Combining Semantic Parsing Frameworks for Automated Knowledge Base Construction*

Martin Verrev[1,*,†]

[1]Tallinn University of Technology, Akadeemia Tee 15a, Tallinn, 12618, Estonia

Abstract

One of the crucial tasks for constructing a knowledge base for commonsense question answering is to automate extracting background knowledge from unstructured sources. While structured data sources like ConceptNet, Quasimodo, ATOMIC, and ontologies like WordNet provide facts and simple rules, they contain a lot of un-parsed English phrases and also lack most of the common everyday knowledge that everyone is expected to know. For both enriching these knowledge bases and asking questions using natural language, we need to perform semantic parsing of natural language phrases and sentences in a way that would be compatible with the structured data sources used. The contribution of this paper is combining AMR and UD notations to perform the said task - extracting the meaning from unstructured texts and representing it as knowledge graphs that are transformed into first-order logic formulae that can then be used for answering questions on the provided passage. The paper provides an example of such an experimental system, intended to be used with the Graph Knowledge (GK) logic engine.

Keywords

knowledge extraction, natural language understanding, commonsense reasoning, meaning representations

1. Introduction

Question-answering systems that rely purely on vector-based approaches struggle with answering questions based on commonsense knowledge, the most apparent shortcoming being a lack of transparency and interpretability while performing inference. The results obtained may be caused either by actual correlations or superficial cues that have been demonstrated to exist in commonsense reasoning benchmarks. On the other hand, humans are capable of solving tasks that need reasoning based on often incomplete information. That is possible due to having prior commonsense knowledge - facts about the everyday world everyone is expected to know. Natural language is the medium of capturing such knowledge. Such representations capture the meaning of a sentence as understood by a native speaker to the point where it can be used to train a system for automated reasoning. The paper describes one such system used for constructing such a knowledge base. In addition, the paper describes experiments conducted to measure the suitability of such a system.

6th Workshop on Advances In Argumentation In Artificial Intelligence (AI 2022), November 28 – December 2, 2022, University of Udine, Udine, Italy

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

martin.verrev@taltech.ee (M. Verrev)

0000-0003-4890-9283 (M. Verrev)

1.1. The Task of Semantic Parsing

Semantic parsing is a subfield of natural language understanding that maps natural-language utterances to detailed representations of their meaning. Such representations reflect the meaning of a sentence as understood by a native speaker to the point where it can be used for question answering or automated reasoning - one of such systems being *GK* [1] that uses JSON-LD-LOGIC – a notation that combines first-order logic with JSON that is compatible with both the JSON-LD standard and the TPTP format [2]. This enables the programmatic management of logical problems and provides a specification for the output format for the knowledge base so created. The format thus allows encoding knowledge from different sources in a unified format.

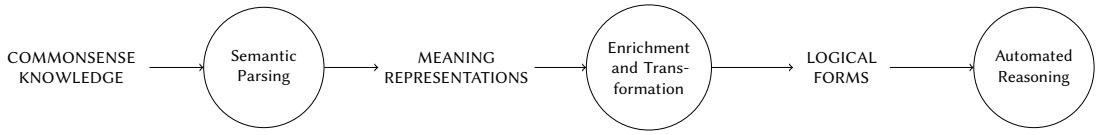

Figure 1: A broad overview of constructing a knowledge-base in the context of automated reasoning.

Given a passage, it is transformed into a graph representation via the task of semantic parsing. As in general current semantic parsers use the sentence as a unit of information, a sentence segmentation task is performed. A context is created during the enhancement and transformation phase to overcome that limitation. The intermediate semantic parse result can be enriched either with external information - e.g. Quasimodo [1], DBpedia [2], Wikidata [3] or any other structured datasource. In addition, translation rules are applied during this stage to provide uniform representations across multiple datasources. The representations are converted to first-order-logic formulate that can be saved and restored to be run on an automated reasoner for question answering.

2. Methodology

For constructing the experimental system, the following methodology was used. See our previous paper [3] for the details of steps 1-3.

1. During the preliminary phase, a review was conducted of existing semantic parsing frameworks to identify the key features and attributes of each framework.
2. Parsers were identified and chosen for each framework. Exclusion criteria were developed and applied to select a set of parsers for further evaluation.
3. For evaluating the parsers a corpus was defined and measures specified. Experiments were conducted to evaluate the parsers.
4. As a result the parsers and models were chosen: AMR for semantic representation and UD for syntactic and dependency parse trees. As both AMR and UD use sentences as a unit of meaning, a context was constructed that refers to the whole discourse.

[1] https://quasimodo.mpi-inf.mpg.de/
[2] https://github.com/dbpedia/
[3] https://www.wikidata.org/

5. For constructing the context, the classes of sentences were identified: conceptual, that do not describe specific situations but general concepts; factual which describe named entities and situational - describing concrete situations. A dictionary was constructed to map PropBank roles to predicates based on the sentence type.
6. Logical forms in JSON-LD-LOGIC were constructed based on the passage provided. An experimental system was deployed.

3. Identification of Technologies

During the initial phase, a survey and a set of experiments were conducted by the author to identify the features of common semantic parsing frameworks: see [3] for additional details. The findings indicated that a hybrid system should be constructed that uses a set of parsers: optionally UDS parser for pre-processing the input passage for simplifying the sentence structure; AMR for unanchored representations to generalize the syntactic layer; and UD to constrain and specify the generalized graphs for the passage provided.

3.1. Identification of Frameworks

The following frameworks were identified during the preliminary phase of the study:

AMR is a notation based on propositional logic and neo-Davidsonian semantics. AMR encodes the semantics of a sentence into a directed graph. Nodes in the graph represent semantic terms in the sentence and the edges identify the semantic relations between nodes. Concepts can be either English verbs, PropBank framesets, or specific keywords. AMR includes frame arguments based on Propbank conventions, general semantic relations; and relations for quantities, date-entities, and lists. It lacks universal quantifiers and support for inflectional morphology. Annotations do not link between concept and original span – thus input has to be aligned first [4].

Universal Conceptual Cognitive Annotation (UCCA) is a language-agnostic annotation scheme based on Basic Linguistic Theory where natural language utterances are converted to a graph containing purely semantic categories and structure where the base layer is a scene describing movement, action, or event [5]. The focus of UCCA has been ease of annotation and the foundational layer can be extended by extra domain or language specific layers.

Universal Dependencies (UD) is a framework for consistent annotation of grammar. It solves the problem of different corpora having different tagsets and annotation schemes [6] by providing a universal scheme and category set: suitable for parser development, cross-lingual support, and language parsing, allowing language-based extensions if necessary [7]. It combines Stanford Dependencies, Google universal part-of-speech tags, and Interset interlingua[4] for morphosyntactic tagsets - e.g. encoding additional morphological information to the syntactic form [8]. UD corpora consist of over 200 treebanks in over 100 languages.

Elementary Dependency Structures (EDS) present an approach to Minimal Recursion Semantics (MRS) banking. MRS is an approach where each input item in a corpus is paired with elementary predicates - meaning single relation with its associated arguments - followed by

[4] https://ufal.mff.cuni.cz/interset

manual disambiguation of quantifiers [9]. The semantic form is based on the notion of semantic discriminants - local dependencies extracted from full-fledged semantic representation [10].

Prague Tectogrammatical Graphs (PDT) provides annotations in English and Czech languages. The English sentences are from a complete English Web Text Treebank [5] and a parallel Czech corpus morphologically annotated and parsed into surface-syntax dependency trees in the Prague Dependency Treebank (PDT) 2.0 annotation style based on the same sentences. Noteworthy is the annotations having multiple layers - an analytical (surface-syntax) layer consisting of dependency structures, semantic labels, argument structure, and ellipsis resolution; and a manually constructed deep-syntax tectogrammatical layer on top of that [11].

Discourse Representation Structures (DRS) is a semantic formalism based on Discourse Representation Theory. In contrast to ordinary treebanks, the units of annotation in the corpus are texts rather than isolated sentences. [12]. Basic DRSs consist of discourse referents like x representing entities and discourse conditions like $man(x)$ representing information about discourse referents. [13] The corpus is based on Groningen Meaning Bank that annotates English texts with formal meaning representations rooted in Combinatory Categorial Grammar[14].

Universal Decompositional Semantics (UDS) framework is different from other formalisms because it decodes the meaning in a feature-based scheme — using continuous scales rather than categorical labels. The meaning is captured as a node- and edge-level attribute in a single semantic graph having the structure deterministically extracted from Universal Dependencies. UDS treats parsing as a sequence-to-graph problem - the graph nodes created are based on input sequence, and edges are dynamically added during generation [15].

3.1.1. Choosing the Frameworks

We follow the classification of Kollar [16] for generated dependency graphs based on the relation of graph elements to surface tokens. For bi-lexical dependency graphs (*type 0*), the graph nodes correspond to surface lexical units. Anchored semantic graphs (*type 1*) are characterized by relaxing the correspondence relations between nodes and tokens while still explicitly annotating the correspondence between nodes and parts of the sentence. For unanchored dependency graphs (*type 2*), the correspondence between the nodes and tokens is not explicitly annotated.

Table 1
Summary of Semantic Representation Frameworks

Name	Unit of Annotation	Flavor	Format	Primary Languages
AMR	sentence	unanchored	Penman	English (+Chinese)
UCCA	sentence	anchored	XML	English (+4 others)
UD	sentence	bi-lexical	CoNLL-U	multilingual
EDS	sentence	anchored	DAG	English
PTG	sentence	anchored	DAG	English, Czech
DRS	passage	anchored	nested boxes	English
UDS	sentence	anchored	Predicates + UD (CoNLL-U)	English

[5] https://catalog.ldc.upenn.edu/LDC2015T13

A summary of semantic parsing frameworks is presented in Table 2 with chosen frameworks highlighted. AMR was chosen as an un-anchored representation, providing the highest level of abstraction from surface tokens. From anchored frameworks that provide a level of abstraction from the surface form but still retain portions of it, UCCA supports the single sentence as a unit of information, and DRS for supporting a passage consisting of multiple sentences was chosen. In addition, UDS was added to the test battery due to deterministically including UD bi-lexical annotations in addition to extracting predicates from the input sentence.

3.2. Identification of Parsers

A representative sample of parsers was identified for conducting the experiments that are summarized in Table 2 with chosen parsers highlighted. The following inclusion criteria were applied: *availability* - only publicly available parsers were included; *development activity* - how active and up-to-date the development of said parser is and *accuracy* - the parsing accuracy of said tool based on literature;

During the initial review, 16 parsers were identified. For conducting the experiments the following exclusion criteria were applied: *interoperability* As the parsers will be integrated into an existing workflow, those not using Python were discarded; *freshness* The parsers where the last commit was made after January 2020 were discarded; *lightness* - given a choice between otherwise matching and equally performant parsers, the more lightweight one was chosen.

Table 2
Summary of Semantic Parsers

Notation	Parser	Platform	Stars	Forks	Commits	Last Commit
AMR	JAMR[6]	scala	192	50	825	March 2019
AMR	Transition AMR parser[7]	python	144	35	1838	November 2022
AMR	**amrlib**[8]	python	146	22	166	March 2022
UCCA	UCCA parser[9]	python	18	7	6	June 2019
UCCA	**TUPA**[10]	python	73	22	2135	December 2020
UD	UDepLambda[11]	java	85	22	225	July 2018
UD	uuparser[12]	python	77	26	125	October 2020
UD	stanza[13]	python	6400	830	3146	September 2022
EDS	Pydelphin[14]	python	68	24	1043	October 2022
EDS	HRG Parser[15]	python,java	9	0	4	October 2018
PTG	Perin[16]	python	41	5	36	Oct 04 2021
DRS	**TreeDRSparsing**[17]	python	5	2	59	March 2020
DRS	EncDecDRSParsing[18]	python	36	11	15	August 2019
UDS	**Predpatt**[19]	python	110	23	59	February 2021
UDS	MISO[20]	python	7	1	1019	September 2021

[6] https://github.com/jflanigan/jamr
[7] https://github.com/IBM/transition-amr-parser

3.3. Evaluation of Parsing Frameworks

Initial experiments were conducted to measure the performance of said parsers. For this, a test corpus was constructed to evaluate the robustness of chosen parsers. The sources for sentences were: CommonsenseQA[21] (312 sentences), Geoquery Data [22] (5 sentences) and synthetic examples capturing the essential linguistic features for translating the text to logical form (32 sentences). These features are: handling simple facts; extraction of predicates from traditional set theory; extraction of universal and existential quantifiers; handling of negation; handling logical connectives: conjunction, disjunction, and implication; handling of equality; handling of multiple variables and identification and extraction of questions. After initial experiments, the baseline test corpus was pruned, and as a result, a minimal corpus - consisting of 58 sentences (594 tokens) remained. The corpus and output data can be found at https://cs.taltech.ee/research/commonsense/.

Due to a variety of annotation schemes and not having the 'correct' gold-label annotations, two evaluation measures were defined: granularity and robustness. *Granularity* was defined as the ratio of tokens in the input sentence compared to the number of semantic attributes captured, averaged over the whole corpus. To evaluate *robustness*, human evaluation was conducted by the author.

Each result was manually graded on a scale of 0..1. If the information captured was deemed complete and accurate, it was graded 1. If a portion of information was missing, it was graded 0.5. If arbitrary or non-relevant information was added – as it is hard to detect such errors in the knowledge base – the score was lowered by 0.3 points. If essential information was not present or the parse failed, the grade was 0. For each framework, the grades were averaged over the whole corpus.

The results of an experiment are documented in Table 3. The robustness metric suggests that all frameworks chosen performed similarly well - though none achieved flawless results - providing incomplete parsing results. The granularity metric indicates additional information generated by the parsing process - a lower value indicates additional information being added - temporal variables in the case of UDS and named entities for AMR. The usefulness of such information is dependent on the context of the results being used.

[8] https://github.com/bjascob/amrlib
[9] https://github.com/SUDA-LA/ucca-parser
[10] https://github.com/danielhers/tupa
[11] https://github.com/sivareddyg/UDepLambda
[12] https://github.com/UppsalaNLP/uuparser
[13] https://github.com/stanfordnlp/stanza
[14] https://github.com/delph-in/pydelphin
[15] https://github.com/draplater/hrg-parser
[16] https://github.com/ufal/perin
[17] https://github.com/LeonCrashCode/TreeDRSparsing
[18] https://github.com/EdinburghNLP/EncDecDRSparsing
[19] https://github.com/hltcoe/PredPatt
[20] https://github.com/esteng/miso_uds
[21] https://huggingface.co/datasets/commonsense_qa
[22] https://www.cs.utexas.edu/users/ml/nldata/geoquery.html

Table 3
Comparative summary of robustness and granularity values for semantic parsing frameworks

Framework	Parser	Model	Robustness	Granularity
AMR	AMRLib	Parse T5 v0.2.0	0.94	0.14
UCCA	TUPA	ucca-bilstm-1.3.1	0.96	0.19
UDS	PredPatt	UDS 1.0	0.92	0.08
DRS	TreeDrsParser	*built-in*	0.92	0.02

4. Combining the Parsers

Due to the ambiguous nature of natural language, no parser alone was ideally suitable for the task. On the other hand, all the parsers chosen performed well in different aspects of capturing the meaning. The robustness of UDS is suitable for preprocessing the input - simplifying the structure of the sentence, and splitting it into key components. At the same time, the splitting boundary for some input sentences seemed arbitrary. On the other hand, AMR is suitable for explicit negation extraction and question detection. Additionally, the Penman output format is suitable for further post-processing due to its rigid yet flexible structure. On the other hand - adding additional hand annotations is not a viable option, and for the current task, we must rely on publicly available annotations. UCCA performed the best on the correctness scale but did not explicitly state negation and entity recognition. At the same time, due to annotation tooling, it is possible to implement the required layers if deemed necessary.

To evaluate using the parsers in parallel a hybrid system for representing knowledge in first-order logic using said parsers in an ensemble. AMR was chosen for unanchored representations to generalize the syntactic layer, and UD to constrain and specify the generalized graphs.

Amrlib was chosen with Parse T5 0.20.0 model[23]. **Stanza**, a Python NLP toolkit was used for text processing tasks: text tokenization, named entity recognition, part-of-speech tagging, constituency parsing, dependency parsing, and lemmatization.

4.1. Conversion to Logical Form

We will follow the method outlined by Hatzilygeroudis [17] to construct the logical form from the sentence: find predicates and specify their arguments; construct corresponding atoms; divide atoms on the same level into groups; specify connectives between atoms of each group and construct corresponding formulas; divide formulas and/or any of the remaining atoms of the same level into groups; specify connectives between elements of each group; specify quantifiers for the variables and finally construct the final FOL formula.

The pipeline is the following:

1. Given an input passage tokenize it into sentences.
2. For each sentence extract its semantic representation via AMR and syntactic structure via UD

[23] https://github.com/bjascob/amrlib-models/releases/download/model_parse_t5-v0_2_0/model_parse_t5-v0_2_0.tar.gz

3. Perform sentence classification. If no sentence type can be determined, ignore it and store it for further analysis.
4. Create a local context for the sentence
5. Combine the contexts for individual sentences.
6. Use AMR representation with global context to construct the logical clauses.
7. *(*Not implemented) Given a question parse the question and answer it on generated clauses. If given, it is assumed the questions are provided at the end of the passage. The naive approach assumes that the question ends with a question mark or starts with one of the seven question words in English.

4.2. Limitations of the Pure AMR Based Approach

AMR parse graphs are represented in Penman notation with well-formed results that can easily be parsed to first-order logic [18] as demonstrated by *amr2fol* [24] project. Still, AMR has several constraints when parsing natural language and fails unexpectedly. This is not due to the limitations of AMR itself but more due to the ambiguous nature of natural language. Given the general domain, it is not possible to craft the rules or train the model that covers the full scope of the language.

Two types of inconsistencies were recognized when conducting the preliminary experiments: informational where incorrect or wrong type information is inferred. Generally, this happened with named entity recognition; and structural, where the structure of the parse tree does not reflect its true meaning, e.g 'black and white' interpreted as a single property in a listing of colors.

In addition, it was recognized that AMR does not support scenes (in contrast to UCCA and DRS). Thus, a manual context creation for coreference resolution similar to the approach described for DocAMR representation a [19] was implemented.

4.3. Passage Context

To overcome the limitation of scene support, the context was created for a sequence of sentences similar to [20]. For describing situations it was assumed that the sequence of sentences follows the sequence of events. Additionally, the context keeps track of concepts and named entities: having "John" occurring in a passage and later "he" we can assume that "he" refers to "John" when answering questions. A sample of context object is presented in Figure 2.

4.4. Translation Rules

A minimal ontology was constructed to provide interoperability across several contexts. Based on sentence class, mappings were generated for AMR core attributes. The following rules were applied

1. For verbs, lemmatized version was used as a predicate: 'walked' → 'walk-01' → 'walk'

[24] https://github.com/papagandalf/amr2fol

Figure 2: A sample context object for the sentence "Brutus stabs Caesar with a knife.".

```
{ 'amr_root': {'lemma': 'stab', 'upos': 'VERB'},
'entities': [ {'text': 'Brutus', 'type': 'PERSON'},
              {'text': 'Caesar', 'type': 'PERSON'}],
'IDX': 0,
'question': False,
'type': 'sit',
'ud_root': {'lemma': 'stab', 'upos': 'VERB'}}
```

2. For custom AMR predicates, custom mappings were created e.g: 'have-org-role-91' → 'role' or ':poss' → 'belongsTo'
3. Sentence class-based semantic role mappings were applied, e.g. 'ARG0' denotes typically *agent* role in general but *instrument* in other cases. For this, a sentence classification used during the context creation step was used.

4.5. Sentence Classification

Based on the knowledge represented therein, we label the sentence into one of three classes:

(a) *Conceptual statements* does not describe a specific situation and are not dependent on uncommon circumstances. Typically, they describe concepts or relations between concepts. *Example:* John is a man.
(b) *Fact statements* describe named entities and are also not dependent on uncommon circumstances. *Example:* Tallinn is the capital of Estonia.
(c) *Situational statements* describe a concrete situation and events happening within this situation. *Example:* Brutus stabbed Caesar with a knife.

To evaluate sentence classification accuracy and optimize the heuristic parameters for classification accuracy, an experiment was constructed. A dataset was chosen for each sentence type. Heuristics based on UD were constructed to classify the sentences. A classifier was constructed and *F1* score was calculated to finetune the parameters of heuristics and verify the accuracy of the classifier.

Table 4
Sentence classification evaluation results

Sentence Type	Dataset	Snt. count	F1 Score
Conceptual	OpenbookQA	100	0.91
Fact	DBPedia 1.4	100	0.18
Situational	SocialIQa	110	0.95

The following sources were chosen for test data: OpenbookQA [25] for conceptual sentences, DBPedia [26] for factual sentences and Social IQa dataset [27] for situational sentences.

[25] https://ai2-public-datasets.s3.amazonaws.com/open-book-qa/OpenBookQA-V1-Sep2018.zip
[26] https://huggingface.co/datasets/p
[27] https://leaderboard.allenai.org/socialiqa/submissions/get-started

The accuracy for the classification of factual sentences is much lower than others - due to classifying the sentences as situational.

5. Conclusions

A preliminary study was conducted to identify the most common semantic parsing frameworks. A set of parsers were chosen, and a corpus was created to evaluate the suitability of said frameworks for automated knowledge base construction.

Based on the results of said experiments, the technologies were chosen, and an experimental system was constructed to extract logical representations and perform text-to-logic conversion – combining AMR unanchored parse trees with bi-lexical UD annotations. The experimental system is found at https://cs.taltech.ee/research/commonsense

The system built has currently several limitations

- *Compound sentence segmentation.* Given compound sentences they are not split before parsing, resulting in increased complexity of generated logical forms.
- *Context order.* It is assumed the events in the passage occur in the order of the sequence of sentences provided.
- *Question detection and scope.* Question parsing is not implemented in the current version of the system.

Nonetheless, combining AMR with UD has several benefits. It allows us to: (a) improve, prune and specify the generated parse trees and (b) help to generate and specify the context: identifying and storing the references to named entities and classifying the sentence type.

References

[1] T. Tammet, D. Draheim, P. Järv, GK: Implementing Full First Order Default Logic for Commonsense Reasoning (System Description), in: J. Blanchette, L. Kovács, D. Pattinson (Eds.), Automated Reasoning, Springer International Publishing, Cham, 2022, pp. 300–309.

[2] T. Tammet, G. Sutcliffe, Combining JSON-LD with First Order Logic, in: 2021 IEEE 15th International Conference on Semantic Computing (ICSC), IEEE, 2021, pp. 256–261.

[3] M. Verrev, in: Evaluation of Semantic Parsing Frameworks for Automated Knowledge Base Construction, To apperar in ISDA2022. Lecture Notes in Networks and Systems, Springer, 2022.

[4] W.-T. Chen, M. Palmer, Unsupervised AMR-Dependency Parse Alignment, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, 2017, pp. 558–567.

[5] O. Abend, A. Rappoport, UCCA: A Semantics-based Grammatical Annotation Scheme., in: IWCS, volume 13, 2013, pp. 1–12.

[6] D. Zeman, Reusable Tagset Conversion Using Tagset Drivers, in: Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), 2008.

[7] K. Haverinen, J. Nyblom, T. Viljanen, V. Laippala, S. Kohonen, A. Missilä, S. Ojala, T. Salakoski, F. Ginter, Building the essential resources for Finnish: the Turku Dependency Treebank, Language Resources and Evaluation (2014).

[8] J. Nivre, M.-C. de Marneffe, F. Ginter, J. Hajič, C. D. Manning, S. Pyysalo, S. Schuster, F. Tyers, D. Zeman, Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection, in: Proceedings of the 12th Language Resources and Evaluation Conference, European Language Resources Association, 2020, pp. 4034–4043. URL: https://www.aclweb.org/anthology/2020.lrec-1.497.

[9] A. Copestake, D. Flickinger, C. Pollard, I. A. Sag, Minimal Recursion Semantics: An introduction, Research on Language and computation 3 (2005) 281–332.

[10] S. Oepen, J. T. Lønning, Discriminant-based MRS banking, in: LREC, 2006, pp. 1250–1255.

[11] J. Hajic, E. Hajicová, J. Panevová, P. Sgall, O. Bojar, S. Cinková, E. Fucíková, M. Mikulová, P. Pajas, J. Popelka, et al., Announcing Prague Czech-English Dependency Treebank 2.0, in: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), 2012, pp. 3153–3160.

[12] V. Basile, J. Bos, K. Evang, N. Venhuizen, Developing A Large Semantically Annotated Corpus, in: LREC 2012, Eighth International Conference on Language Resources and Evaluation, 2012.

[13] Y. Liu, W. Che, B. Zheng, B. Qin, T. Liu, An AMR Aligner Tuned by Transition-based Parser, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2422–2430.

[14] J. Bos, V. Basile, K. Evang, N. J. Venhuizen, J. Bjerva, The Groningen Meaning Bank, in: Handbook Of Linguistic Annotation, Springer, 2017, pp. 463–496.

[15] A. S. White, D. Reisinger, K. Sakaguchi, T. Vieira, S. Zhang, R. Rudinger, K. Rawlins, B. Van Durme, Universal Decompositional Semantics On Universal Dependencies, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1713–1723.

[16] T. Kollar, D. Berry, L. Stuart, K. Owczarzak, T. Chung, L. Mathias, M. Kayser, B. Snow, S. Matsoukas, The Alexa Meaning Representation Language, in: NAACL-HLT (3), 2018, pp. 177–184.

[17] I. Hatzilygeroudis, Teaching NL to FOL and FOL to CF Conversions., in: FLAIRS Conference, 2007, pp. 309–314.

[18] J. Bos, Squib: Expressive Power of Abstract Meaning Representations, Computational Linguistics 42 (2016) 527–535. doi:10.1162/COLI_a_00257.

[19] T. Naseem, A. Blodgett, S. Kumaravel, T. O'Gorman, Y.-S. Lee, J. Flanigan, R. F. Astudillo, R. Florian, S. Roukos, N. Schneider, DocAMR: Multi-Sentence AMR Representation and Evaluation, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021.

[20] J. Bos, Separating Argument Structure From Logical Structure In AMR, arXiv preprint arXiv:1908.01355 (2019).