Priyadarshi (2017)) had two subtasks on AMR parsing: in the parsing subtask, participants had to generate AMR graphs for English sentences of biomedical domain, and in the generation subtask participants were asked to produce English sentences from AMR graphs in the news/forum domain. For training the systems, the Bio-AMR

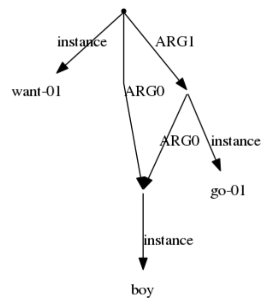


Figure 3.9: Representing the meaning of “The boy wants to go”(Banarescu et al. (2013), page 179).

	Full AMR	Instances	Attributes	Relations
RIGA	0.6196	0.7298	0.6288	0.5507
Brandeis/cemantix.org/RPI	0.6195	0.7433	0.6043	0.5494
CU-NLP	0.6060	0.7338	0.6141	0.5323

Table 3.3: The best three systems of the 2016 shared task

v0.8 and LDC2016E25 datasets were made available. The Bio-AMR v0.8 contains 6,452 AMR annotations of sentences from cancer-related papers, and LDC2016E25 contains 39,260 sentences and the corresponding 51,402 AMRS. For the additional resources available, see ((May and Priyadarshi, 2017, p. 537.)).

In the parsing subtask, 5 teams participated, one of them submitted two systems. The team The Meaning Factory (van Noord and Bos (2017)) 's TMF-1 is a character-level sequence-to-sequence deep learning model, while TMF-2 is based on CAMR (Wang et al. (2016)). UIT-DANGNT-CLNLP submitted two wrapper layers for CAMR (Nguyen and Nguyen (2017)). Oxford implemented a neural encoder-decoder (Buys and Blunsom (2017)) and RIGOTRIO submitted their parser from the 2016 task (Gruzitis et al. (2017)) after implementing extensions for this task. CMU used their system of 2016 (Flanigan et al. (2016a)) and refused to submit a new system description paper. The results are presented in Table 3.4.

In Table 3.4, 'Unlabeled' indicates that all argument labels were replaced with a single general label. In 'No WSD', the PropBank frames which indicate different senses, are conflated. For 'NER', only named entities are scored, and 'Wiki' means that only wikifications are scored. 'Negation' means

	Smatch	Unlab.	No WSD	NER	Wiki
TMF-1	0.46	0.5	0.46	0.51	0.46
TMF-2	0.58	0.63	0.58	0.58	0.4
UIT-DANGNT-CLNLP	**0.61**	**0.65**	**0.61**	**0.66**	0.35
Oxford	0.59	0.63	0.59	**0.66**	0.18
CMU	0.44	0.47	0.44	0.48	0.59
RIGOTRIO	0.54	**0.59**	0.54	0.46	0
	Smatch	Neg.	Concepts	Reent.	SRL
TMF-1	0.46	0	0.63	0.29	0.43
TMF-2	0.58	0.24	0.76	0.35	0.54
UIT-DANGNT-CLNLP	**0.61**	0.24	**0.78**	0.37	0.56
Oxford	0.59	0.27	0.74	**0.43**	**0.57**
CMU	0.44	**0.33**	0.65	0.27	0.41
RIGOTRIO	0.54	0.31	0.71	0.34	0.51

Table 3.4: Main parsing results: For Smatch (Cai and Knight (2013)), a mean of ten runs with ten restarts per run is shown; standard deviation was about 0.0003 per system. For the remaining ablations, a single run was used (May and Priyadarshi (2017) page 539.)

that only concepts with an outgoing polarity are scored, while in 'Concepts' relations are omitted. In 'Reentrancies', only concepts with at least two incoming relations are scored. Finally, 'Semantic Role Labeling' (SRL) means that only relations corresponding to PropBank are scored.

4 teams submitted their systems for the generation subtask. The system CMU was the same as the year before (Flanigan et al. (2016b)),Sheffield inverted previous work on transition-based parsers (Lampouras and Vlachos (2017)), RIGOTRIO uses transformation-based rules for 10% of the AMRs, the remaining were converted to text using the JAMR tool. FORGe (Mille et al. (2017)) used rule-based graph-transducers, and ISI is an extension of Pourdamghani et al. (2016). Evaluation results are presented in Table 3.5.

Beside BLEU, three other metrics were used for evaluation. Win + tie percentage indicate the sum of better pairwise comparisons and equal comparisons. Win considers ties as loses, and TrueSkill (Sakaguchi et al. (2014)), which is a metric for player rankings of videogame competitions rewards unexpected events more than expected ones.

The fact that so few teams participated in the task indicates that AMR

	Win	Win + Tie	Trueskill	BLEU
RIGOTRIO	**54.91**	**81.49**	**1.07**	18.82
CMU	50.36	72.48	0.85	**19.01**
FORGe	43.64	57.43	0.45	4.74
ISI	26.05	38.39	-1.19	10.92
Sheffield	8.38	21.16	-2.20	3.32

Table 3.5: Main generation results (May and Priyadarshi (2017), page 542)

parsing still counts as a very challenging task. Although parsing of biomedical domain seemed more difficult, the results are no worse than the other subtask's. The best teams used the same technology that dominated the 2016 task.

3.2.4 4lang

4lang is a formalism which builds directed graphs for semantic representation. In such graphs, nodes stand for concepts, which do not have any grammatical attributes, and contains shared knowledge of competent speakers. These can be connected via three types of edges, namely 0, 1 and 2 (Kornai et al. (2015). 0-edge represents attribution ($apple \xrightarrow{0} delicious$), the IS_A relation ($emu \xrightarrow{0} bird$) and unary predication ($cat \xrightarrow{0} meow$). 1 and 2-edges connect binary predicates to their arguments ($John \xleftarrow{1} buy \xrightarrow{2} book$). The most common binaries can be found in Table 3.6.

The 4lang library[2] contains tools for building directed graphs from raw text (text_to_4lang) and dictionary definitions (dict_to_4lang). The core module of the 4lang library, dep_to_4lang obtains dependency relations from text by processing the output of the Stanford parser (DeMarneffe et al. (2006)). A mapping was created manually from Stanford dependencies (De Marneffe and Manning (2008)) to subgraphs of nine possible graph configurations (Table 3.7). We present the reimplementation of this mapping with several modifications and enhancements in Section 4.4.

4lang is also a name of a manually created concept dictionary (Kornai and Makrai (2013)) which contains more than 2000 definitions of language-independent concepts. Definitions are generic and contain only the core information (except true homonyms), so they are suitable for all possible

[2] `https://github.com/kornai/4lang`

HAS	`shirt` $\xleftarrow{1}$ `HAS` $\xrightarrow{2}$ `collar`
IN	`letter` $\xleftarrow{1}$ `IN` $\xrightarrow{2}$ `envelope`
AT	`move` $\xleftarrow{1}$ `AT` $\xrightarrow{2}$ `way`
CAUSE	`humor` $\xleftarrow{1}$ `CAUSE` $\xrightarrow{2}$ `laugh`
INSTRUMENT	`sew` $\xleftarrow{1}$ `INSTRUMENT` $\xrightarrow{2}$ `needle`
PART_OF	`leaf` $\xleftarrow{1}$ `PART_OF` $\xrightarrow{2}$ `plant`
ON	`smile` $\xleftarrow{1}$ `ON` $\xrightarrow{2}$ `face`
ER	`slow` $\xleftarrow{1}$ `ER` $\xrightarrow{2}$ `speed`
FOLLOW	`Friday` $\xleftarrow{1}$ `FOLLOW` $\xrightarrow{2}$ `Thursday`
MAKE	`bee` $\xleftarrow{1}$ `MAKE` $\xrightarrow{2}$ `honey`

Table 3.6: Most common binaries in the `4lang` dictionary (Recski, 2018, p. 4.).

uses of the given word. 4lang gets its name for the fact that its concepts are mapped in four languages (Hungarian, English, Latin, Polish).

Dependency	Edge
amod advmod npadvmod acomp dep num prt	$w_1 \xrightarrow{0} w_2$
nsubj csubj xsubj agent	$w_1 \underset{0}{\overset{1}{\rightleftharpoons}} w_2$
dobj pobj nsubjpass csubjpass pcomp xcomp	$w_1 \xrightarrow{2} w_2$
appos	$w_1 \underset{0}{\overset{0}{\rightleftharpoons}} w_2$
poss prep_of	$w_2 \xleftarrow{1} \texttt{HAS} \xrightarrow{2} w_1$
tmod	$w_1 \xleftarrow{1} \texttt{AT} \xrightarrow{2} w_2$
prep_with	$w_1 \xleftarrow{1} \texttt{INSTRUMENT} \xrightarrow{2} w_2$
prep_without	$w_1 \xleftarrow{1} \texttt{LACK} \xrightarrow{2} w_2$
prep_P	$w_1 \xleftarrow{1} \texttt{P} \xrightarrow{2} w_2$

Table 3.7: Mapping from Stanford dependency relations to 4lang subgraphs (Recski, 2018, p. 12.).

Chapter 4

Parsing with IRTGs

Many graph formalisms were discussed in the previous chapters, such as graph-based dependency parsing, AMR and 4lang. As these formalisms use graph transformations, the task of semantic parsing can be viewed as a graph transformation problem. This chapter explains s-graphs and the use of interpreted regular tree grammars (IRTG) for implementing graph transformations. Section 4.1 provides a description of IRTGs in general, followed by Section 4.2 which describe the Algebraic Language Toolkit (Alto). Then we present our reimplementation and extension of the dep_to_4lang functionality. Our system contains several enhancements, such as UD-conformity and the treatment of the UD relation `case`.

4.1 IRTGs and s-graphs

In his paper titled *Semantic construction with graph grammars*, Koller (2015) discusses interpreted regular tree grammars (IRTGs). The grammar consists of rewrite rules embedded within operations of one or more algebras. Thus, when a rule gets applied on one algebra, the corresponding operations are executed on objects in each algebra. When processing rules, first a derivation tree is built using regular tree grammars (RTGs), which are for replacing nonterminals with the use of production rules, as in Figure 4.1. Formally, a grammar like this is a structure $G = (N, \Sigma, P, S)$, where N is a signature of nonterminal symbols, Σ is a signature of terminal symbols, $S \in N$ is a distinguished start symbol, and P is a finite set of productions of the form $B \rightarrow t$, where B is a nonterminal symbol, and $t \in T_{N \cup \Sigma}$ (Gécseg

RTG rule	homomorphisms
S -> r_1(NP,VP)	1: $x_1 \bullet x_2$ 2: $x_1 \bullet x_2$
VP -> r_2(V, NP)	1: $x_1 \bullet x_2$ 2: $x_2 \bullet x_1$
NP -> r_3	1: John 2: Hans
NP -> r_4	1: the box 2: die Kiste
NP -> r_5	1: opens 2: öffnet

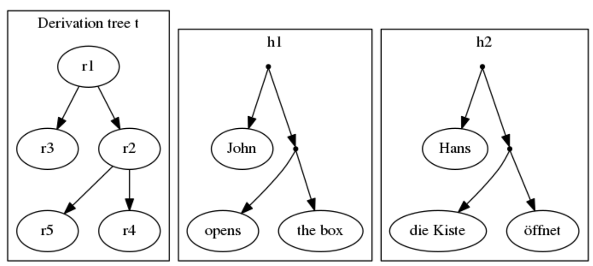


Figure 4.1: An IRTG with an example derivation (Koller (2015), page 4)

and Steinby (1997)). After the tree is built, it is interpreted as a tuple $(a_1, ..., a_k) \in A_1 \times ... \times A_k$ of elements from the algebras $A_1, ..., A_k$. The derivation of elements is done by mapping. The derivation tree (t) is mapped to a term using tree homomorphism function (h). It expands rules from the initial set of trees to the others. Then the term is evaluated over the algebra. In the example of Figure 4.1, homomorphisms are string concatenations, corresponding to RTG rules.

An algebra that has been previously used for semantic parsing is the s-graph algebra or HR algebra (Courcelle (1993), mentioned in Koller (2015)). An s-graph grammar is an IRTG where at least one algebra is the s-graph grammar (Groschwitz et al. (2015)). An s-graph's nodes may be marked with a set of source names from a fixed finite set. Intuitively, source names identify nodes that should be merged when subgraphs are merged by algebra operations. Sources of the graph are the nodes which carry source names.

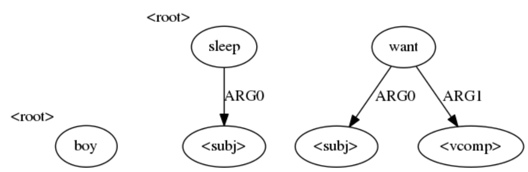


Figure 4.2: S-graph examples (Koller (2015), page 5)

An s-graph can consist of a single node which is the root source. The s-graph algebra defines three operations for combining graphs: rename, forget and merge. The result of the rename operation is a graph which is like the original except given source names have been changed to another source name. Forget results in a graph which is like the original except a given source name is removed from all source nodes with that name. Any other source names are retained. Merge returns a graph that contains all the nodes and edges of its operands such that nodes with the same source names are mapped to the same nodes in the result, having all the adjacent edges of these original nodes. The process of combining subgraphs together will be further discussed in Section 4.2. Two subgraphs cannot be merged if they share the same edges. A graph parser needs to handle extensible subgraphs only. A subgraph is extensible if there is another subgraph such that by merging these two subgraphs the result is a graph that contains all the edges in the base graph. For a more formal explanation of the HR algebra discussed in this section, see Koller and Kuhlmann (2011).

In semantic applications, e.g. Koller (2015), source names correspond to the semantic argument positions of the given grammar. An example of its use for semantic representation is shown in Figure 4.2. The argument structure of the verbs is represented as follows: the word itself is the `root` source node, which indicates the starting point of the representation. The other nodes are, in the case of the verb *want*, the `subj` and `vcomp`-sources, respectively. The `root` sources of the arguments will be inserted there, as seen in Figure 4.3.

Other formalisms can also be used for manipulating graphs. Chiang et al. (2013) discusses parsing with hyperedge replacement grammars (HRGs). HRG is also a context-free rewriting formalism. Koller (2015) points out some differences between IRTGs and HRGs. HRGs are used for manipulating hypergraphs. Such graphs may contain hyperedges with an arbitrary number of endpoints, which are labeled with nonterminal symbols. Rule

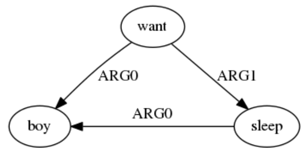


Figure 4.3: S-graphs of Figure 4.2 combined (Koller (2015), page 6)

applications replace a hyperedge with the graph on the right side, thus the endpoints of the nonterminal hyperedge become the *external nodes* (Koller (2015)) of the graph. A main difference between the two formalisms is that HRG rules build graphs in a top-down way, while IRTGs work with a bottom-up approach, using simple graph-combining operations and also they name the semantic argument positions, unlike HRGs.

4.2 Alto

The Algebraic Language Toolkit, or Alto[1] (Gontrum et al. (2017)) is an open-source parser for IRTGs discussed in Section 4.1. It is able to express a variety of algorithms generically as operations on tree grammar.

When constructing a grammar file for Alto, one must specify the interpretations. The interpretation consists of an algebra and a mapping from tree nodes to terms and variables. In the example case of `interpretation graph: de.up.ling.irtg.algebra.graph.GraphAlgebra`, a graph algebra is specified.

As discussed in Section 4.1, the HR grammar uses the operations `merge`, `rename` and `forget` for combining subgraphs. A basic rule from our grammar can be seen in Figure 4.4. `_nsubj` is the name of the abstract operation. In the first line, RTG rule $X \rightarrow \_nsubj(X, X)$ is interpreted as two subgraphs which we intend to merge with the base graph `A: "(g<gov> :1 (d<dep> :0 g))"`. Initially, the first subgraph X is merged to the base graph, by renaming the root source node of it to `gov`. Then the second subgraph X is merged, its root is renamed to `dep`. After merging the subgraphs together, the label `dep` is forgotten, as it becomes an internal node and later rules won't refer to it. The final step is to rename the `gov` node to `root`. The process is illustrated on Figure 4.5. Figure 4.6 illustrates this further on the sentence 'John loves

[1]https://bitbucket.org/tclup/alto

```
X -> _nsubj(X, X)
[graph] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :nsubj (d<dep>))"),
            r_dep(?2)
        )
    )
)
[fourlang] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :1 (d<dep> :0 g))"),
            r_dep(?2)
        )
    )
)
```

Figure 4.4: An example from our grammar, illustrating the operations of the HR algebra. The grammar is discussed further in Section 4.4.

Mary'.

4.3 Alto for AMRs

The authors of Groschwitz et al. (2015) evaluated their system (i. e. Alto) on version 1.4 of the "Little Prince" AMR-bank[2]. It consists of 1562 manually annotated sentences. Their experiments evaluated parsing times on the same dataset. First they tested a top-down and a bottom-up algorithm. The bottom-up algorithm outperformed the top-down approach as the latter spent more time analyzing ungrammatical graphs, and needed to be aborted after the runtime grew too large. Then the authors compared their system to Bolinas (Andreas et al. (2013)), a bottom-up graph parsing system based on Chiang et al. (2013) 's algorithm for HRG. Alto outperformed the previously

[2] amr.isi.edu

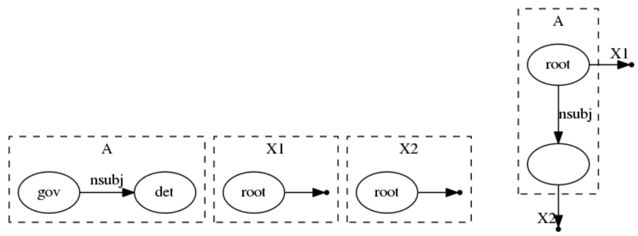


Figure 4.5: The state of the three subgraphs before and after executing the operations of Figure 4.4.

state of the art Bolinas by several orders of magnitude.

Koller (2015) illustrates the treatment of complements using the grammar of Table 4.1. For representing the sentence *The boy sleeps*, a derivation tree is generated. The homomorphism h_s projects the tree to the term $h_s(t)$, which will result in the string *the boy sleeps* in the string algebra. Simultaneously, the other homomorphism, h_g projects the derivation tree to $h_g(t)$, which creates the s-graph on Figure 4.7. The same grammar is also able to analyze sentences with control verbs. For deriving the sentence *The boy wants to sleep* (Figure 4.8), the rule for *sleep* must be used as the VP argument of want_1. Before merging G_1 and G_3, the root source must be renamed to `vcomp`, so that argument position can be filled. G_3's subj source is not renamed, so the merge operation can fuse the subj-arguments of *sleep* and *wants*, resulting the s-graph of Figure 4.8. Raising verbs (*wants* in the example) can also be handled by this grammar. The rule passes its grammatical object to its complement, where `subj` is renamed to `obj`, so the object will fill the role of the subject.

Koller (2015) also explains modification. AMR models modification as follows: the modification edges point from the modifier to the modifiee. An s-graph grammar can represent it by merging `root`-sources without renaming them beforehand. In the grammar of Table 4.9, the author uses shorthand notations for basic s-graphs. For example, the rule for `coord` merges its renamed arguments with a three-noded s-graph. One node is a `root`-source and the other two are unlabeled nodes for the source names 1 and 2. This graph is abbreviated as G_{coord}. Similar notations refer to similar graphs in the rules of *snores* and *sometimes*. The derivation can be seen in Figure 4.9. The meaning of the relative clause is represented by the subtree starting at `rc`. It combines the s-graph for the relative pronoun, which is a single unla-

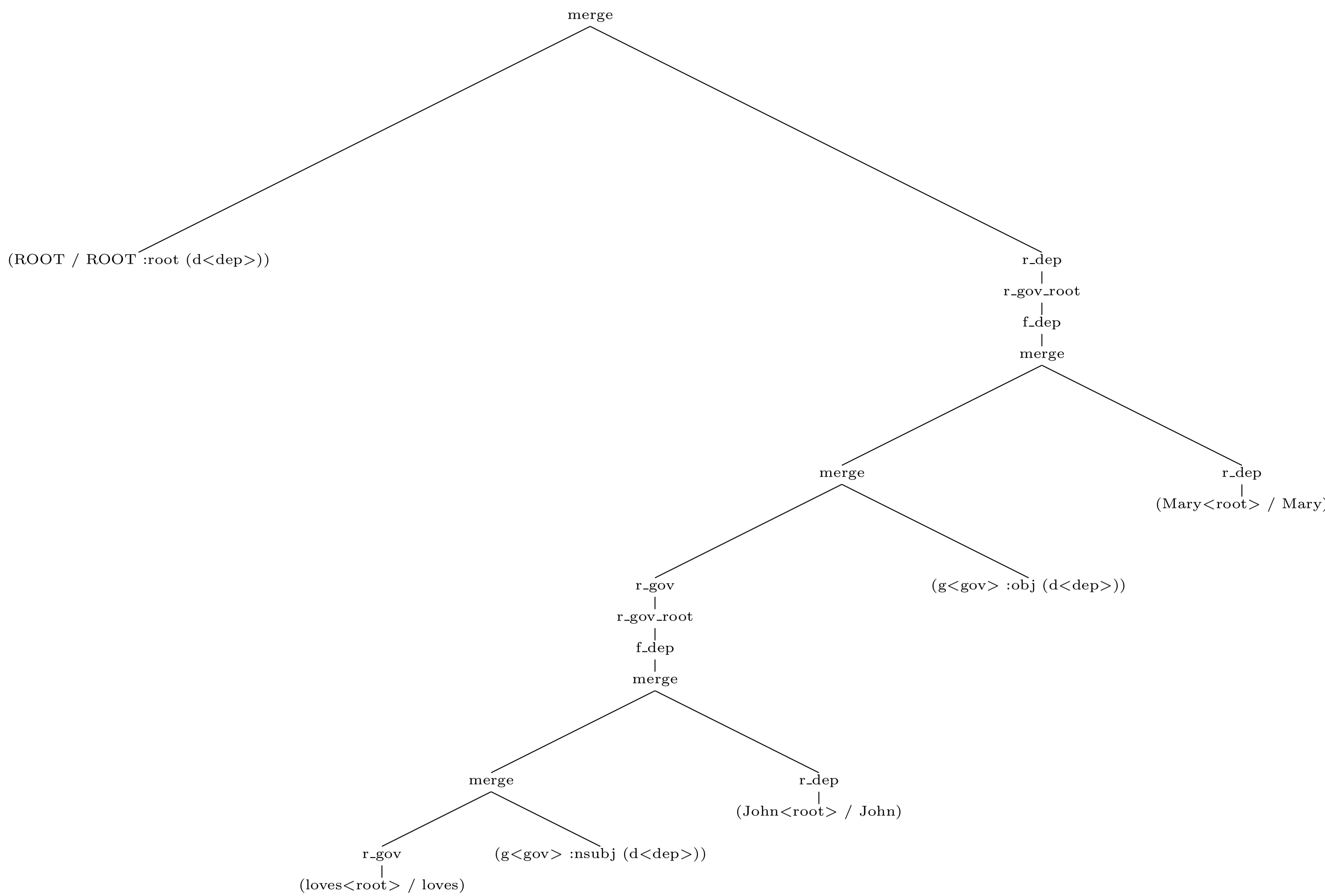


Figure 4.6: Alto's parsing of the sentence 'John loves Mary'.

RTG rule	homomorphisms
S $\rightarrow$ comb_subj(NP, VP)	s: $x1 \bullet x2$ g: f_subj $(x2 \vert\vert x1[subj])$
VP $\rightarrow$ sleep	s: sleep g: $G3$
NP $\rightarrow$ boy	s: the boy g: $G2$
VP $\rightarrow$ want_1(VP)	s: wants to $\bullet x1$ g: f_vcomp$(G1 \vert\vert x1[vcomp])$
VP $\rightarrow$ want_2(NP, VP)	s: wants $\bullet x1 \bullet to \bullet x2$ g: f_vcomp$(G1 \vert\vert F_{obj}(x1[obj] \vert\vert x2[subj \rightarrow obj, root \rightarrow vcomp]))$

Table 4.1: An s-graph grammar that illustrates complements (Koller (2015), page 6)

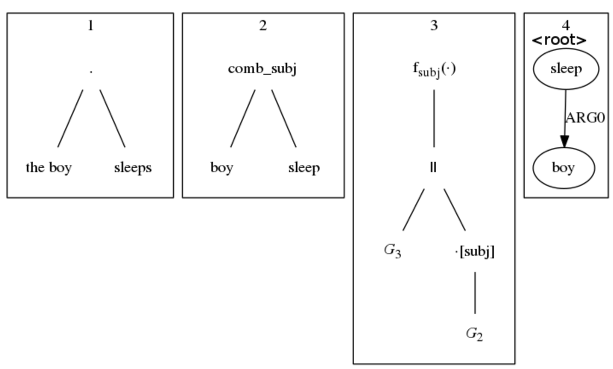


Figure 4.7: A derivation for “the boy sleeps”, using the grammar in Table 4.1 (Koller (2015), page 7)

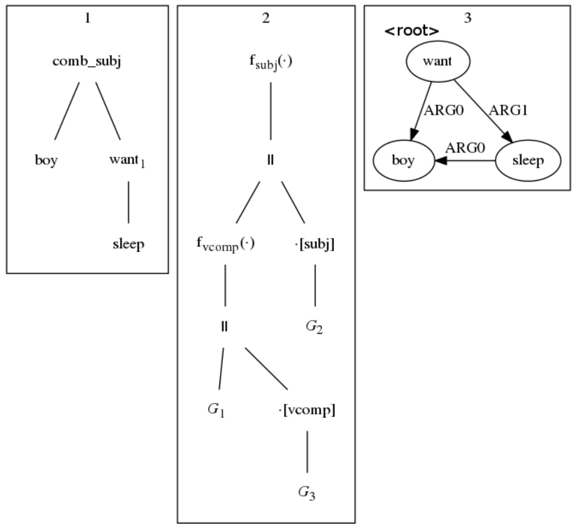


Figure 4.8: A derivation of “the boy wants to sleep”, using the grammar in Table 4.1 (Koller (2015), page 7)

RTG rule	homomorphisms
NP → nmod_rc(NP, RC)	s: $x1 \bullet x2$ g: $x1\|\|x2$
RC → rc(RP, VP)	s: $x1 \bullet x2$ g: $(f_{root}(x2\|\|x1[subj]))[subj \to root]$
RP → who	s: who g: <root>
VP → coord(VP, VP)	s: $x1 \bullet and \bullet x2$ g: $f_1, 2((< 1 >\leftarrow and < root >\to< 2 >)\|\|x1[1]\|\|x2[2])$
VP *rightarrow* sometimes(VP)	s: sometimes $\bullet x1$ g: (<root> $\leftarrow sometimes\|\|x1$
VP → snore	s: snores g: snore<root> $\to< subj >$

Table 4.2: An s-graph grammar featuring modification (Koller (2015), page 8)

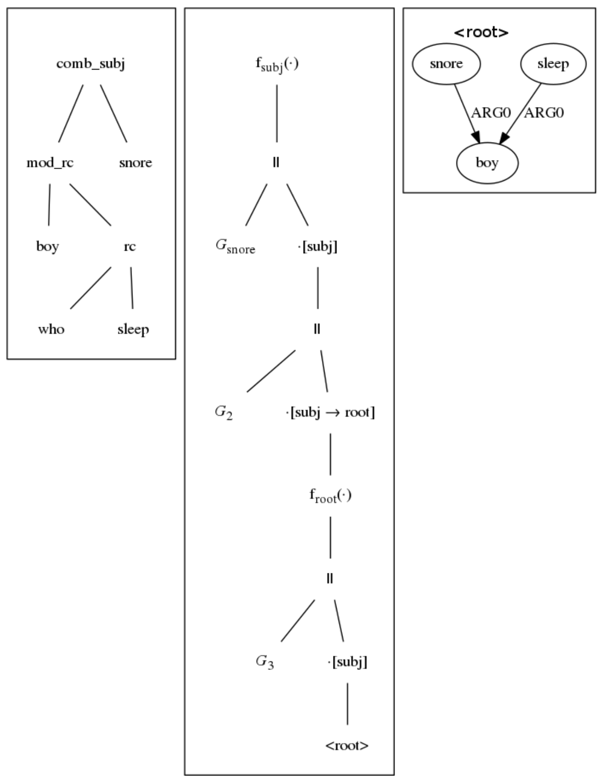


Figure 4.9: A derivation for “the boy who sleeps snores” using the grammar in Table 4.2 (Koller (2015), page 8)

beled node, with the subgraph for *sleep*. The relative pronoun's `root` node is renamed to `subj`. Merging happens without forgetting the `root` source and designating a new one. The grammar handles the s-graphs for adjuncts as they have a `root`-source which represents the place for inserting the modifiee. To summarize, complements and adjuncts are treated differently in the AMR-Bank. For combining a head with its complements, the roots of the complements will be renamed to their argument positions, then the argument names are forgotten, as the complements have been filled. Adjuncts, on the other hand, can be combined to their head with simply merging them, renaming and forgetting is not needed.

4.4 Mapping UD dependencies to 4lang graphs

In this section we present an IRTG which reimplements dep_to_4lang functionality, extends it to support UD, and some additional phenomena, such as the UD relation `case`. Most of the mapping simply upgrades the previously existing edges for Stanford Dependencies to Universal Dependencies, but several modifications and enhancements also had been made. These issues are discussed in Section 4.4.2. The grammar file is available on GitHub.[3]

There is a significant amount of overlap between Stanford Dependencies and Universal Dependencies, but they differ, for example, in the following: prepositional modifiers are replaced with `case`, and the passive dependencies, such as `nsubjpass` are now treated as subtypes. Table 4.3 presents the UD-conform version of 4lang rules.

4.4.1 Basic edge types

The majority of the rules simply replace the name of the dependency relation to a 4lang edge. As mentioned in Section 3.2.4, 0-edges denote attribution, predication and the IS_A relation. 1 and 2-edges connect arguments to a binary predicate. A simple rule like Figure 4.10 had already been explained in details in Section 4.2. Simply said, two subgraphs (the two Xs in the parentheses in the first line) are merged with an initial subgraph between the quotation marks. Rules for 2-edges are constructed in a very similar fashion, the dependency name is simply replaced by the label `2`. Figure 4.11 shows

[3] `https://github.com/kornai/4lang/blob/master/exp/alto/ud/en_ud_bi.irtg`

Dependency	Edge
advcl	$w_1 \xrightarrow{0} w_2$
advmod	
amod	
nmod	
nummod	
appos	$w_1 \underset{0}{\overset{0}{\rightleftharpoons}} w_2$
dislocated	
csubj	$w_1 \underset{0}{\overset{1}{\rightleftharpoons}} w_2$
nsubj	
ccomp	$w_1 \xrightarrow{2} w_2$
obj	
xcomp	

Table 4.3: UD-conform version of the mapping of Table 3.7.

the IRTG rule for the $w_1 \underset{0}{\overset{0}{\rightleftharpoons}} w_2$ edge type. $w_1 \underset{0}{\overset{1}{\rightleftharpoons}} w_2$ edges are implemented similarly.

4.4.2 The case relation

UD uses the relation case for three different purposes. As mentioned before, prepositional modifiers of SD were replaced with the relation case in UD. Thus, all these phenomena should have the same edge configuration in 4lang as prepositional modifiers.

The first configuration, case in conjunction with obl, provides an analysis for constructions like in Figure 4.12. The corresponding 4lang graph is represented in Figure 4.13. Figure 4.14 presents the IRTG rule. The RTG rule

```
X -> _obl_case(X, X, X)
```

specifies three subgraphs to be merged into the initial graph "(g<gov> :obl (d1<dep1> :case (d2<dep2>)))". Initially, the first subgraph's root node is renamed to gov, and merged into the central graph. Then the second subgraph's root is renamed to dep1, and merged. The third subgraph's

```
X -> _nummod(X, X)
[graph] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :nummod (d<dep>))"),
            r_dep(?2)
        )
    )
)
[fourlang] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :is_a (d<dep>))"),
            r_dep(?2)
        )
    )
)
```

Figure 4.10: An example for $w_1 \xrightarrow{0} w_2$ -edges from our grammar.

```
X -> _appos(X, X)
[graph] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :appos(d<dep>))"),
            r_dep(?2)
        )
    )
)
[fourlang] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :0 (d<dep> :0 g))"),
            r_dep(?2)
        )
    )
)
```

Figure 4.11: An example for $w_1 \overset{0}{\underset{0}{\rightleftharpoons}} w_2$ -edges from our grammar.

`root` is then renamed to `dep2` and also merged. After these steps, the labels `dep1` and `dep2` are forgotten, and as a final step, `gov` is renamed to `root`.

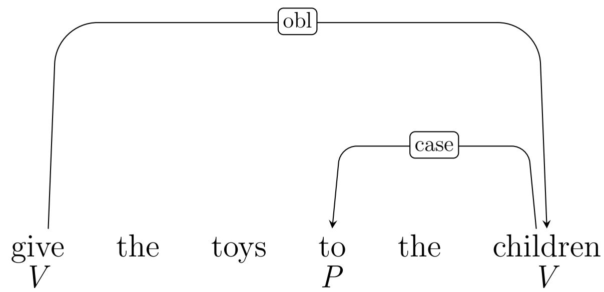


Figure 4.12: Treatment of `case` with the oblique nominal. Source: http://universaldependencies.org/u/dep/obl.html.

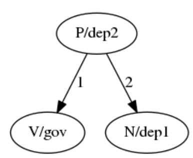


Figure 4.13: 4lang representations of `case` in conjunction with `obl`.

In the second configuration, `case` and `nmod` stand together, as in Figure 4.15. This requires the exact same treatment described above. Figure 4.16 provides an example of the third configuration, `nsubj` and `case`. Although its dependency graph is structured differently, the 4lang graph is the same as in the previous configurations. As the relation `case` never appears without one of these three other dependents, it is unnecessary to specify a rule for `case` alone. However, `obl`, `nmod` and `nsubj` appear without `case`. For such occurrences a rule is specified for each of them. The presence of `case` forces the rules discussed to be applied when necessary.

4.4.3 Some other issues

As in the previous dep_to_4lang mapping, we ignore technical relations such as `orphan`, `goeswith` and `reparandum`. `iobj` should be treated in the definitions. Our system doesn't currently handle purely grammatical functions,

```
        X -> _obl_case(X, X, X)
        [graph] r_gov_root(
    f_dep2(
        f_dep1(
            merge(
                merge(
                    merge(
                        r_gov(?1),
                        "(g<gov> :obl (d1<dep1> :case (d2<dep2>)))"
                    ), r_dep1(?2)
                ),
                r_dep2(?3)
            )
        )
    )
)
        [fourlang] r_gov_root(
    f_dep2(
        f_dep1(
            merge(
                merge(
                    merge(
                        r_gov(?1),
                        "(d2<dep2> :1 (g<gov>) :2 (d1<dep1>))"
                    ),
                    r_dep1(?2)
                ),
                r_dep2(?3)
            )
        )
    )
)
```

Figure 4.14: Rule for `case` with the oblique nominal.

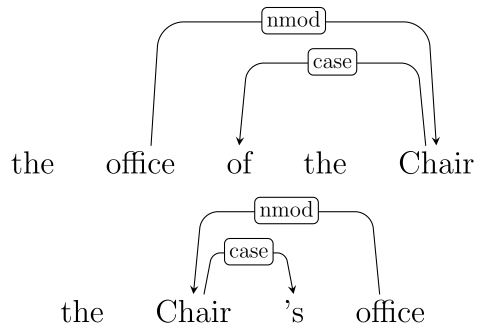


Figure 4.15: Treatment of `case` with the nominal modifier. Source: universaldependencies.org/u/dep/case.html.

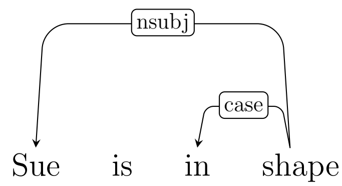


Figure 4.16: Treatment of `case` with the nominal subject. Source: http://universaldependencies.org/u/dep/case.html.

such as `vocative`, `expl`, `aux` etc. as well as the relations `compound` and `parataxis`.

Noun-noun compounds are notably difficult semantic phenomena. Kiparsky et al. (1982) notes, cited by Kornai (2018), that *ropeladder* means a *ladder made of rope*, manslaughter is *slaughter undergone by man*, and *testtube* denotes a *tube used for test*. The relation `parataxis` connects two syntactically independent clauses, so it is unclear whether we need to connect them via 4lang edges. Another use is shown in Figure 4.17, which presents an example where two standalone sentences are connected to a single sentence. In this case, as the words *world* and *CIA* refer to the same entity, it could be treated in 4lang graphs as the relation `appos`: $w_1 \overset{0}{\underset{0}{\rightleftharpoons}} w_2$. Figure 4.18 shows a reported speech example. 4lang graphs should assign this type to the edge type $w_1 \overset{1}{\underset{0}{\rightleftharpoons}} w_2$. Since the dependency structure does not differentiate between these two cases, our system cannot do so.

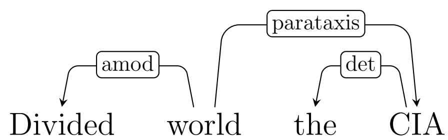


Figure 4.17: Dependency structure of the sentence 'Divided world the CIA'. Source: http://universaldependencies.org/u/dep/parataxis.html.

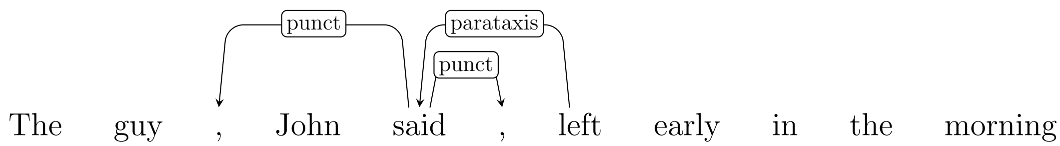


Figure 4.18: Dependency structure of the sentence 'The guy, John said, left early in the morning'. Source: http://universaldependencies.org/u/dep/parataxis.html

4.4.4 Subtypes for English

As mentioned before, SD's passive dependencies are handled as language-specific subtypes in UD. Table 4.4 presents the treatment of subtypes for English. `obl:npmod` is used in constructions like *middle-aged*. Syntactically, this relation is an argument of a verb, but regarding its function, it is a nominal modifier, so it is treated as $w_1 \xrightarrow{0} w_2$. When the nominal modifier and the oblique nominal specifies time, it is labeled as `nmod:tmod` and `obl:tmod`, respectively. We adopted the 4lang representation of SD's relation `tmod`: $w_1 \xleftarrow{1} \texttt{AT} \xrightarrow{2} w_2$. `nmod:poss` is present in constructions like *the cat's owner*. The 4lang representation of SD's relation `poss`, $w_2 \xleftarrow{1} \texttt{HAS} \xrightarrow{2} w_1$, had been adopted for such cases. Figure 4.19 gives an example IRTG rule for handling these types of configurations.

Dependency	Edge
obl:npmod	$w_1 \xrightarrow{0} w_2$
nmod:tmod obl:tmod	$w_1 \xleftarrow{1} \texttt{AT} \xrightarrow{2} w_2$
nmod:poss	$w_2 \xleftarrow{1} \texttt{HAS} \xrightarrow{2} w_1$

Table 4.4: Mapping of the language-specific subtypes of English to 4lang subgraphs

```
X -> _nummod(X, X)
[graph] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(g<gov> :nmod_tmod (d<dep>))"),
            r_dep(?2)
        )
    )
)
[fourlang] r_gov_root(
    f_dep(
        merge(
            merge(r_gov(?1), "(AT / AT :2 (d<dep>) :1 (g<gov>))"),
            r_dep(?2)
        )
    )
)
```

Figure 4.19: The treatment of `nmod_tmod`.

Chapter 5

Conclusion and future work

In this work an IRTG was presented which reimplements and extends dep_to_4lang functionality. We have mapped the previously existing edges for Stanford Dependencies to Universal Dependencies, also the following extensions and enhancements have been made: 1. UD-conformity, 2. handling of the UD relation `case`. In the UD formalism, `case` is used for three different purposes. Correspondingly, the rules `obl_case`, `nmod_case` and `nsubj_case` have been implemented.

As for ongoing work, we are currently working on dep_to_AMR, which maps UD dependencies to AMR graphs. We aim to develop a system which is capable of UD-4lang-AMR conversion. In the future we plan to implement a parallel interpretation of constituents, dependencies and semantics, as the patterns which maps constituency trees to dependency graphs in dependency parsers can be implemented as IRTGs. In such a system, surface-meaning correspondences could be encoded explicitly.

Bibliography

Andreas, J., Bauer, D., Chiang, D., Hermann, K. M., Jones, B., and Knight, K. (2013). A primer on graph processing with Bolinas. `https://www.isi.edu/licensed-sw/bolinas/bolinas_tutorial.pdf`. Online tutorial.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Barzdins, G. and Gosko, D. (2016). Riga at SemEval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *arXiv preprint arXiv:1604.01278*.

Björkelund, A., Falenska, A., Yu, X., and Kuhn, J. (2017). IMS at the CoNLL 2017 UD Shared Task: CRFs and Perceptrons Meet Neural Networks. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 89–97, Beijing, China. Coling 2010 Organizing Committee.

Brachman, R. J. (1977). What's in a concept: structural foundations for semantic networks. *International journal of man-machine studies*, 9(2):127–152.

Buchholz, S. and Marsi, E. (2006). CoNLL-X Shared Task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Buys, J. and Blunsom, P. (2017). Oxford at SemEval-2017 task 9: Neural AMR parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 914–919.

Cai, S. and Knight, K. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 748–752.

Camacho-Collados, J., Pilehvar, M. T., Collier, N., and Navigli, R. (2017). SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Chiang, D., Andreas, J., Bauer, D., Hermann, K. M., Jones, B., and Knight, K. (2013). Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 924–932.

Courcelle, B. (1993). Graph grammars, monadic second-order logic and the theory of graph minors. *Contemporary Mathematics*, 147:565–565.

Cui, H., Sun, R., Li, K., Kan, M.-Y., and Chua, T.-S. (2005). Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 400–407. ACM.

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

De Marneffe, M.-C. and Manning, C. D. (2008). Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

DeMarneffe, M.-C., MacCartney, W., and Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 449–454, Genoa, Italy.

Dozat, T., Qi, P., and Manning, C. D. (2017). Stanford's graph-based neural dependency parser at the CoNLL 2017 Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Farkas, R., Vincze, V., and Schmid, H. (2012). Dependency parsing of Hungarian: Baseline results and challenges. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 55–65. Association for Computational Linguistics.

Flanigan, J., Dyer, C., Smith, N. A., and Carbonell, J. (2016a). CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206.

Flanigan, J., Dyer, C., Smith, N. A., and Carbonell, J. (2016b). Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739.

Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Foland, W. and Martin, J. H. (2016). CU-NLP at SemEval-2016 task 8: AMR parsing using LSTM-based recurrent neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1197–1201.

Gaifman, C. (1965). Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Gontrum, J., Groschwitz, J., Koller, A., and Teichmann, C. (2017). Alto: Rapid prototyping for parsing and translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32.

Groschwitz, J., Koller, A., and Teichmann, C. (2015). Graph parsing with s-graph grammars. In *Proceedings of the 53rd ACL and 7th IJCNLP*, Beijing.

Gruzitis, N., Gosko, D., and Barzdins, G. (2017). RIGOTRIO at SemEval-2017 task 9: Combining machine learning and grammar engineering for AMR parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 924–928.

Gécseg, F. and Steinby, M. (1997). Tree languages. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages, Vol. 3.*, page 1–68.

Jurafsky, D. and Martin, J. H. (2018a). Computing with word senses. `https://web.stanford.edu/~jurafsky/slp3/`. 3rd ed. draft.

Jurafsky, D. and Martin, J. H. (2018b). Dependency parsing. `https://web.stanford.edu/~jurafsky/slp3/`. 3rd ed. draft.

Katz, J. and Fodor, J. A. (1963). The structure of a semantic theory. *Language*, 39:170–210.

Kiparsky, P. et al. (1982). From cyclic phonology to lexical phonology. *The structure of phonological representations*, 1:131–175.

Koller, A. (2015). Semantic construction with graph grammars. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, London.

Koller, A. and Kuhlmann, M. (2011). A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT)*, Dublin.

Kornai, A. (2018). *Semantics*. Springer Verlag.

Kornai, A., Ács, J., Makrai, M., Nemeskey, D. M., Pajkossy, K., and Recski, G. (2015). Competence in lexical semantics. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*, pages 165–175, Denver, Colorado. Association for Computational Linguistics.

Kornai, A. and Makrai, M. (2013). A 4lang fogalmi szótár. In Tanács, A. and Vincze, V., editors, *IX. Magyar Számitógépes Nyelvészeti Konferencia*, pages 62–70.

Lampouras, G. and Vlachos, A. (2017). Sheffield at SemEval-2017 task 9: Transition-based language generation from AMR. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 586–591.

Maruyama, H. (1990). Structural disambiguation with constraint propagation. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics*, pages 31–38. Association for Computational Linguistics.

May, J. (2016). SemEval-2016 Task 8: Meaning Representation Parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California. Association for Computational Linguistics.

May, J. and Priyadarshi, J. (2017). SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Mille, S., Carlini, R., Burga, A., and Wanner, L. (2017). FORGe at SemEval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 920–923.

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

Miller, T., Hempelmann, C., and Gurevych, I. (2017). SemEval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68.

Nguyen, K. and Nguyen, D. (2017). UIT-DANGNT-CLNLP at SemEval-2017 task 9: Building scientific concept fixing patterns for improving camr. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 909–913.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Nivre, J., Agić, Ž., Ahrenberg, L., Antonsen, L., Aranzabe, M. J., Asahara, M., Ateyah, L., Attia, M., Atutxa, A., Augustinus, L., Badmaeva, E., Ballesteros, M., Banerjee, E., Bank, S., Barbu Mititelu, V., Bauer, J., Bengoetxea, K., Bhat, R. A., Bick, E., Bobicev, V., Börstell, C., Bosco, C., Bouma, G., Bowman, S., Burchardt, A., Candito, M., Caron, G., Cebiroğlu Eryiğit, G., Celano, G. G. A., Cetin, S., Chalub, F., Choi, J., Cinková, S., Çöltekin, Ç., Connor, M., Davidson, E., de Marneffe, M.-C., de Paiva, V., Diaz de Ilarraza, A., Dirix, P., Dobrovoljc, K., Dozat, T., Droganova, K., Dwivedi, P., Eli, M., Elkahky, A., Erjavec, T., Farkas, R., Fernandez Alcalde, H., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors, M., Gerdes, K., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y., Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grūzītis, N., Guillaume, B., Habash, N., Hajič, J., Hajič jr., J., Hà My, L., Harris, K., Haug, D., Hladká, B., Hlaváčová, J., Hociung, F., Hohle, P., Ion, R., Irimia, E., Jelínek, T., Johannsen, A., Jørgensen, F., Kaşıkara, H., Kanayama, H., Kanerva, J., Kayadelen, T., Kettnerová, V., Kirchner, J., Kotsyba, N., Krek, S., Laippala, V., Lambertino, L., Lando, T., Lee, J., Lê Hong, P., Lenci, A., Lertpradit, S., Leung, H., Li, C. Y., Li, J., Li, K., Ljubešić, N., Loginova, O., Lyashevskaya, O., Lynn, T., Macketanz, V., Makazhanov, A., Mandl, M., Manning, C., Mărănduc, C., Mareček, D., Marheinecke, K., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Mendonça, G., Miekka, N., Missilä, A., Mititelu, C., Miyao, Y., Montemagni, S., More, A., Moreno Romero, L., Mori, S., Moskalevskyi, B., Muischnek, K., Müürisep, K., Nainwani, P., Nedoluzhko, A., Nešpore-Bērzkalne, G., Nguyen Thi, L., Nguyen Thi Minh, H., Nikolaev, V., Nurmi, H., Ojala, S., Osenova, P., Östling, R., Øvrelid, L., Pascual, E., Passarotti, M., Perez, C.-A., Perrier, G., Petrov, S., Piitulainen, J., Pitler, E., Plank, B., Popel, M., Pretkalniņa, L., Prokopidis, P., Puolakainen, T., Pyysalo, S., Rademaker, A., Ramasamy, L., Rama, T., Ravishankar, V., Real, L., Reddy, S., Rehm, G., Rinaldi, L., Rituma, L., Romanenko, M., Rosa, R., Rovati, D., Sagot, B., Saleh, S., Samardžić, T., Sanguinetti, M., Saulīte, B., Schuster, S., Seddah, D., Seeker, W., Seraji, M., Shen, M., Shimada, A., Sichinava, D., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Stella, A., Straka, M., Strnadová, J., Suhr, A., Sulubacak, U., Szántó, Z., Taji, D., Tanaka, T., Trosterud, T., Trukhina, A., Tsarfaty, R., Tyers, F., Uematsu, S., Urešová, Z., Uria, L., Uszkoreit, H., Vajjala, S., van Niekerk, D., van Noord, G., Varga, V., Villemonte de la Clergerie, E., Vincze, V., Wallin, L., Washington, J. N., Wirén, M., Wong, T.-s., Yu, Z., Žabokrtský, Z., Zeldes, A., Zeman, D., and Zhu, H. (2017). Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Nivre, J. and McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. *Proceedings of ACL-08: HLT*, pages 950–958.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Petrov, S., Das, D., and McDonald, R. (2011). A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Pourdamghani, N., Knight, K., and Hermjakob, U. (2016). Generating English from Abstract Meaning Representations. In *Proceedings of the 9th International Natural Language Generation conference*, pages 21–25.

Quillian, M. R. (1969). The teachable language comprehender. *Communications of the ACM*, 12:459–476.

Recski, G. (2018). Building concept definitions from explanatory dictionaries. *International Journal of Lexicography*.

Sakaguchi, K., Post, M., and Van Durme, B. (2014). Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11.

Schank, R. C. and Rieger III, C. J. (1974). Inference and the computer understanding of natural language. *Artificial Intelligence*, 5(4):373–412.

Shi, T., Wu, F. G., Chen, X., and Cheng, Y. (2017). Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada. Association for Computational Linguistics.

Sowa, J. (1976). Conceptual graphs for a data base interface. *Journal of Research and Development*.

Sowa, J. F. (1992). Conceptual graphs as a universal knowledge representation. *Computers & Mathematics with Applications*, 23(2-5):75–93.

van Noord, R. and Bos, J. (2017). The Meaning Factory at SemEval-2017 task 9: Producing AMRs with neural semantic parsing. *arXiv preprint arXiv:1704.02156*.

Vincze, V., Simkó, K. I., Szántó, Z., and Farkas, R. (2017). Universal Dependencies and morphology for Hungarian-and on the price of universality. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Vincze, V., Szauter, D., Almási, A., Móra, G., Alexin, Z., and Csirik, J. (2010). Hungarian dependency treebank. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and

Tapias, D., editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Wang, C., Pradhan, S., Pan, X., Ji, H., and Xue, N. (2016). CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178.

Wang, C., Xue, N., and Pradhan, S. (2015). A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks. *Representation and Understanding: Studies in Cognitive Science*, pages 35–82.

Wu, Y., Zhang, Q., Huang, X., and Wu, L. (2009). Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1533–1541. Association for Computational Linguistics.

Zeman, D. (2008). Reusable tagset conversion using tagset drivers. In *LREC*, volume 2008, pages 28–30.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., dePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, c., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R., and Li, J. (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2008). A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.

Ács, E. and Recski, G. (2018). Evaluation of Universal Dependency parsers for Hungarian. In Vincze, V., editor, *XIV. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2018)*, pages 295–304.