Abstract Meaning Representation: A State-of-the-Art Review

Nasim Tohidi[1], Chitra Dadkhah[2]*, Alexander Gelbukh[3]

[1] Artificial Engineering Department, Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran (e-mail: n.tohidi@email.kntu.ac.ir).

[2] Artificial Engineering Department, Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran (e-mail: dadkhah@kntu.ac.ir).

[3] CIC, Instituto Politécnico Nacional, Mexico City, Mexico (e-mail: gelbukh@cic.ipn.mx).

*Corresponding Author

Received 2 Oct. 2022 Received in revised form 24 Jan. 2023 Accepted 1 May 2023

Type of Article: Review paper

***Abstract*-The application of Abstract Meaning Representation (AMR) is widely increasing as a principal form of structured sentence semantics, and it is considered as a turning point for Natural Language Processing (NLP) research. AMRs are rooted and labeled graphs, which capture semantics on sentence level and abstract away from Morpho-Syntactic properties. The nodes of the graph represent meaning concepts, and the edge labels show relationships between them. In this paper, we give a brief review about the existing approaches of generating text from AMR and parsing input text to produce AMR by studying various research from 2013 to 2022. Besides, we explain how the researchers have been used AMR for prevalent NLP tasks. Afterwards, we describe the existing datasets and evaluation metrics, which can be used in this regard. Finally, we discuss some basic features and challenges of AMR.**

***Keywords*: Abstract Meaning Representation, Generation, Parsing, Natural Language Processing, Text**

I. INTRODUCTION

Every human can answer the below question easily, in a given context, but it is very complicated for machines to analyze this question in natural language:

Who did what to whom?

Determining the true meaning of human natural language by machine has always been one of the main goals of Natural Language Processing researchers. Using Machine Learning (ML) methods for training the data, which are the common approach in this field, the challenge has changed to have a meaningful representation that is computationally friendly and can be used to annotate a large amount of data in multiple natural languages, consistently.

It is about two decades or more, that NLP analysis relied completely on syntactic Treebanks Corpora to make machines to get the meaning of human natural languages. When Penn Treebank project [1] released the first large-scale Treebank, even more syntactic Treebanks have been proposed for a wide range of languages. Then, they have been used to build principal NLP systems, such as Part-Of-Speech (POS) taggers, Question Answering (QA), Machine Translation (MT) and Text Summarization (TS) systems [2-4].

By passing from the syntactic structure analysis to semantics, scientists found that statistical parsers are not well suited for meaning representation production. In semantic analysis, complicated structures, which are very difficult to capture by parse tree structures and their limitations have often been encountered. For instance, in semantic network structure, nodes are often equivalent to the argument of more than one predicate. So, it can be useful for finding semantically less important words, hence, leaving nodes that do not add any further meaning to the final result, unattached. To solve the problems posed by this limitation and do a direct semantic analysis of all sentences, recent research have shifted to parsing with graph-structured representations. Because, syntactic Treebanks had been vital for enhancing the performance of syntactic parsers, emerge techniques with semantic parsing using Sembanks, which are sets of English sentences paired with their related semantic representations [5].

For example, the semantic structure of the sentence *The dog treed the cat* is considerably complicated, and translating it to other natural languages may not be easy. Because the verb *treed* is an example of skewing between semantics and grammar. *Tree* is applied as a verb in this sentence and the event has happened caused to go up. In other words, the sentence means the dog chased the cat, thus, the cat went up into a tree or the dog caused the cat to go up into a tree [6].

Usually, semantic parsing involves domain dependence. Some application domains are: The Air Travel Information Service (ATIS), the Robocup Coach Language (Clang), and Database Query Application (GeoQuery). Although, we need definitely large semantic banks for broad coverage in NLP, various related projects have been launched like the Groningen Meaning Bank (GMB) [7], the Semantic Treebank (ST) [8], UCCA [9], the Prague Dependency Bank [10] and UNL [11].

Afterwards, Banarescu et al. [12] tried to annotate the logical meaning of sentences in Abstract Meaning Representation (AMR), which constituted semantic roles, questions, co-reference, modality, negation, and linguistic phenomena. Therefore, by producing a notable corpus and a correctable logical semantic input format, the AMR creators hope to be able to encourage important advances in Statistical Machine Translation (SMT), Natural Language Generation (NLG), and Statistical Natural Language Understanding (SNLU).

The existence of AMR parsers and their quality performance has encouraged many researchers to work on integrating the whole sentence meaning into NLP applications. In this regard, Tohidi and Dadkhah [13] provided a short review of AMR applications in downstream tasks. To the best of our knowledge, the distribution of the different applications of AMR in various NLP tasks is as follows: 31% for Machine Comprehension, 18% for each Text Summarization and Question Answering, 13% for Entity Linking and Linked Data, 9% for each Machine Translation and Information Retrieval and 2% for other NLP tasks.

In this paper, we investigated AMR model and reviewed the existing methods for parsing natural language to AMR and generating it from AMR. In this paper we categorized Parsing methods into five categories: 1) Grammar-based, 2) Graph-based, 3) Transition-based, 4) Sequence-to-sequence-based and 5) Conversion-based and Generating methods into six categories: 1) Tree-transducer-based, 2) Graph-grammar-based, 3) Graph-based, 4) Sequence-to-sequence-based, 5) Rule-based and 6) Transition-based. Besides we discussed evaluation metrics and the datasets used in related works and the main applications of AMR. In addition, we explained some common challenges of working with AMR structure.

We noticed that the distribution of the different languages that have been used in previous AMR related works is as follows: 82% for English, 7% for Chinese, 2% for each Japanese and Czech languages, 1% for each Persian, German, Spanish, Portuguese, French, Korean, etc. based on the research. As expected, most of the research have been done on English language, and more effort needed for other languages. One of the reasons for less research in non-English languages could be its structure. AMR designed specifically for English, and it may not be easy to use the whole format in some languages.

The rest of this paper structured as follows: Section 2 investigates AMR briefly. Section 3 gives an overview of AMR parsing and its related works. Section 4 studies generation text from AMR and its related works. Section 5 discusses common evaluation methods in this area, some of the AMR challenges, and the basic features about AMRs. Finally, Section 6 provides the conclusion.

II. ABSTRACT MEANING REPRESENTATION

In this section the basic and the improved form of AMR are explained respectively.

A. Basic AMR

There are two types of meaning representations: symbolic representations and distributed representations. AMR is an example of the first one and represents the semantic of an English utterance as a set of relations between predicates and entities, which packaged in a graph-based structure. In the graph, nodes are equivalent to variables that represent individuals, entities and predicates of the utterance. AMR uses a neo-Davidsonian view of predicate meanings and treats predicates as atomic individuals.

First, it would be helpful to look at some examples of AMR to get more familiar with its structure. Fig. 1 represents AMRs in the form of trees for the following two sentences:

1) The children moaned.
2) Ribble handed out envelopes to the children.

```
(e / moan-01
    :ARG0 (x / child))
```

```
(e / give-01
    :ARG0 (x / person: named "Ribble")
    :ARG1 (y / envelope)
    :ARG2 (z / child))
```

Figure 1. The AMRs for the two mentioned sentences [14].

Every AMR has a root that is unique and is displayed as the first node within the related tree. Here, we can see variables: *e*, *x*, etc., concepts such as *moan*-01 and *child*, constants like *Ribble*, and roles such as ARG0, etc. Also the slash here indicates an instance, for example: *x/child* means that *x* is an instance of the concept *child*. Furthermore, the colon symbol is a punctuation symbol to represent roles, and parentheses show which role is

related to which concept. In the structure, line breaks are optional. Additionally, AMRs can have a linear format. As an example, considering the sentence *A boy read a booklet*, the linear format is as follows:

(*e*/*read*-01: ARG0 (*x*/*boy*): ARG1 (*y*/*booklet*))

A notable feature of AMRs is the role inversion ability. This feature swaps the arguments of a selected relation, for example: R (x_1, x_2) ≡ R-of (x_2, x_1). By role inversion (R-of), AMRs with the same meaning will be created, however their structure is not equivalent. As Fig. 2 indicates, the role inversion can be applied for the above example, as its AMR has two arguments and role inversion can be applied on any of them.

(x / boy :ARG0-of (e / read-01 :ARG1 (y / booklet)))	(y / booklet :ARG1-of (e / read-01 :ARG0 (x / boy)))

Figure 2. Role inversion feature of AMR.

The left AMR in Fig. 2 places the focus on the word *boy*. In the right AMR, the focus is on the *booklet*, which paraphrases the sentence as *a booklet that was read by a boy*. This remarkable feature also has its limitations: it is not possible to focus on both *boy* and *booklet*, since it will be an AMR graph with two roots.

AMRs commonly considered as tree structures; however, they can be seen as directed acyclic graphs with a single root too, that vertices are variables and edges denote roles and instances. As a result, AMRs can be converted into sets of triples [15]. The tree structure is more useful for semantic interpretation since we must be able to determine the scope for operators like negation.

It is very simple to provide a semantic and theoretical interpretation. AMR can be made just by converting roles into two-place predicates, concepts, and events into one-place predicates, and by quantifying the existence of all variables introduced by events and concepts. Furthermore, it is noteworthy that this kind of representation does not allow us to include scope-based operators systematically, such as quantification, negation, and projection.

Moreover, a formal definition of AMRs syntax can be provided and a recursive translation function from AMR to FOL (First Order Logic) can be produced. The function, that has many similarities with the conversion from AMR to λ-calculus. The following notational definitions (simple AMRs syntax and semantics) are used in this regard [14, 16].

Definition 1 states that constants and instance assignments that decorated with outgoing roles are all AMRs. It may be a bit counterintuitive; since different types of semantic objects place in one equation.

Definition 1: $A ::= c \mid (x/P) \mid x/P\colon R_1A_1 \ldots\colon R_nA_n$

Where A_i represents i[th] AMR, x and c represent variables and constants, respectively. Also, R_i and P are equivalent to i[th] roles and properties, respectively.

The following translation function, Definition 2, would clarify the issue as it can translate all AMR structures into a kind of proposition. The most suitable method to understand is by interpreting an AMR *c* and AMR (*x*/*P*) as: *there exists an entity denoted by the constant c* and *an x with property P*, respectively.

Definition 2: $\| c.\phi \| = \phi(c)$

$$\| (x/P).\phi \| = \exists x(P(x) \wedge \phi(x))$$

$$\| (x/P\colon R_1A_1 \ldots\colon R_nA_n).\phi \| = \exists x(P(x) \wedge \| A_1.\lambda y \cdot R_1(x.y) \| \wedge \phi(x))$$

A λ-expression ϕ is a function, or subroutine without a name, which could be applied to FOL terms as arguments in order to yield new FOL expressions where the variables are bounded to the argument terms [17]. To cope with the right scope assignment, the roles translation delayed by transforming them into λ-expressions abstracting over role players. For example, the expression λy. *R*(*x*,*y*) is gotten, if the pre-translated *x* is linked to the not-translated AMR *A* (in Definition 1) via a role *R*. It results in a recursive function of translation, which maps a AMR paired with a λ-expression (for roles) to a FOL formula.

In translating process, when the target concept is related to the other ones, it is not clear that what semantic material they would introduce, and the roles should be attached to which edges. To deal with this, with the help of λ-bound formulas (that represent roles), the translation function postpones the decision. However, for starting the translation of a new AMR, we should start with the first node, which is the root. This node does not link to other concepts through outgoing roles, so it is necessary to give it a dummy tautology formula: λx.

Fig. 3 represents a sample derivation for the sentence *the teacher shouted.*

‖(e / shout-01 :ARG0 (x / teacher)), λu.⊤‖	=	[2.c]
∃e(shout-01(e)∧‖(x/teacher), λy.ARG0(e,y)‖∧λu.⊤(e))	=	[2.b]
∃e(shout-01(e)∧ ∃x(teacher(x)∧ λy.ARG0(e,y)(x))∧λu.⊤(e))	=	[β-conv]
∃e(shout-01(e)∧∃x(teacher(x)∧ ARG0(e,x))∧⊤)	=	[⊤-elim]
∃e(shout-01(e)∧∃x(teacher(x)∧ ARG0(e,x)))		

Figure 3. A sample derivation for an example sentence [14].

The produced structure is a closed formula, which means that all its variables are bounded because the translation certifies that no free occurrences of variables can be revealed. In addition, simple AMRs that are very similar to the controlled DRT[1] fragment are presented, interestingly [18]. Simple AMRs are in the two-variable fragment of FOL. It should be noted that FOL is not decidable. In contrast, the two-variable fragment is a decidable FOL, in which formulas have a maximum of

[1] Discourse Representation Theory

two variables with different names; however, it does not have function symbols, yet probably have equality. In addition, it has the property of a finite model, that is, if a fragment formula can be satisfied, it can also be satisfied in a finite model.

Fig. 4 illustrates an example of AMR annotation of the sentence *the police want to arrest Micheal Karras* from real data. Usually, nodes recognized with their variable. For instance, *w* labeled with the concept *want-*01. Moreover, the labeled edges connecting nodes are relations, such as ARG0. Moreover, nodes that do not have variables are constants, such as *Michael*. They usually used to represent name, negation, or number.

The mentioned figure shown that usually AMR concepts can be related with a single word in the sentence that constitutes a one-to-one mapping. However, sometimes there are concepts, which cannot easily be associated with any specific word in the sentence. These concepts usually indicate inferred knowledge, which invoked by certain phrases or implicit relationships between disparate clauses. This type of concepts called *Abstract Concepts*. For instance, the concept *person* in Fig. 4 is an inferred named entity type for *Michael Karras*.

Banarescu et al. in [12] claimed that AMR cannot be considered as an Interlingua, the characteristic that it abstracts away from surface Morpho-syntactic differences, makes it very attractive to implement cross-lingual AMR banks based on resembling principles.

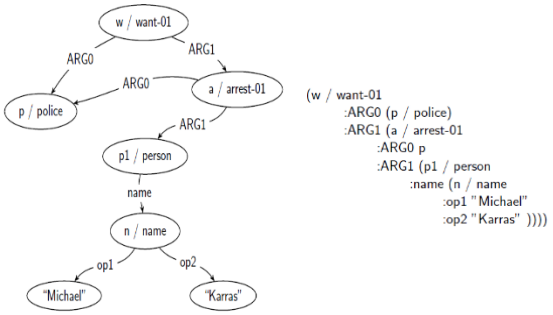

Figure 4. AMR graph and the related PENMAN notation [19].

Furthermore, Li et al. [20] shown that it is completely possible to align English-Chinese AMRs. According to this study of 100 English-Chinese sentences, which have been annotated manually with AMR pairs, the authors displayed that AMR formalism can be feasibly applied for other languages than English.

Recently, Wein and Schneider in [21] proposed an annotation layout for structural divergences classification in cross-lingual meaning representations. They also produced 50 related Spanish- English annotated examples. They presented that structural divergence in cross-lingual meaning representations pairs can considered as a meaningful proxy of divergences between parallel texts. Hence, tools that depend on highly literal translations, like pre-trained MT systems, can efficiently exploit this structural divergence annotation layout to cross-lingual AMR of the data. To be more precise, they introduced a categorization to specify both the type and the reason of the divergence as being due to semantic, annotation or syntactic divergence.

B. Improved AMR

Some recent research tried to improve the basic version of AMR and concentrated on AMR's challenges and shortcomings. In this regard, Pustejovsky et al. [22] introduced an extension to AMRs that tackled the semantic weaknesses of AMR while keeping its cognitive plainness. In particular, they handled quantification, negation, and modality phenomena that had not been among AMR specifications previously. Their proposed representation maintained the predicative nature of AMR and embedded it under a scope graph when it is needed. Plus, their proposed representation was different from other treatments of modal and quantification scope phenomena because it was more transparent and defined default scope whenever feasible. Fig. 5 illustrates the representation for a sentence according to their proposed model and shows their representation was suitable and efficient, because a rooted graph structure could be kept with the scope relation as the root node.

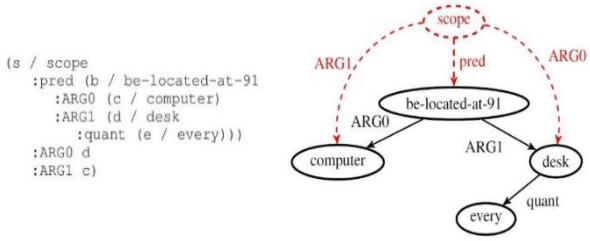

Figure 5. Representation for the sentence *A computer is on every desk* [22].

O'Gorman et al. [23] added a layer of annotation on top of the AMR-2017 graph-bank (LDC2017T10) and produced a corpus called Multi-Sentence AMR (MS-AMR). The added layer represented co-reference and implicit arguments beyond the sentence level. Fig. 6 illustrates an example in which each <identchain> element assembles mentions of the same entity. Unlike other co-reference annotation layouts, these mentions are nodes in the AMR graphs (not pieces of text). Besides, the annotation distinguishes what implicit roles of predicate nodes the entity fills.

```
−<identchain relationid="rel-3">
   <mention concept="he" id="DF-200-192400-625_7557.12" variable="h"/>
   <mention concept="person" id="DF-200-192400-625_7557.11" variable="p"/>
   <implicitrole argument="ARG0" id="DF-200-192400-625_7557.12"
                 parentconcept="want-01" parentvariable="w2"/>
</identchain>
```

Figure 6. Co-reference chain example in MS-AMR corpus [24].

Anikina et al. [24] assessed the performance of various co-reference resolution implements on the MS-AMR annotations. They worked on the token level by mapping the co-reference annotations from the nodes to the sentences and on the node level by mapping the implements' co-reference predictions to the nodes of the graph. Consequently, they recognized that AllenNLP with SpanBERT embedding, in general, attained the best results. Afterwards, they presented how the output of a co-reference model can be integrated into the predictions of an AMR parser. They exploited the neural semantic parser that produced a graph for the input sentence compositionally, with the aim of mapping co-referent input tokens to the predicted graph's nodes. The co-reference chains, which are annotated in the MS-AMR corpus are really heterogeneous. At the token level, mentions of the same chain can be represented as nouns, pronouns or verbs. Fig. 7 presents an example in which the chain consists of various concepts at the node level: *it*, *thing*, *harm*-01, *cut*-01. These kinds of chains are not easy to predict for the AllenNLP co-reference model since they have different POSs and are semantically nontrivial (*cut*/*harm*). In the test set of this corpus, 35% of all co-reference chains contained entities, which were expressed with multiple different POS.

```
−<identchain relationid="rel-1">
   <mention concept="it" id="DF-200-192400-625_7557.9" variable="i2"/>
   <mention concept="thing" id="DF-200-192400-625_7557.24" variable="t3"/>
   <mention concept="it" id="DF-200-192400-625_7557.27" variable="i2"/>
   <mention concept="harm-01" id="DF-200-192400-625_7557.35" variable="h"/>
   <mention concept="cut-01" id="DF-200-192400-625_7557.3" variable="c4"/>
   <mention concept="do-02" id="DF-200-192400-625_7557.17" variable="d2"/>
   <mention concept="cut-01" id="DF-200-192400-625_7557.8" variable="c"/>
   <mention concept="this" id="DF-200-192400-625_7557.29" variable="t"/>
   <mention concept="harm-01" id="DF-200-192400-625_7557.1" variable="h"/>
   <mention concept="cut-01" id="DF-200-192400-625_7557.6" variable="c"/>
   <mention concept="it" id="DF-200-192400-625_7557.36" variable="i"/>
   <mention concept="thing" id="DF-200-192400-625_7557.28" variable="t2"/>
</identchain>
```

Figure 7. Heterogeneous co-reference chain example from MS-AMR [24].

Although, AMR parsing approach already manages some cases of co-reference in the AMR graphs, some AMR nodes can create co-reference chains without having any token alignments. For instance, the AMR graph of the sentence *Speak to a consultant* has a separate node *you* as ARG0 of *speak*-01. However, this node does not correspond to any token in the sentence. In the test set of this corpus, 9% of all co-referent mentions do not have any alignments and the token-based co-reference resolvers could not tackle them.

Another point which was proved in [24] was that incomplete or incorrect node-token alignments could have negative effect on the performance. This happens when the gold annotation contains generic concepts, which are represented in the AMR graphs but not detected at the token level. In the test set of this corpus, 10% of all co-referent nodes referred to generic concepts such as *country*, *person*, or *thing*. It becomes an issue when AllenNLP finds the co-reference with more particular nodes like *dad* or *China*. In brief, tokens like *dad* is aligned to the related node like *dad* in the AMR graph whilst the more generic nodes like *person* do not have an alignment. Although, the gold co-reference chain contained only *person* as a member that results in the incorrect classification of *dad* as false positive however both nodes correspond to the same entity actually.

Additionally, the authors recognized cases of wrongly resolved personal pronouns as some texts were borrowed from forums and the spokesman could switch during the conversation. Therefore, different people could understand different meanings. For instance, in the sentence *Or should* I_1 *... just keep an eye on the anxiety until it becomes a problem? Well* I_2 *wouldn't try to keep an eye on anxiety for a start because that will make* u_1 *tense* from MS-AMR test set, the first sentence has the pronoun I_1 which refers to the same entity as u_1 in the second sentence. The I_2 pronoun in the second sentence corresponds to a different speaker. As the input text for the co-reference tool did not have any Meta information about the speakers, the tool resolved both occurrences of *I* as referring to the same entity. In the test set of this corpus, this problem influenced 9% of the co-reference chains.

In another research, Fu et al. [25] designed an end-to-end AMR co-reference resolution framework with the aim of creating multi-sentence AMRs. Their proposed model diminished error propagation in comparison with the previous rule-based and pipeline methods. Besides, for both in- and out-domain situations, it was more robust. Considering two sentences; *Bill left for Paris* and *He arrived at noon*, Fig. 8 illustrates the schema of their proposed framework that included a graph encoder, a concept identifier, and an antecedent prediction module.

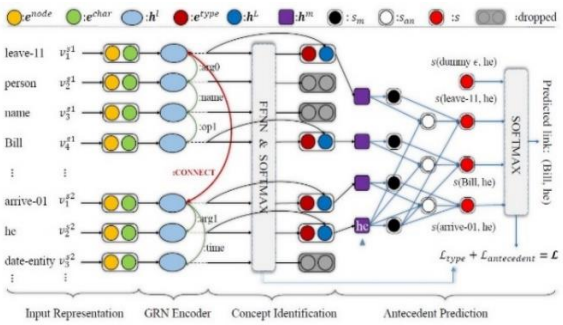

Figure 8. The end-to-end AMR co-reference resolution framework proposed in [25].

As seen in Fig. 8, they exploited a graph neural network to represent input AMRs for inducing expressive characteristics. They built connections between sentence-level AMRs by linking their roots, with the aim of enabling cross-sentence information exchange. More, they defined a concept identification module to detect functional graph (non-concept) nodes like *person*, entity nodes such as *Bill*, verbal nodes with implicit role like *arrive*-01 and other regular nodes such as *leave*-11 to

upgrade the performance level. The final prior prediction is taken from the chosen nodes and all their conceivable prior candidates.

AMR does not have a systematic treatment of projection phenomena, thus, its translation to logical form is not easy. Lai et al. [26] designed a translation function from AMR to FOL by applying continuation semantics that enabled to detect the semantic context of a statement in the form of an argument. It was a natural extension of AMR's principal structure, which aimed to model basic projection phenomena, like negation and quantification, as well as complicated phenomena, like donkey anaphora and bound variables. In their definition, a continuation of an expression encodes surrounding contextual information related to its interpretation. In particular, the continuation assumption supposed that some natural language statements define methods, which consider their own semantic context as an argument. According to their work, continuations had remarkable results for the representations related to the predicative core in the structure of an AMR graph. A case in point is that it enabled the graph to be rooted at the predicate level as well as considering the continuation as a correlated argument to the relation related to that predicate. Therefore, their approach enabled to use standard AMRs without attaching properties or changing the graphs, plus allowed to extract valid inferences.

In general, continuations could be considered as a pragmatic solution to AMR to FOL translation issue. Considering an expression's continuation as an associated argument to the relation associated with that predicate, the AMR's focus was maintained on the predicative core, however, still enabled valid inferences from projection phenomena to fall out. The model proposed in [26] did not have under-specification problem, so it allowed both to prioritize the most feasible interpretation of a scope ambiguity and to detect fewer common interpretations where necessary. Additionally, this model did not alter standard AMR edges, nodes, or leaves, enabling to exploit existing AMR corpora. Mapping fundamental projection phenomena to AMR could smooth the path for more comprehensive meaning representation, enabling plain translation of complicated phenomena, like negative raising that prove speaker intent and belief. Another remarkable superiority of the continuation-passing model for utterance or sentence level expressions was the modest way it could be extended to AMR models that have been used in human-robot interaction dialogues. In other words, their introduced model could adopt the dynamic semantics of discourse moves as continuations. In addition, in continuation semantics, the order of application specified the relative scope of a predicate and each of its arguments. Unlike previous research, in [26] the outgoing roles of a predicate which should be ordered were taken, and the order of application to be the order the arguments were written in the AMR.

Bonn et al. [27] proposed an expansion to the AMR annotation layout, which detected fine-grained pragmatically and semantically inferred spatial information in grounded corpora. They introduced a lexical group conceptualization and set of spatial annotation tools created in the context of a multimodal corpus including 185 3D structure-building dialogues between a human architect and builder in Minecraft. Minecraft presented a specifically advantageous spatial relation-elicitation environment as it automatically tracked orientations and locations of objects and avatars in the space based on an absolute Cartesian coordinate system. Via a two-step document and sentence level annotation process proposed to detect implicit information, they exerted these bearings and coordinates in the AMRs together with spatial framework annotation to build the spatial language in the ground of dialogues to absolute space. This supplement took the fine-grained spatial semantics and object grounding approaches of previous layouts and apply them into MS-AMR. The outcome was an annotation tool that could manage fine-grained implicit and explicit nested spatial relations, which were structured in quantified space and combined fluidly with event dynamics. In addition, as the spatial annotations were merged into the domain-general AMR graphs, their proposed method determined information about the way spatial relations were displayed in the context of whole sentences and overall discourse. In brief, they proposed span single- and multi-sentence annotation. At single-sentence one, a general semantic roles and frames set beside relation-specific role-sets were introduced. Both the role-sets and conceptualization targeted lexical units from various POSs (and then adverbs and prepositions) and took considered intrinsic properties and extrinsic relations related to orientation, location, configuration, direction, extent, topology, and particularly Frame of Reference (FoR). At the multi-sentence one, they introduced layers of co-reference annotation and bridging which detected implicit spatial knowledge and assisted in grounding. A significant addition to MS-AMR was incorporating an existential dummy AMR graph, which placed the configurational stage for the spatial entities represented in the dialogue. This dummy AMR graph introduced especial spatial frameworks for each entity and the environment and explained the way these frameworks mapped together. Although Spatial AMR targeted to be adaptable to other environments, they implemented it in the particular context of their corpus of Minecraft structure-building dialogues as an example of its range and characteristic. This annotated corpus was applied to train a semantic parser, for which they reported primary baseline outcomes.

As mentioned before, AMR does not represent scope information, which leads to an issue for its total expressivity and particularly for drawing inferences from negated expressions. This phenomenon is called «positive interpretations» of negated expressions, where implicit positive meaning is detected using the opposite of the negation's focus inference. Stein and Donatelli [28] studied methods for representing Potential Positive Interpretations (PPIs) in AMR. They defined a logically motivated AMR structure for PPIs, which built the focus of negation explicit and planned an initial proposal for a systematic approach to produce this more expressive structure. In this regard, they attempted to detect if these structures can be systematically generated from the negated sentences AMR. Thus, they introduced a logically motivated AMR structure which made both negation focus and scope explicit. Further, they modeled an initial plan for transforming common AMRs to this more expressive model. Their primary evaluations about scope explicit supported previously made experiments that AMR missed expressive capacity. According to their observation, this weakness was particularly problematic for PPIs whose bound meaning results from an interaction of the information and negation structure. In brief, their proposed method allowed expressing the negations foci in AMR without altering the principal AMR.

As AMR cannot represent non-veridical intentional contexts completely, Williamson et al. [29] addressed the problem of non-veridicality without resorting to layered graphs via a mapping from AMRs into Simply-Typed Lambda Calculus (STLC). This solution, in some cases, needed a new role, called *:content*, to be introduced that worked as an intentional operator. Besides, in de re or de dicto ambiguities they tackled the quantifier scope interaction and intentional operators. With this aim, they added a scope node and designed an explicit multi-dimensional meaning using Cooper storage that resulted in deriving the de re and de dicto scope readings and intermediate scope readings. It is noteworthy that the *:content* role and its intentional translation can facilitate downstream NLP tasks. Particularly, different state predicates trigger different lexical inferences due to their semantic nature and state considering whether they are non-veridical like *believe*-01, factive like *know*-01 or counter-factive such as *pretend*-01.

It has been mentioned that AMR has potential usefulness in NLP tasks like MT and NLG. Although, it suffers from lacking some aspects, like eliminating implicit time information that carries meaning. To cover this weakness, Donatelli et. al. [30] attempted to add aspect and tense tags to AMR, which leads to enhance its ability in meaning representation. Bakal [31] designed a rules-based approach to attach the roles from Donatelli et al. to standard AMR trees, applying semantic information encoded in the dependency tree representation on the same sentence. In brief, Bakal's study had presented the necessity of a couple of modifications related to the time rules in the standard AMR. In these works, all the represented sentences had at least one type of aspect and tense noted implicitly, and in most cases, eliminating the aspect and tense would result in a notable difference in meaning. As they mentioned, on condition that AMR wants to be applied as a pragmatic meaning representation model, it requires including this information. In this regard, the proposed method could be exploited to fortify it. Besides, it can be considered that the Bakal's classifier was fairly accurate at encoding and detecting the needed information.

III. AMR PARSING

The process of mapping sentences in natural language to their semantic representations is Meaning Representation or Semantic Parsing. Therefore, the AMR parsing task is mapping natural language strings to AMR semantic graphs. In recent releases of the AMR bank, a great sympathy in this task has drowned and a considerable number of efforts has been done on it as follows:

Szubert et al. [32] mentioned and discussed various linguistic phenomena responsible for reentrancies in AMR, like Co-reference, Coordination, Control, Adjunct control, Ellipsis, Relative clause, Nominal Control and Verbalization, then grouped reentrancy triggers in three types: syntactic, pragmatic, and AMR-specific. Additionally, they divided error types that AMR parsers usually make with respect to it in different categories. Table 1 depicts percentage of reentrancies in the LDC2015E86 training set according to their paper.

TABLE I
PERCENTAGE OF REENTRANCIES IN LDC2015E86 TRAINING SET.

Phenomenon	Frequency (%)
Co-reference	37
Adjunct control	16
Control verbs	4
Coordination	17
Verbalization	14
Pragmatic overreach	3
Ellipsis	2
Control-like structure	2
Annotation mistakes	5
Total	100

Accordingly, they ran oracle experiments to probe the effects of the error types on both overall parsing score and reentrancy prediction (Table 2). The actions they applied are as follow:

- ADD: Adding a reentrancy edge.
- ADD-ADDN: Adding a reentrancy edge and a node.
- REMOVE: Removing a reentrancy edge.
- REMOVE-RMN: Removing a reentrancy edge and a node.

- MERGE: Merging two nodes.
- MERGE-RMN: Merging two nodes and removing a node.
- SPLIT: Splitting a node in two already existing nodes.
- SPLIT-ADDN: Splitting a node in one existing node and a new node.
- ADD-SIB: Adding an edge between siblings.
- ADD-SIB-ADDN: Adding a node and an edge with one of its siblings.
- REMOVE-SIB: Removing an edge between siblings.
- REMOVE-SIB-RMN: Removing an edge between siblings and one of the siblings.

TABLE 2
RESULTS OF APPLYING ACTIONS ON THE TEST SPLIT OF LDC2015E86 AND LDC2017T10 [32].

Action	LDC2015E86			LDC2017T10		
	Frequency	Smatch	Reentrancy	Frequency	Smatch	Reentrancy
ADD	1292.0	1.7	10.4	1305.7	1.7	10.3
ADD-ADDN	330.0	0.8	4.2	281.3	0.7	3.1
RM	545.7	0.4	-0.1	572.3	0.4	-0.1
RM-RMN	217.0	0.3	0.6	224.7	0.2	0.8
MERGE	187.3	0.4	1.6	193.3	0.4	1.7
MERGE-RMN	94.3	0.3	1.0	84.0	0.2	0.9
SPLIT	574.7	1.2	1.8	541.3	1.1	1.7
SPLIT-ADDN	333.0	0.9	-0.2	347.3	0.9	0
ADD-SIB	128.0	0.2	1.3	119.7	0.1	1.2
ADD-SIB-ADDN	99.7	0.1	-0.1	104.3	0.1	0
RM-SIB	69.3	0.1	0.2	89.3	0	0.2
RM-SIB-RMN	0	0	-0.1	0	0	0
All	3108.3	4.6	18.8	3093.7	4.4	18.0

In Table 2, Frequency is the number of times the action could be applied, Smatch equals the parsing score, and Reentrancy shows the reentrancies prediction score. The table showed that the best improvements could be observed when applying all actions and the most relevant single oracle action was ADD. By correcting these errors, they could enhance Smatch in parsing performance and in reentrancy prediction. They detected main sources of reentrancies which have been ignored before. However, their heuristic models failed to find the causes of many reentrancies.

We have categorized the AMR parsing methods into five main groups, which in the following sections, we will explain previous research in each category.

A. Grammar-based method

In these methods, a grammar is used to limit the graphs that are desired during a parsing process. Then, the graph with the highest score, which determined by a scoring function, would be the output. There are many grammar-based approaches to AMR parsing. Most of them include parsers based on Synchronous Hyper-edge Replacement Grammars (SHRGs), Combinatory Categorical Grammars (CCG), and Directed Acyclic Graph (DAG) automata.

Chiang et al. [33] represented HRG[1], which is one of the leading context-free rewriting grammars. It can be applied as an impressive formalism for graph recognition task on NLP applications, which works with large-scale graphs. They also expanded the HRG formulation to its Synchronous version SHRG, as the foundation for possible related tasks, like graph generation and parsing.

Later, Peng et al. [34] and Peng and Gildea [35], for the first time suggested a real system for parsing AMRs with the use of SHRG. This parser modeled natural language strings with CFG[2] and AMR graphs with HRG. Then, in the source side, a synchronous grammar was formalized with CFG and similarly, in the target side, with HRG. Having this synchronous grammar, the proposed parser could parse the considered sentence using CFG rules, and meantime could evolve the related graph from the derivation by deduction. Afterwards, to extract the appropriate set of SHRG rules from the training data, a sampling algorithm called Markov Chain Monte Carlo (MCMC) was used to learn the most probable derivation according to sentence to AMR alignments. They exploited MCMC algorithms to learn SHRG rules from a forest that represents likely derivations consistent with a fixed string-to-graph alignment. Fig. 9 illustrates a sampled derivation from the forest.

[1] Hyper-edge Replacement Grammars

[2] Context Free Grammar

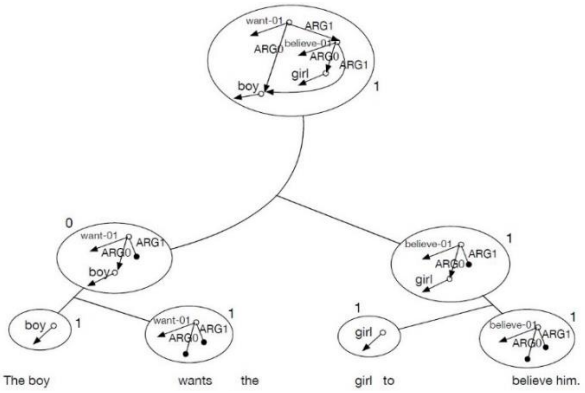

Figure 9. The sampled derivation for the (sentence, AMR graph) pair [34].

Another grammar formalism is Combinatory Categorical Grammar (CCG) which establishes a transparent interface between underlying meaning representation and surface syntax, has been widely applied in semantic parsing applications. First time Artzi et al. [36] suggested applying CCG formalism for parsing AMR. They proposed a process with two phases, to take advantage of this formalism completely and keep a relatively compact grammar. In the first phase, CCG is used for parsing a considered sentence to an unspecified logical structure. It was a lambda-calculus equal form of AMR representation, which does not include non-compositional reasoning. In the second phase, a factor graph model eliminates the unspecified part of structure that needs global deduction. CCG-based AMR parser has some notable benefits. For example, it does not need a sentence-to-AMR alignment. This feature helps to prevent the error propagation from the aligner. Furthermore, the feature enables it to be generalize to more languages. Fig. 10 illustrates the factor graph used in generating the complete derivation for the sentence *Pyongyang officials denied their involvement*, including all the variables and a subset of the factors.

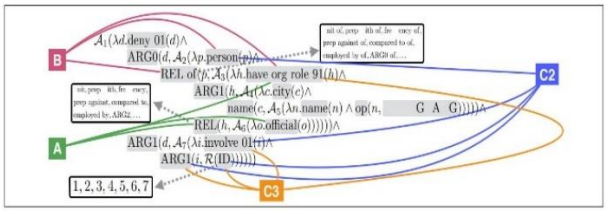

Figure 10. A visualization of the factor graph constructed for an example sentence's derivation [36].

In Fig. 10 variables are highlighted in gray and the set of possible assignments are marked with a dashed arrow. Plus, solid lines represent edges. This example just includes a subset of the factors. Factor *A* takes selectional preference between the have-org-role-91 and *official* to detect the REL relation. Factor *B* does the same for *person* and have-org-role-91 to detect REF-of. Both factors C2 and C3 account for selectional preferences when resolving ID. In C2, they considered the assignment 2, which makes an ARG1 relation between *involve*-01 and *person*. Similarly, C3 considers the assignment 3.

B. Graph-based method

These techniques attempt to construct a graph, by maximizing a scoring function for graphs.

Flanigan et al. [37] introduced Graph-based AMR parsing methods and JAMR was the first AMR parser they proposed. The idea of JAMR method is inspired from graph-based techniques that are suitable for non-projective syntactic dependency parsing. This approach is extensively used for identifying relations. They proposed an algorithm, which first identified the concepts by using a semi-Markov model and then identified the relations between them by searching for a Maximum Spanning Connected subGraph (MSCG). In order to get English AMR alignments in the pre-processing step, they represented a heuristic aligner based on a pre-defined set of rules.

Jones et al. [38] introduced the first meaning-based SMT by applying a graph-structured semantic representation as a transfer layer between the source and the target language. They defined 3 methods for every stage of the semantics-based translation pipeline: one graph-to-word alignment algorithm and 2 synchronous grammar rule extraction algorithms. Plus, they studied the impact of syntactic annotations on semantics-based translation by designing 2 alternative rule extraction algorithms: one that needed just semantic annotations and one that used syntactic annotations and checked the influence of language and syntax bias in meaning representation structures by doing tests with 2 meaning representations, one biased toward an English syntax-like structure and one that was language neutral.

Pourdamghani et al. [39] presented a statistical alignment model that unlike heuristic approaches, would automatically improve as more data becomes available. While the potential applications of AMR are manifold, their research explored the use of AMR in SMT. Their proposed model had 3 phases: pre-processing, training, and post-processing. In the pre-processing phase, they linearized the AMR graphs in order to convert them into strings, clean both the AMR and English sides by eliminating stop words and simple stemming and add a set of corresponding AMR/English token pairs to the corpus to help the training phase. The training phase was based on IBM models, but they modified the learning algorithm to symmetrically learn the parameters. Eventually, in the post-processing stage they rebuild the aligned AMR graph.

In addition, Xue et al. [40] studied the use of AMR as mediator between multiple languages, suggesting that a higher level of abstraction might be significant to account

for lexicalization differences between languages. They translated 100 English sentences that had AMRs into Chinese and Czech in order to build AMRs for them. Then, they did a cross-linguistic comparison of English to Chinese and Czech AMRs, and it showed that in both cases the structure of AMRs for the language pairs were aligned acceptably and cases of linguistic divergence. Additionally, they recognized that the level of compatibility of AMR between English and Czech is lower than between English and Chinese. Thus, they declared that these comparisons could be useful for further improving the annotation standards for each of the 3 languages and can result in more compatible annotation guidelines between the languages.

In recent years, the JAMR parser framework has been investigated in two research. Both considered that the concept identification issue is a serious bottleneck and worked on it to solve this problem in the parser. Werling et al. [41] proposed an action set, which generate concepts, and during the concept identification stage, trained a classifier to generate extra concepts. These actions are very similar to the sources of concepts that proposed by Flanigan et al. [42]. Moreover, in the latter paper, authors suggested the infinite ramp loss to enhance the performance.

In another research, Zhou et al. [43] introduced a graph-based parser that applied beam search method to recognize both relations and concepts. They applied the method for relation identification in an incremental fashion (Fig. 11 – Algorithm 1), and then incorporated the decoder into a unified framework based on multiple-beam search (Fig. 11 – Algorithm 2), which allowed for the bi-directional information flow between the two subtasks in a single incremental model. They assigned weight to relations and concepts. Therefore, the resulting graph would have the highest score that satisfied similar constrains as JAMR approach. In addition, they applied the feature proposed in [37], and received an increase value for Smatch [15] F1 score by five points. Furthermore, in their research more features for concept ID were also introduced, which results in another improvement by 3 points.

Algorithm 1 The incremental decoding algorithm for relation identification.

Input: A sequence of concept fragments $(c_1, c_2, \ldots, c_n)$
Output: Best AMR graph including $(c_1, c_2, \ldots, c_n)$

$agenda \leftarrow$ {Empty-graph}
for $i \leftarrow 1 \ldots n$ **do**
  for $state$ in $agenda$ **do**
    CalEdgeScores($state, c_i$)
    $(p_1, p_2, \ldots, p_m) \leftarrow$ FindComponents($state$)
    $innerAgenda \leftarrow state$
    for $j \leftarrow m \ldots 1$ **do**
      $buf \leftarrow NULL$
      for $item$ in $innerAgenda$ **do**
        if $i < n$ **then**
          //Add only c_i to the item
          $newitem \leftarrow item$
          AddItem($newitem, c_i$)
          AppendAgenda($buf, newitem, i, n$)
        // Add a left arc from c_i to p_j to the $item$
        $newitem \leftarrow item$
        $le \leftarrow$ GetMaxLeftEdge(c_i, p_j)
        AddItem($newitem, c_i, le$)
        AppendAgenda($buf, newitem, i, n$)
        //Add a right arc from p_j to c_i the $item$
        $newitem \leftarrow item$
        $re \leftarrow$ GetMaxRightEdge(p_j, c_i)
        AddItem($newitem, c_i, le$)
        AppendAgenda($buf, newitem, i, n$)
        //Add both left and right arc to the $item$
        AddItem($item, c_i, le, re$)
        AppendAgenda($buf, item, i, n$)
      $innerAgenda \leftarrow B$-best(buf)
  $agenda \leftarrow innerAgenda$
return $agenda$[0]
function AppendAgenda($buf, item, i, n$)
  //parameter n represents the terminal position
  if $i = n$ **then**
    CalRootFeatures($item$)
  AppendItem($buf, item$)

Algorithm 2 The joint decoding algorithm.

Input: Input sentence $x = (w_1, w_2, \ldots, w_n)$
Output: Best AMR graph derived from x

$agendas[0] \leftarrow \emptyset$
$last \leftarrow$ Scan(x)
for $i \leftarrow 1 \ldots n$ **do**
  $list \leftarrow$ Lookup(x, i)
  if $list$.size > 0 **then**
    $preAgenda \leftarrow agendas[i-1]$
    for $cf \in list$ **do**
      $end \leftarrow i + cf$.size $- 1$
      if $preAgenda$.size $= 0$ **then**
        $g \leftarrow$ Graph.empty
        CalConceptFeatures(g, cf)
        AppConcept($agendas, end, g, cf, last$)
      else
        for $item \in preAgenda$ **do**
          $g \leftarrow item$
          CalConceptFeatures(g, cf)
          AppConcept($agendas, end, g, cf, last$)
    Union($agendas, i, i-1$)
  else
    $agendas[i] \leftarrow agendas[i-1]$
$bestGraph \leftarrow agendas[last][0]$
return $bestGraph$

Figure 11. The pseudocode for algorithms used in [43].

The next graph-based AMR parser is the one that has been represented in [44] and then improved in [45]. In this parser, five Bi-LSTM[1] networks were used, to make probability prediction for concepts, attributes, named entities, core argument relations and non-core relations. Like the JAMR approach that has been introduced before, this parser detected concept fragments first. Afterwards, it linked them to each other via adding some edges meet similar constraints, exactly like JAMR approach. The remarkable difference between this parser

[1] Long Short-Term Memory

and JAMR, is the algorithm that has been used in this parser to add the required edges. This algorithm is greedy, that calculated probabilities just after adding each edge.

Rao et al. [46] used learning methods, as search tools, to predict (find) target AMR graphs. Same as JAMR approach, concepts and relations are predicted, respectively. In the next step, this prediction may use the previous predictions results. In their research, the decomposer (graph to fragments) and the aligner, were completely similar to JAMR, however, it did not have a linkage constraint. Instead, the graph is linked by choosing a top node and linking all components to this selected node.

C. Transition-based method

These algorithms produce a graph within a sequence of actions. When an action is chosen, the alternatives of that action are not revised again.

Parsers, which use these methods, consider the parsing process as application of a sequence of actions. Wang et al. [47], for the first time, introduced the idea of these algorithms for AMR parsing as transition-based dependency parsing algorithms. When a transition-based algorithm is applied, the AMR graph would be created in an incremental way, using actions that were chosen by a desired classifier.

The transition-based method has been applied in previous research to transform the existing dependency trees to AMR graphs, and to directly transform sentences with possible extra annotations to AMR graphs [48-52].

For instance, Wang et al. [53] proposed a two-stage framework to parse a sentence into its AMR. Firstly, they applied a dependency parser to generate a dependency tree for the sentence. Secondly, they introduced a transition-based algorithm that transforms the dependency tree to an AMR graph. In this regard, they defined eight types of actions for the actions set.

Brandt et al. [54] tried to improve AMR parsing using preposition semantic role labeling information retrieved from a multi-layer feed-forward neural network. Prepositional semantics was included as features into the transition-based AMR parsing system CAMR. The inclusion of the features changed the CAMR behavior when creating meaning representations triggered by prepositional semantics. Prepositions in conjunction with their arguments made a significant contribution to the meaning of sentences and therefore were a very intuitive supplement to AMR parsing.

Goodman et al. [55] designed 2 extensions to AMR: 1) Noise reduction, 2) Targeted exploration. The first one targeted the result of the complexity of the task and relieved the noise in the feature representation. The second one aimed the exploration steps of imitation learning towards scopes which were probable to give the most information in the context of a large action-space. They considered imitation learning methods as a toolbox which could be adjusted to fit the task's specifications.

Ballesteros and Al-Onaizan [56] proposed an AMR parser which generated AMR parses from plain text directly. They applied Stack-LSTMs in order to represent the parser state and make decisions greedily. The input of their parser was plain text sentences, and, through rich word representations, it predicted all actions (in a single algorithm) needed to generate an AMR graph representation for an input sentence; it managed the detection and annotation of named entities, word sense disambiguation and it made connections between the nodes detected towards building a predicate argument structure. Although the system which runs with just words is very competitive, they further improved the results incorporating POS tags and dependency trees into their model. Table 3 presents the parser actions and the effect on the parser state (contents of the stack, buffer) and how the graph was changed after applying the actions.

TABLE 3
PARSER TRANSITIONS [56].

Stack_t	Buffer_t	Action	Stack_{t+1}	Buffer_{t+1}	Graph
S	*u, B*	SHIFT	*u, S*	*B*	-
u, S	*B*	CONFIRM	*n, S*	*B*	-
u, S	*B*	REDUCE	*S*	*B*	-
u, v, S	*B*	MERGE	*(u, v), S*	*B*	-
u, S	*B*	ENTITY(l)	*(u : l), S*	*B*	-
u, S	*B*	DEPENDENT(r. d)	*u, S*	*B*	$u \xrightarrow{r} d$
u, v, S	*B*	RA(r)	*u, v, S*	*B*	$u \xrightarrow{r} v$
u, v, S	*B*	LA(r)	*u, v, S*	*B*	$u \xrightarrow{r} v$
u, v, S	*B*	SWAP	*u, S*	*V, B*	-

Moreover, Fig. 12 depicts transition sequence for the sentence *It should be vigorously advocated*.

ACTION	STACK	BUFFER
INIT		It, should, be, vigorously, advocated, R
SHIFT	it	should, be, vigorously, advocated, R
CONFIRM	it	should, be, vigorously, advocated, R
SHIFT	should, it	be, vigorously, advocated, , R
CONFIRM	recommend-01, it	be, vigorously, advocated, R
SWAP	recommend-01	it, be, vigorously, advocated, R
SHIFT	it, recommend-01	be, vigorously, advocated, R
SHIFT	be, it, recommend-01	vigorously, advocated, R
REDUCE	it, recommend-01	vigorously, advocated, R
SHIFT	vigorously, it, recommend-01	advocated, R
CONFIRM	vigorous, it, recommend-01	advocated, R
SWAP	vigorous, recommend-01	it, advocated, R
SWAP	vigorous	recommend-01, it, advocated, R
SHIFT	recommend-01, vigorous	it, advocated, R
SHIFT	it, recommend-01, vigorous	advocated , R
SHIFT	it, recommend-01, vigorous	advocated, R
SHIFT	advocated, it, recommend-01, vigorous	R
CONFIRM	advocate-01, it, recommend-01, vigorous	R
LA(ARG1)	advocate-01, it, recommend-01, vigorous	R
SWAP	advocate-01, recommend-01, vigorous	it R
SHIFT	it, advocate-01, recommend-01, vigorous	R
REDUCE	advocate-01, recommend-01, vigorous	R
RA(ARG1)	advocate-01, recommend-01, vigorous	R
SWAP	advocate-01, vigorous	recommend-01, R
SHIFT	recommend01, advocate-01, vigorous	R
SHIFT	R, recommend01, advocate-01, vigorous	
LA(root)	R, recommend01, advocate-01, vigorous	
REDUCE	recommend01, advocate-01, vigorous	
REDUCE	advocate-01, vigorous	
LA(manner)	advocate-01, vigorous	
REDUCE	vigorous	
REDUCE		

```
(r / recommend-01
    :ARG1 (a / advocate-01
          :ARG1 (i / it)
          :manner (v / vigorous)))
```

Figure 12. Transition sequence for an example sentence (*R* represents the root symbol) [56].

Damonte et al. [57] introduced a transition-based approach which created AMR graphs in linear time via processing the sentences in a left-to-right manner. This approach was inspired by the ArcEager transition method for dependency tree parsing and was trained with feedforward neural networks. The authors noted that there are three main differences between AMRs and dependency trees which need extra adjustments for dependency parsers to be applied on AMRs. The first difference between these two structures is projectivity. English dependency trees are commonly projective, which means that there will not be any crossing arcs on condition that the edges have been drawn in the semi-plane above the words. Although this limitation is agitated in English syntactic theories, it is not motivated for AMR structures. Secondly, unlike dependency trees, AMRs are graphs (not trees), as they could have nodes with multiple parents (reentrant nodes). Thus, in order to handle reentrancy, they dropped this constraint. Thirdly, in AMR there is not any straight mapping between a node in the graph and a word in the sentence. Hence, to determine alignments between the tokens in the sentence and the nodes in the AMR graph, they ran the JAMR aligner.

Peng et al. [58] introduced a technique that linearized AMR graphs in a way that could detect the interaction of relations and concepts. With the aim of tackling the data sparsity problem for the target vocabulary, they designed a categorization approach that initially mapped low frequency concepts and entity subgraphs in order to a reduced set of category types. Plus, they applied heuristic alignments in order to connect source side spans and target side concepts or subgraphs, for mapping each type to concepts of its corresponding target side. In decoding phase, they exploited the mapping dictionary, which was learned from the training data, or heuristic rules for certain types with the aim of mapping the target types to their corresponding translation as a post-processing procedure. In general, they introduced multiple techniques to create the sequence-to-sequence model work competitively against conventional AMR parsing systems.

Groschwitz et al. [59] proposed a semantic parser for AMR that parsed strings into tree representations of the compositional construction of an AMR graph. They exploited standard neural methods for super-tagging and parsing dependency tree, limited by a linguistically principled typed system. To be more precise, they tried to make the compositional structure of the AMR explicit. They considered an AMR as concepts including atomic graphs that represent the words meanings. Besides, they merged compositionally by applying linguistic operations for merging a head with its modifiers and arguments. The authors designed this construction as terms over the AM algebra as proposed in [60], which had not any parser. In this regard, they illustrated that these terms can considered as dependency trees of the string, and they defined a dependency parser to map it to the related tree. In other words, they merged a neural super-tagger to detect the initial graphs for the individual words with a neural dependency approach. One significant issue is that the decoding issue was NP-complete, as the output term of the AM algebra had to be well-typed semantically. Therefore, they introduced two approximation algorithms. One of them took the unlabeled dependency tree as given and the other one presumed all dependencies were projective.

Non-projective parsing could be beneficial for managing reentrancy and arbitrary loops natively in AMR graphs. In this regard, Vilares and Gómez-Rodríguez [61] proposed a non-projective transition-based parser that worked in a left-to-right greedy manner. In their model, at every parsing setup, an oracle decides whether to make a concept or to link a pair of existing concepts. The single-head and acyclicity[1] constraints are not required in AMR, because arbitrary graphs are allowed. Thus, in their method, reentrancy and cycles were exploited in order to model semantic relations

[1] https://link.springer.com/chapter/10.1007/978-1-349-20568-4_1

among concepts and to detect co-references. In this respect, they could propose a natural way to deal with them by eliminating the constraints from the Covington transition system. Besides, AMR parsing needs words to transform into concepts, and dependency parsing performed on a constant-length sequence. However, in AMR, words can be eliminated or produced a single or several concepts. Hence, in their work, further transitions and lookup tables designed in order to generate concepts when necessary.

Wang et al. [62] represented a parser that formalized the AMR parsing procedure as a converting task from a dependency tree to an AMR graph. In addition, the perceptron algorithm was used to learn a linear model. In this PhD thesis, the author mentioned various properties of AMR parsing and accordingly introduced various algorithms to address them. In order to create the fundamental framework for graph parsing, Wang proposed a transition-based method that formalized AMR parsing as tree-to-graph transformation. Furthermore, different natural extensions to the parser explored, like inferring abstract concepts and exploring the richer feature space. Moreover, for handling the sparsity properties of AMR, the author applied a neural sequence labeling method to identify concepts. Afterwards, Wang introduced an automatic aligner that was far more suitable for the sentence-to-graph alignment plan. In addition, the author defined an end-to-end Neural AMR parser that explored the possibility of handling all AMR phenomenon by applying an integrated model. Eventually, He could extend all this on English AMR parsing to Chinese AMR corpus without notable modification.

Guo and Lu [63] defined a simple and efficient transition-based AMR. They operated the search in a purified search space using a compact AMR graph and an improved oracle. First, they introduced a compact representation for AMR graph. It made concepts and relations of an AMR graph easier and simplified the learning of the whole system. Then, based on that, they proposed a method to build the action sequence applied for training their model. The compact AMR graph included removing concepts and relations from an original AMR graph. For removing concepts, they divided AMR concepts into two groups: Lexical and Non-Lexical. Lexical concepts were extracted directly from tokens in the sentence, like lemmas, predicates with sense tags and tokens with quotation marks. Non-Lexical concepts were extracted by their child concepts, not directly from tokens in the sentence. Besides, for removing relations, they omitted specific relations in the original AMR graph in order to refine the search space. In addition, they designed some properties, which caused the compact graph to refine the search space more, such as Acyclicity, Simple, Non-terminal restricted and Reentrancy restricted. They achieved remarkable results for Named Entities, Concepts and Wikification. Additionally, they obtained acceptable results on reentrancy identification. However, they eliminated several reentrant edges during training phase. Their proposed compact AMR graph could encode significant information, but their model did not have good enough performance on predicating Negations. In this regard, one reason was that the parser was a word-level one, and it was not easy for it to detect morphological specifications, like prefixes «in», «un», «il» etc.

Naseem et al. [64] reinforced the Stack-LSTM AMR parser. They did this via fortifying training with self-critical Policy Learning and considering the sentence-level Smatch score as reward of sampled graphs. Besides, they merged some AMR-to-text alignments with an attention mechanism, and they applied named entities, pre-processed concept identification and contextualized embedding in order to fulfill the parser. Their proposed method was specifically suitable for AMR parsing, because it solved the problems arising from the lack of token-level AMR-to-text alignments. In addition, they applied different modifications that were inspired from neural MT.

Welch et al. [65] investigated the impact of using world knowledge in semantic parsing with AMR. They tried to find different types of errors of AMR parsers and tackle these errors by exploiting world knowledge to decrease them. The authors concentrated on three groups of knowledge from Wikipedia entity links, WordNet hypernyms and super senses, and retraining a named entity recognizer with the aim of detecting concepts in AMR. Their initial results showed that the retrained entity recognizer was not flawless and could not detect all concepts in AMR. Besides, they studied the limitations of the named entity features by using a set of oracles, which proved it could have positive impact on performance and condition that it can detect various subsets of AMR concepts. After examining the impact of various types of world knowledge for AMR semantic parsing, they investigated the upper bound on world knowledge by applying gold annotations and attempted to give modern visions about the world knowledge's potential in computational techniques to AMR parsing. Eventually, they proposed methods to improve the performance of AMR parsers. Specifically, they recognized that the combination of WordNet features named entities could lead to perfect results on almost all metrics, except Negation and SRL. Moreover, they tried to exploit other types of world knowledge, such as word embedding and node embedding, and it did not result in expected improvements. This presented that following research methods in AMR parsing should concentrate on improvements in training data or parsing approaches.

Gu et al. [66] designed a general AMR parsing model that employed a two-stack-based transition algorithm. They applied the model on Chinese and English datasets.

The model could parse the input sentences to AMR graphs in linear time, incrementally. The main approach of their paper was based on the arc-eager algorithm. It used heuristic search method for concept recognition. In order to modify the transition algorithm to suit AMR transformation, they introduced a two-stack-based transition algorithm to solve the problem in AMR parsing. This algorithm was an extended Shift/Reduce decoding algorithm based on two stacks. They applied more appropriate feature representation to enrich feature representation learning in the prediction of transition actions. As shown in Fig. 13, their proposed model divided into 6 phases.

In Fig. 13, the main aim of AMR concept annotation, named-entity recognition and pre-training phases was pre-processing of their method. According to external datasets, they trained word embedding using Word2Vec model as the input of this model. The AMR concept annotation, in phase 1, aligned words to concepts and builds an alignment table through an aligner, which is the input for the concept recognition phase. Additionally, they utilize Corenlp[1] to label named entities in dataset as features for the concept recognition module and transition-based AMR parsing module.

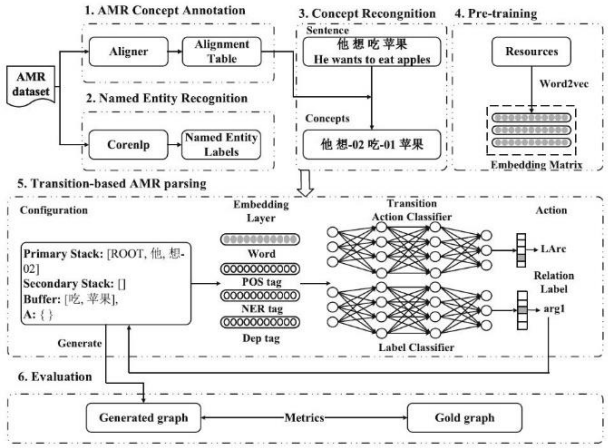

Figure 13. General automatic semantic parsing model from [66].

Zhou et al. [67] introduced a transition-based model which, incorporated hard-attention over sentences with a target-side action pointer mechanism to decouple source tokens from node representations and address alignments. They constructed the transitions and the pointer mechanism through straightforward modifications within a single Transformer architecture. Parser state and graph structure information were effectively encoded using attention heads. In other words, they proposed an Action-Pointer mechanism that could simply manage the generation of arbitrary graph constructs, such as multiple nodes per token and reentrancy. Their structural modeling with incremental encoding of parser and graph states based on a single Transformer architecture achieved the best results on all AMR corpora among previous methods with resembling learnable parameter sizes.

D. Sequence-to-sequence-based method

These methods, which are also called MT (Machine Translation) methods, generate the textual format for an AMR graph, straightly from the system input by sequence-to-sequence neural methods. Commonly, it needs some pre-/post-processing for simplifying the task.

Due to the increasing applications of deep learning methods in NLP, researchers tried to use deep learning algorithms for AMR parsing. As one of the most efficient examples of them, neural networks have been widely used in this field. The notable achievements of neural network in computer vision and speech recognition field could attract many scientists to try to reach similar results in NLP branches, too.

Barzdins and Gosko [68] for the first time used the sequence-to-sequence method for neural machine translation. Actually, they treated preorder traversal of AMR as foreign language strings and in this way, they did AMR (PENMAN notation) parsing. Meanwhile, they presented 2 extensions to the AMR Smatch scoring script. The first one combined the Smatch scoring script with the C6.0 rule-based classifier to produce a human-readable report on the error patterns frequency observed in the scored AMR graphs. The second one combined a per-sentence Smatch with an ensemble method for selecting the best AMR graph among the set of AMR graphs for the same sentence.

Reciprocally, Konstas et al. [69] addressed the sparsity issue by a self-training approach that employed a huge unannotated external corpus set. They defined a paired training procedure for improving both the text-to-AMR parser and AMR-to-text generator. They first applied self-training to bootstrap a high-quality AMR parser from millions of unlabeled Gigaword sentences, afterwards applied the automatically parsed AMR graphs to pre-train an AMR generator. This paired training allowed both the parser and generator to learn accurate representations of fluent English text from weakly labeled examples, which were then fine-tuned by human annotated AMR data. Furthermore, they introduced a pre-processing procedure for the AMR graphs, that contained anonymizing entities and dates, grouping entity categories, and encoding nesting information in brief ways. Their pre-processing procedure boosted handling the data sparsity as well as substantially reducing the complexity of the AMR graphs. In this regard, they presented that any depth first traversal of the AMR is an effective linearization, and it is even possible to use a different random order for each example.

[1] https://stanfordnlp.github.io/CoreNLP/

Foland and Martin [45] broke down the task of AMR parsing into some distinct subtasks and did each subtask using Bi-directional LSTM. Their system received an input sentence in form of a vector of word embeddings and applied a series of recurrent neural networks to 1) identify the basic set of nodes and subgraphs that comprised the AMR, 2) detect the set of predicate-argument relations among those concepts, and 3) discover any relevant modifier relations that were present. As shown in Fig. 14, the parser extracted features from the sentence that were processed by a bidirectional LSTM network (B-LSTM) in order to create a set of AMR subgraphs, that included 1 or 2 concepts, besides, their internal relations to each other. Afterwards, features were processed by a pair of B-LSTM networks based on the sentence and these subgraphs in order to compute the probabilities of relations between all subgraphs. Then, all subgraphs were connected by applying an iterative, greedy algorithm to compute a single component graph, with all subgraphs connected by relations. Separately, another two B-LSTM networks computed attribute and name categories, that were then appended to the graph. Eventually, the subgraphs were expanded into the most probable AMR concept and relation primitives to produce the final AMR.

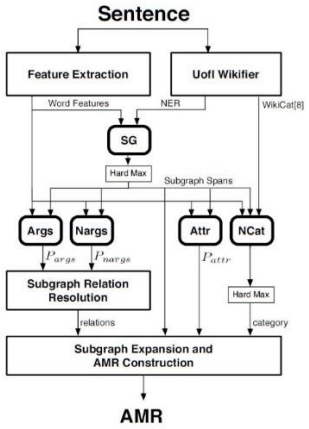

Figure 14. A high-level block diagram of the parser [45].

Since transition-based parsing methods are efficient and simple, it is very absorbing to incorporate it with neural approach. In this regard, Buys and Blunsom [70] introduced a neural encoder-decoder transition-based system in order to parse semantic graphs. They used the sequence-to-sequence framework for learning the conversion from natural language to action sequences. Their proposed system was the first full-coverage semantic graph parser for Minimal Recursion Semantics (MRS). The model architecture applied stack-based embedding features, predicting graphs jointly with unlexicalized predicates and their token alignments. The system included a stack of graph nodes being processed and a buffer, holding a single node at a time. Plus, the main transition actions were shift, reduce, left-arc, right-arc. The action taken at each step is given, along with the state of the stack and buffer after the action is applied, and any arcs added. Shift transitions generate the alignments and predicates of the nodes placed on the buffer. Items on the stack and buffer have the form (node index, alignment, predicate label), and arcs are of the form (head index, argument label, dependent index). Semantic representation of the sentence *Everybody wants to meet John* is illustrated in Fig. 15. The graph is based on the Elementary Dependency Structure (EDS) representation of MRS. The alignments are given together with the corresponding tokens, and lemmas of surface predicates and constants. Table 4 demonstrates an example transition sequence together with the stack and buffer after each step.

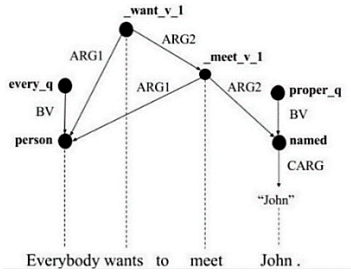

Figure 15. Semantic representation of an example sentence [70].

TABLE 4
AN EXAMPLE TRANSITION SEQUENCE RELATED TO REPRESENTATION IN FIG. 15 [70].

Action	Stack	Buffer	Arc added
Init(1, person)	[]	(1, 1, person)	-
Sh(1, every_q)	[(1, 1, person)]	(2, 1, every_q)	-
La(BV)	[(1, 1, person)]	(2, 1, every_q)	(2, BV, 1)
Sh(2, _v_1)	[(1, 1, person), (2, 1, every_q)]	(2, 1, _v_1)	-
re	[(1, 1, person)]	(3, 2, _v_1)	-
La(ARG1)	[(1, 1, person)]	(3, 2, _v_1)	(3, ARG1, 1)

In a similar way, Misra and Artzi [71] considered the CCG AMR parsing method, as a shift-reduce transition system. Besides, they learned it with a neural network algorithm. Their proposed parser exploited a neural network architecture that balances model capacity and computational cost. They trained by transferring a model from a computationally expensive log-linear CKY parser. Their learner tackled 2 challenges: 1) selecting the best parse for learning when the CKY parser generates multiple correct trees, 2) learning from partial derivations when the CKY parser fails to parse.

Van Noord and Bos [72] investigated three different techniques for dealing with co-indexed variables in the output of neural semantic parsing of AMRs. They proposed them to address reentrancy for AMR parsing:

1) Copying concepts within training and restoring co-indexation in a post-processing phase; 2) Indexing of co-indexation explicitly; 3) Applying absolute paths to designate co-indexing. In general, their introduced technique exploited a character-based sequence-to-sequence model in order to translate sentences to AMRs, also included pre-processing and post-processing phases. Their main goal was to detect the best techniques to manage reentrancy in neural semantic parsing and to illustrate the exact influence that each of the techniques had on the whole performance.

Zhang et al. [73] designed an attention-based approach, which considered AMR parsing as sequence-to-graph transduction. Most of AMR parsers apply pre-trained aligners, data augmentation and external semantic resources, however, their suggested model was aligner-free, and it could be trained efficiently using limited amounts of labeled AMR data. In addition, one of the main goals of their proposed model, supported by an extended pointer-generator network, was to manage reentrancy. It had two primary modules: Node and Edge prediction, which used an extended Pointer-Generator Network and a Deep Biaffine Classifier, respectively.

As it is presented in Fig. 16, the first module included four main components: an encoder embedding layer, an encoder, a decoder embedding layer, and a decoder. For each decoding time-step, three probabilities $p_{tgt}. p_{src}$ and p_{gen}were calculated. The source, target and vocabulary attention distributions were weighted respectively by the three probabilities, and after that they summed to reach the final distribution, and then make the ultimate prediction.

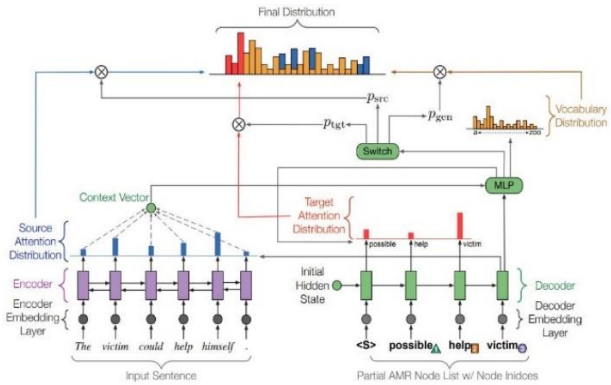

Figure 16. Node prediction [73].

Regarding the second module as shown in Fig. 17, unlike most methods, which re-encode AMR nodes, they applied decoder-hidden states directly from the network in the first module as the input to the Deep Biaffine Classifier [74].

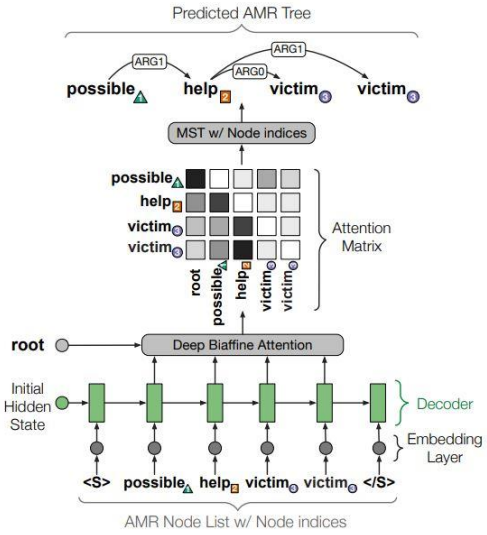

Figure 17. Edge prediction [73].

In Fig. 16 and 17, exploiting decoder-hidden states as input has two main advantages as follows:

1) In the input-feeding strategy, hidden states of the decoder included contextualized information from both the input sentence and the predicted nodes.
2) As decoder-hidden states were exploited for both node and edge prediction, in this approach the two modules can be jointly trained.

Bai et al. [75] studied graph self-supervised training in order to enhance the structure awareness level of pre-trained language models over AMR graphs. Specifically, they introduced two graph auto-encoding approaches for graph-to-graph pre-training and four methods for integrating text and graph information within pre-training. Afterwards, they introduced a unified model to fill the gap between pre-training and fine-tuning applications. In this regard, they exploited BART [76] as the primary sequence-to-sequence method, and proposed graph pre-training approaches and a unified pre-training model for both AMR parsing and AMR-to-text generation. This approach led to pre-training on AMR structures exploiting BART. In brief, the authors defined four auxiliary pre-training tasks in order to enhance the rate of information exchange between text and graphs as follows:

1) $tg2\hat{t}$: Graph-augmented text de-noising in which an AMR graph was considered as supplementary input to boost the reconstruction of masked text.
2) $tg2\hat{g}$: Text-augmented graph de-noising in which text boosted the reconstruction of masked graph.
3) $\hat{t}g2\hat{t}$: Noisy graph-augmented text de-noising in which the output text was generated according to a pair of masked graph and masked text.

4) $\hat{t}g2\hat{g}$: Noisy text-augmented graph de-noising in which an output graph was generated according to a pair of masked graph and masked text.

Besides, in their proposed model, fine-tuning tasks meant that there was an empty graph/text in the primary input, which led to an input format of $\bar{t}g2g$ for AMR-to-text generation and $t\bar{g}2g$ for AMR parsing. Using this model, pre-training and fine-tuning tasks shared the same input format, thereby facilitating knowledge transfer from pre-training to fine-tuning.

E. Conversion-based and other methods

In Conversion-based methods, the output of some semantic parsers is converted to AMR graphs. Conversion-based approach comprises transforming dependency trees to AMR [53] and transforming parsers' outputs in the form of other meaning representations to AMR. These parsers have designed for logical form of Microsoft and the Treebank semantics corpus and Boxer's discourse representation structures.

As an example of remaining methods, Pust et al. [77] tried to create AMRs via a statistical MT system. They treated English-to-AMR conversion within the string-to-tree, Syntax-Based Machine Translation (SBMT) framework. In this regard, they transformed the AMR structure into a suitable form for the SBMT. Besides, they defined an AMR-specific language model and added data and features extracted from semantic resources.

Lyu and Titov [78] proposed a neural parser that considered alignments as latent variables in a joint probabilistic concepts model, alignments and relations. For a careful deduction, they needed to marginalize over alignments, which was infeasible. Therefore, they applied the variation auto-encoding framework, besides a continuous relaxation of the discrete alignments. Besides, they proofed that joint modeling was better than a pipeline of align and parse.

One of the long-standing topics in natural language processing is semantic parsing. This has been used for meaning representation in basic NLU applications, including shallow rearrangements of text and deep meaning representations based on first order logic or other types of it. The mentioned systems applied meaning analyzers, which were designed for restricted domains. In similar deep meaning analyzers, the level of complexity was also increased. The open domain examples of these meaning analyzers have been designed too. Sometimes, they used a statistical parser to construct the required analysis.

Applying supervised learning algorithms on semantic parsers, started for restricted domains using shallow and then deep meaning representations, however, we also could find some earlier research on learning meaning parsers in the literature.

For the first time, Gildea and Jurafsky [79] suggested using supervised learning algorithms for open domain semantic parsers. Their work advocated structure of shallow meaning parsers, and their focus was on the semantic role labeling task and dependency parsing of these parsers. Their effort followed by other researchers to reach the goal of learning deep meaning parsers. Therefore, they could learn larger deep parsers, which still were restricted domain, using indirect supervising methods. Furthermore, their attempts were made to achieve the goal of designing robust open domain deep semantic parsers. These learning processes were done using partial annotation, unsupervised learning algorithms and indirect supervising methods.

Klimeš [80-82] did some successive research for learning an open domain semantic parser with supervised ML algorithms, which were applied on the Prague Dependency Treebank (PDT). This procedure also called deep syntactic analysis and the PDT's tec-to-grammatical layer could be assumed as a deep meaning representation. Interestingly, the representation had many similarities with AMR, although, it had some different handling methods for different semantic forms.

IV. AMR GENERATION

There has been several research for statistical/non-statistical NLG from various input representations, which in the following some of the most important ones will be introduced.

Scientists have recognized that the generation process includes three different problems as follows:

1) Text planning: Planning what to express to reach a communicative goal.
2) Sentence planning: Splitting information into sentences and concepts.
3) Surface realization: Declaring concepts in natural language.

Overall, open domain Deep Semantic Generators (DSGs) have been designed using hand-written rules, and restricted domain DSGs have been learned with supervised learning algorithms. Besides, open domain DSGs have been learned by applying supervised learning techniques, and several hand-written rule-based generators have been constructed using statistical methods. However, AMR generators are the first open domain DSGs, which were learned by applying supervised learning algorithms, to the best of our knowledge.

Prime generators used hand-crafted grammars which have been designed using these hand-crafted grammars, for both open and restricted domains approaches. In these systems, the input characteristics were very diverse, such as templates, deep syntax, abstract syntactic trees, and deep meaning representations.

Examples of new surface recognizers with hand-crafted grammars are KPML, PENMAN, RealPro and FUF/SURGE. However, these are general purpose systems, they may need to have fine lexical and

grammatical resources to tune them to a specific domain [83], besides, indeed they are not open domain in practice. Open domain generators, which use hand-crafted grammars with great lexical databases, already exist like SimpleNLG. This generator maps from a slightly lower specification level compared with other ones, such as generators in the MT systems ETAP-3 and TectoMT, that map from deep syntax or UNL to English and Russian or deep syntax to English and Czech (TectoMT).

The early effort applied ML for generation step and merged rule-based generators with statistical approaches. These systems used a rule-based generator, which generated too much, and using an N-gram language model. They finally produced the suitable output [84]. These systems generate the output from abstract syntax trees or from deep meaning representations.

The initial studies, which tried to learn surface recognition from supervised training instances, could map from surface semantics to natural language in restricted domain cases. Each of them learned to map from feature tuples to templates, but the first system that managed to learn a mapping from a deep meaning representation to natural language was done by Wong and Mooney [85]. Their system worked on restricted domains such as RoboCup and GeoQuery, and there is important research, which followed up on them. These systems comprise enough small lexical databases, like KPML that have around 1000 lexical entries. It should be noted that open domain systems comprise many more entries, for example, ETAP-3 has more than 300,000 entries and SimpleNLG includes over 65,000 lexical entries. Furthermore, some works using supervised techniques could learn open domain generators from deep and shallow syntax.

There exists a wide range of research for statistical and non-statistical NLG using different input representation types. On one hand, research like Belz et al. [86] used a deep syntax representation that had some similarities with AMR. For example, these representations were graphs like AMR, with reentrances, and they had a concept inventory from of PropBank. Plus, they concentrated on sentence representation too. On the other hand, the Nitrogen system [84] and the Halogen system [87] both suggested using an input representation, which was a prelude to the modern AMR. Interestingly, they were also named it AMR, however, it was not similar to AMR.

As mentioned before, the methods that have been used in statistical machine translation were also applied on NLG problems. Besides, several grammar-based techniques could be considered as weighted tree-to-string transducers. In this regard, Jones et al. [38] proposed a methodology for translation and generation using SHRG. They used the GeoQuery corpus in their research. Actually, this method can be used for AMR generation, too.

Since AMR is adequately suited for English language, the AMR to English generation task could similarly consider as related NLG issues, like bag generation, generation from logical form or restoring order to unordered dependency trees. Undoubtedly, the deep logic of AMR created a significant challenge for English understanding. Here, it should be noted that another important feature of AMR is its abstraction from language details like time and number.

Flanigan et al. [88] introduced a trained AMR to English generator. In their research, spanning trees generated from AMR graphs and then tree-to-string transducers applied to these trees to finally generated English language. Their generator, to our knowledge, was the first AMR generator in the literature, and was the first open domain generator from a deep meaning representation that was used supervised algorithms for learning.

In addition, Pelja Paul et al. [89] proposed an automatic tool for generating text from AMR. They introduced an editor for AMR which was used in order to generate semantic representation for simple sentences. In this regard, they applied dependency parser and followed the stages: tokenization and POS tagging, dependency parsing, dropping articles (like plurality), focus identification and arguments identification.

We classified the existing AMR generation approaches in six main categories, which explained their related works as follow sections.

A. Tree-transducer-based method

Two main systems have applied these methods for AMR generation, up to now. One of them introduced by Flanigan et al. [88] and the other system proposed by Gruzitis et al. [52]. The latter won the first place in the human evaluation section of SemEval 2017. It incorporated the JAMR system and a handwritten rule-based generator. In that research, AMR graphs transformed to abstract syntax trees. Besides, the Grammatical Framework (GF) generator applied to generate English texts. In their system, if the GF generator could not generate the intended result, the JAMR output would be the alternative plan. This alternative method used in 88 percent of cases in the SemEval 2017 test set.

B. Graph-grammar-based method

Song et al. [90] suggested exploiting a Synchronous Node Replacement Grammar (SNRG) for generating from AMRs. SNRGs are equivalent to Synchronous CFGs (SCFGs), although one side of it is a graph grammar. In their research, they incorporated a SNRG and a feature-based statistical model, including a language model for generating natural language. An

example results of their system for the sentence *now you understand how people like tmt in Fairfax feel* is illustrated in Fig. 18. It can be seen that, in general, in most cases the semantics of the input AMR are correctly translated, like *example*, which is synonymous with *such as*, and *thing*, which is an abstract concept and should not be translated, however, there are a few errors, like *that* which in the result should be *what*, and there should be an *in* between *tmt* and *fairfax*.

(u / understand-01 :ARG0 (y / you) :ARG1 (t2 / thing :ARG1-of (f2 / feel-01 :ARG0 (p2 / person :example (p / person :wiki - :name (t / name :op1 "TMT") :location (c / city :wiki "Fairfax,_Virginia" :name (f / name :op1 "Fairfax")))))) :time (n / now))
Trans: now, you have to understand that people feel about such as tmt fairfax
Ref: now you understand how people like tmt in fairfax feel .

Figure 18. Generation example [90].

C. Graph-based method

Song et al. [91] considered the AMR generation task as an Asymmetric Generalized Traveling Salesman Problem (AGTSP), wherein AGTSP nodes are equivalent with rules for mapping graph fragments of AMR to strings. In each AGTSP graph traversal, the AMR graph covered, and the generated strings ordered based on the orders of the AGTSP nodes that have visited. This is like considering the phrase-based MT as a traveling salesman issue; however, it is generalized to graph-to-string generation task. They also analyzed their model and JAMR-gen[1] with an example AMR and show the AMR representation, the reference, and results in Fig. 19. Firstly, both their model and JAMR-gen outputted an acceptable translation containing most of the meaning from the AMR. Although, their model failed to detect *boy* as the subject, as the feature set does not contain edge labels, like ARG0 and ARG1. Eventually, neither their model nor JAMR-gen could work properly in the situation when a re-entrance node (like *b*/*boy*) required to be translated twice.

(w / want-01 :ARG0 (b / boy) :ARG1 (b2 / believe-01 :ARG0 (g / girl) :ARG1 b))
Ref: the boy wants the girl to believe him
All: a girl wanted to believe him
JAMR-gen: boys want the girl to believe

Figure 19. Generation example [91].

D. Sequence-to-sequence-based method

Sequence-to-sequence methods have been used for AMR generation a lot. Pourdamghani et al. [92] designed phrase-based MT on linearized AMR graphs, to try three separate linearization methods. Remarkably, they could achieve higher BLEU score in comparison with JAMR, however, they have been judged notably worse in the human evaluation section of SemEval 2017. This represented that BLEU score is not suitable for comparing, although it has been widely used in AMR generation applications. It should be noted that in some research BLEU score was seen to be weakly correlated with this score for several generation tasks.

Konstas et al. [69] as mentioned before used sequence-to-sequence neural MT to linearized AMR graphs. They could reach significant progress as compared with previous methods when they employed pseudo-gold standard AMR graphs, that have been derived from performing usual AMR parsers on English Giga-word corpus called LDC2011T07.

Recently, Fan and Gardent [93], unlike most related papers that worked only on generating into English, they focused on leverage advances in pre-training, cross-lingual embedding, and multilingual models in order to build multilingual AMR-to-text models which could generate in 21 different languages. Their results show that for 18 natural languages, their proposed models could outperform baselines that generated into a single natural language. Besides, they analyzed the performance of the models in accurately capturing morphology and word order using human agent evaluation, and the results proved that native speakers considered the output generations to be fluent. In their research, in order to generate from AMRs, they applied neural sequence-to-sequence methods that modeled the input AMR with a Transformer encoder and generated natural language with a Transformer decoder. In their model, as shown in Fig. 20, for all natural languages, the input was an English-centric AMR that derived automatically using the JAMR semantic parser from English text. They pre-trained both the AMR encoder and the multilingual decoder and they used cross-lingual embedding.

[1] https://github.com/jflanigan/jamr/tree/Generator

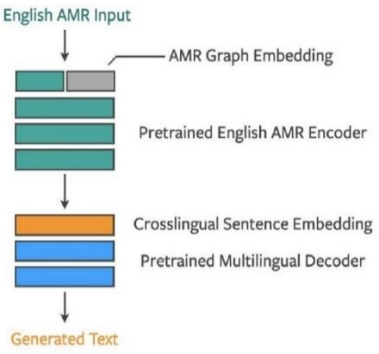

Figure 20. One-to-Many schema for multilingual AMR generation [93].

E. Rule-based method

Up to now, two main rule-based AMR generators have been proposed. Gruzitis et al. [52] represented a system wherein AMR graphs were transformed to abstract syntax trees. Then, the Grammatical Framework (GF) generator is applied to finally generate English. In their system, if the GF generator could not give an appropriate output, the JAMR output would be exploited.

Mille et al. [94] proposed a system which could convert AMR graphs to surface-syntactic structures via applying a series of rule-based graph transducers. The main approach and strategy of their proposed generator was graph transduction of grammars converted, in various steps, abstract AMRs into syntactic structures which included all the morphological characteristics required for retrieving the final forms of all the words. After that the structures from previous step were linearized by a prevalent linearizer that comes from the first surface perception shared task, and eventually the final forms of the words were retrieved. In general, their generator follows the theoretical model of the Meaning-Text Theory (MTT); the names of the intermediate layers mentioned in Table 5 are from the MTT terminology.

TABLE 5
OVERVIEW OF THE AMR-TO-TEXT PIPELINE [94].

	Step	Layer	#rul.
0	Conversion of AMRs format into CoNLL'09 format	ConS	N/A
1	Mapping of AMRs onto predicate-argument graphs	SemS	190
2	Assignment of parts of speech	$SemS_{pos}$	96
3	Derivation of deep syntactic structure	$DSyntS$	267
4	Introduction of function words	SSuntS	294
5	Resolution of agreements	DMorphS	85
6	Linearization	SMorphS	N/A
7	Retrieval of surface forms	Text	1
8	Post-processing	$Text_{final}$	4

F. Transition-based and other methods

These methods consider the generation issue as a sequence of actions that selected by supervised ML algorithms.

Lampouras and Vlachos [95] introduced a system in which AMR graphs were converted by employing a series of actions on syntactic dependency trees or tree structures like dependency trees and in the second step of the process they would be linearized. They designed a three-phase transition-based system for transforming an AMR graph to a dependency tree, in which the final sentence could then be acquired via a tree linearizer. An example transition from AMR graph to dependency tree of the sentence *It makes one tremble even without fear* is shown in Fig. 21.

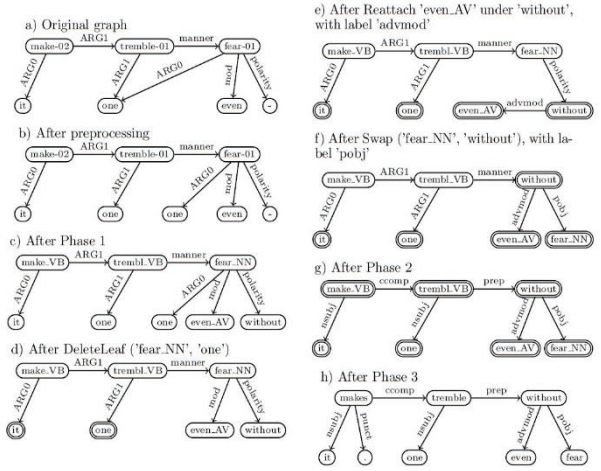

Figure 21. Example transition from AMR graph to dependency tree [95].

Schick [96] proposed a set of actions (transitions) like merging, deletion and swapping of edges and vertices. After applying these transitions to the input, the obtained tree structure was turned into a sentence by visiting its vertices in a specific order. They embedded various kinds of required actions into a transition system. To predict the correct sequence of transitions to be applied for each input, they trained maximum entropy models from a corpus of AMR graphs and corresponding realizations.

Jin and Gildea [97] applied Transformer self-attention on graphs in order to allow global feature propagation. They trained a model to learn shortest paths by exploiting generalized shortest-paths algorithms, instead of feeding them into the vertex self-attention module. Their proposed technique could broaden the receptive field of a graph encoder by subjecting it to all probable graph paths. Then, the authors investigated how this path diversity influences the performance across levels of AMR connectivity, indicating gains on AMRs of higher reentrancy counts and diameters. Their analysis showed that the generated sentences had high semantic coherence for reentrant AMRs. In brief, the first advantage of the proposed method is that there is no need to pre-compute

shortest paths. Second and third ones are that more than one path considered per vertex pair by relation encoder and graph paths selected according to properties in addition to distance. The third advantage is very beneficial because shortest paths may over-simplify the input structure of more densely connected AMRs. Graph instances in which a node has more than one entering edge or instances that have reentrancies, could include competing shortest paths between a node pair. Eliminating either path between the two concepts can alter the AMR's semantics. Besides, shortest paths can insufficiently represent the relation between concept pairs. Their proposed relation encoder solved both of these problems as it remarked all graph paths. In general, they designed two relation encoders using generalized shortest-paths methods for graph encoders.

Ballesteros et al. [98] tried to generate natural language from a deep syntactic representation, which did not have the equivalent number of nodes with the surface syntax. This is the common problem that also should be solved when the system is going to generate from deep meaning representations like AMR. In their systems tree-transducers used in the generation process. They suggested applying a sequence of SVM classifiers for transduction purpose, instead of using a weighted tree-transducer. The data-driven generator was defined as a tree transducer framework that included a cascade of 6 data-driven small tasks. The first 4 tasks captured the actions 1.–4. the 5th linearized the obtained surface-syntactic structures. Fig. 22 provides a sample input and output of each sub-module. The system outputs a 14 column CoNLL'09 linearized format without morphological punctuation or inflections marks.

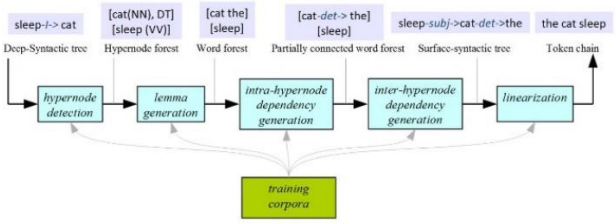

Figure 22. Workflow of the Data-Driven Generator [98].

V. DISCUSSION

In this section, AMR-related evaluation methods including metrics and datasets, AMR challenges and some main features of AMR in the form of some questions are described based on the papers that are studied in this paper.

A. AMR Datasets

One of the former datasets for AMR parsing was LDC2016E25 dataset. LDC released this corpus of AMRs in 2016, and it was a part of the DARPA DEFT program. LDC2016E25 contained 36521, 1368 and 1371 sentences in its train, development and test sets, respectively.

The most used version of the corpus was annotated by teams at LDC, SDL, and the Colorado University, is the expanded form of the previous releases (LDC2014T12, LDC2015E86, LDC2014E41 and LDC2017T10). It contains 39,260 sentences including 19,572, 18,779, 13,051 AMRs from LDC2015E86, LDC2014E41 and LDC2014T12, partitioned into train, Development and test parts, token from various news and discussion forum sources [99]. People, who participated in the generation process, just had AMRs of the additional 1,293 sentences for evaluation task. In addition, the original sentences provided, if required, to evaluators during the human evaluation step.

The most recent version of these datasets is LDC2020T02. The source data of it consists of discussion forums collected for the DARPA BOLT and DEFT programs, transcripts and translations into English of Mandarin Chinese broadcast news programming from China central television, Wall Street Journal (WSJ) text, translated Xinhua news texts, various newswire data from NIST OpenMT evaluations and weblog data that used in the DARPA GALE program.

Table 6 shows the number of trainings, Developments (Dev), and tests as well as the total number of them for each related dataset.

TABLE 6
THE NUMBER OF TRAINING, DEV AND TEST FOR DATASETS [100].

Dataset	Training	Dev	Test	Totals
BOLT DF MT	1061	133	133	1327
Broadcast conversation	214	0	0	214
Weblog and WSJ	0	100	100	200
BOLT DF English	7379	210	229	7818
DEFT DF English	32915	0	0	32195
Aesop fables	49	0	0	49
Guidelines AMRs	970	0	0	970
LORELEI	4441	354	527	5322
2009 Open MT	204	0	0	204
Proxy reports	6603	826	823	8252
Weblog	866	0	0	866
Wikipedia	192	0	0	192
Xinhua MT	741	99	86	926
Total	55635	1722	1898	59255

There is a smaller dataset in the biomedical domain called Bio AMR Corpus[1], which includes annotations of cancer-related PubMed[2] articles, and the result sections of 46 additional PubMed papers. Also, it contains about 1000 sentences from the BEL Bio Creative training corpus and the Chicago Corpus. The Bio AMR corpus divided into 3 parts: dev, training, test, which have 500, 5452 and 500 sentences, respectively.

B. AMR Evaluation

In the following, the AMR metrics and an analysis of parsers and generators methods are provided.

1) Metrics

There is not any widely used evaluation metric for the whole sentence semantic structures, but there are two main criteria for evaluating the existing whole-sentence semantic parsers [101]:

1. Correctness: It evaluates the performance of an NLP task that uses the parsing results.
2. Accuracy: It counts number of sentences that has been parsed completely and correctly.

According to these descriptions, it is necessary to design evaluation methods, which exploit scores that range from 0 to 1 called partial credit for measuring the whole-sentence semantic structures. Using these methods, researchers would be able to distinguish between two similar structures, without insisting on specific domains or tasks.

It is possible to encode AMR graphs in the form of a conjunction of pairs as [variable1, variable2]. Therefore, considering two AMRs, it would be easy to calculate recall, precision, and their combination F1 score, via counting the number of matched pairs. It should be noted that, as AMR does not have inherent alignment among variables, there could be large numbers of probable matches. In this regard, Smatch [15] was introduced to solve this problem. This metric detects the highest F1 score as calculated by (1), which is attainable through a one-to-one matching process of variables in two considered AMRs. Therefore, it can be said from one point of view that the issue become an NP-hard problem, and Smatch metric employs the known hill-climbing approach in order to reach the approximate inference.

As Fig. 23 illustrates, the value of F1 score calculated using (1) for sentences *The boy wants the book* and *The boy wants to go*[3] is 0.727. In (1), *N* is the number of matched equivalence concepts and edges (precision × recall), which is 4, and the total number of concepts and edges between two AMRs are c_1 = 6 and c_2 = 5 respectively.

$$F1\ score = \frac{2 \times N}{c_1 + c_2} = \frac{8}{6+5} = 0.727 \quad (1)$$

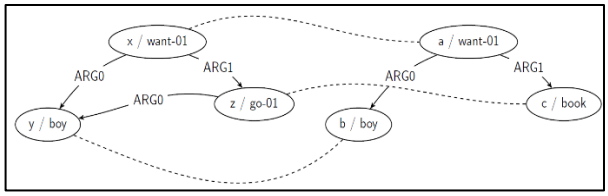

Figure 23. AMR graphs for the two mentioned sentences.

In general, Smatch [15] calculates the two AMR graphs score in terms of their matching edges (triples) by detecting a node (variable) mapping which maximizes the count as (2):

$$Smatch = F1\ score = \frac{2 \times (P \times R)}{(P+R)} \quad (2)$$

Where $P = M/T$and $R = M/G$ is precision and recall measure, respectively. M is the number of matching triples, T is the total number of triples in the first AMR, G is the total number of triples in the second AMR.

On the other hand, the most used method for evaluating generators is applying the Bilingual Evaluation Understudy (BLEU) score [102] that came from MT. Considering a candidate sentence w and a reference sentence; $\widehat{w}$, the primary idea of BLEU is to count the number of matching *N*-grams between w and $\widehat{w}$. Then, it is divided by the total number of *N*-grams in the candidate sentence; w. Usually, this computation is done not just for one but for several values of *N* (for example n = 1, ..., 4) and the results are averaged subsequently. Like Smatch, the BLEU metric ranges from 0 to 1, however, in this paper the values of both metrics multiplied to 100 for simplicity.

Allen et al. [103] introduced a metric that computed the maximum score using any alignment between LF (Logical Form) graphs; however, they did not explain how to find these alignments. Dridan and Oepen [104] used an approach for directly evaluating the meaning representation. This approach assessed the output of a semantic parser by comparing semantic sub-structures, albeit they needed a suitable alignment between semantic sub-structures and sentence spans. On the other hand, Smatch as an evaluation metric, does not need any alignment between the input sentence and its meaning analysis.

Opitz et al. [105] designed criteria which allows to apply a principled evaluation of criteria comparing meaning representations such as AMR. They did a detailed analysis of Smatch and SEMBLEU metrics, in which they illustrated that the latter displayed some unpleasant attributes. For instance, it does not tend to detect the undistinguished rule and performed biases which were not easy to control. In addition, they

[1] This corpus is accessible from https://amr.isi.edu/download/2018-01-25/amr-release-bio-v3.0.txt

[2] https://pubmed.ncbi.nlm.nih.gov/

[3] Example is borrowed from https://github.com/nschneid/amr-tutorial/tree/master/slides

introduced a metric S^2match which had weakness in facing with very scarce semantic deviations and its goal was to cover almost all existing aspects. Besides, they proved its superiority over Smatch and SEMBLEU, as the Smatch strength was in alignment-search between the variables of graphs and the SEMBLEU criterion worked on the ground of a variable-free AMR, which converted it to a bag of k-grams. As they pointed, circumventing a variable alignment search decreases computational cost and warrants complete determinacy. Plus, there is a tendency in implementing the metric based on BLEU, because BLEU is highly well-known in MT. Although, they recognized there was a lack of principled in-depth analogy among attributes of different AMR criteria which would help in choosing suitable metrics that should be applied in each AMR-related task. Moreover, as other metrics it does not measure graded semantic differences, they investigated a new feature in this regard. The semantic differences could appear because of near synonyms like *ruin*/*annihilate*; *thin*/*skinny*/*slim*; *enemy*/*foe* or paraphrases like *can*/*be able to*; *unnecessary*/*not necessary*. It is noteworthy that in a classical syntactic parsing task, criteria did not require to tackle this problem, as input tokens are typically projected to lexical concepts by lemmatization. Thus, two graphs for the same sentence tend to agree on the concepts projected from the input. However, this could be different in semantic parsing in which the projected concepts are usually more abstract.

2) Analysis of parsers and generators methods

We presented the detailed features and evaluation (fine-grained F1 scores) of different AMR parsing methods based on Smatch score in Table 7. This metric is calculated on the predicted graphs without considering any edge label. It shows whether world knowledge results in improving performance on graph structure prediction.

In Table 7; *No WSD*[1] indicates the number of Word Senses appeared after concepts (e.g. both *open*-01 and *open*-02 would be considered as *open*), *Named Entities* are the value that only inspects whether the named entity concepts are correct, *Negations* are the polarity edges of the graph and calculate the correctness of negated concepts, *Concepts* are only concept labels; not edge labels, *Reentrancy* is reentrant edges in the AMR graph, *SRL*[2] is the number of edge labels and *Wikification* is the number of wikifications in the AMR graph.

Table 7 indicates that the AMR model can encode significant information, even though, some of the recent models did not work good-enough in predicating Negations. Possibly, the main reason is that these parsers methods were word-level ones, and it was complicated for them to detect morphological features like prefixes «in», «ir», «un», etc. Thus, it expected to achieve higher performance if the character-based representations applied instead or in combination with previous methods.

Table 8 presents the features of various parsers methods developed on various datasets. From Table 8, it can be concluded that almost all methods either had an approximately high complexity or applied a pipeline methodology. Moreover, in general, seq2seq models approached the AMR graph in an end-to-end way and needed less features. Although often they faced the data sparsity problem, therefore, they used external corpora.

Table 9 presents the evaluation of different AMR parsers on some existing and most used AMR-related datasets based on F1 scores.

Table 10 presents the evaluation result of some of the main generating methods which are discussed in this paper according to Bleu metric.

It is noteworthy that several AMR corpuses in non-English languages, such as Chinese [107], has been produced and exploited in various NLP tasks.

C. AMR challenges

As mentioned in previous sections, AMR relies on PropBank verbal propositions and their arguments, guaranteeing all such semantic structures could be represented. The main purpose of AMR is to strictly abstract meaning. The AMR model abstracts away from morpho-syntactic idiosyncrasies, like word order or word category, and does not consider tense and number, in order to have a short annotation process and acquire concise meaning representations. Besides, most prepositions and articles, which called function words are eliminated, also it would not have universal quantifiers. It should be noted that, as AMR model is heavily dependent on Propbank framesets.

AMR parsing have some primary challenges: abstraction, reentrancy data sparsity, and graph representation. The last one related to this principle that AMR is a non-tree graph formally, so new methods are required to parse input sentences into desired graphs. Wang [19] tackled this problem using a transition-based algorithm, which creates AMR graphs from dependency trees, incrementally.

Abstraction challenge refers to this fact: AMR does not prepare any alignments between relations and concepts in an AMR graph and natural language word tokens. However, this connection is necessary for modeling the transformation process from a sentence dependency tree to an AMR graph. In other words, in contrast to a syntactic parse tree, the AMR graph is abstract. As a result, it may represent any number of sentences. Therefore, it does not contain any alignments between the sentence word tokens and the AMR graph concepts or relations, and the suitable aligner is required

[1] Word Sense Disambiguation

[2] Semantic Role Labels

to map from tokens to concepts. Besides, an AMR parser must have the ability to infer the concepts that have deeper meaning of the sentence and is independent on any tokens, directly.

Wang [19] provided a solution to this challenge, via the following steps:

1) Infer concepts that were not aligned to any particular parts of the sentence directly using transition system
2) Align string to AMR by graph-based aligner, which had benefits of the structural information in AMR graph.

TABLE 7
THE EVALUATION OF SOME AMR PARSING METHODS.

Method	Smatch	Unlabeled	No WSD	Named Entities	Wikification	Negations	Concepts	Reentrancy	SRL
Jones et al. [38]	71.0	74.0	72.0	78.0	71.0	57.0	86.0	49.0	64.0
Wang et al. [47]	67.0	69.0	64.0	75.0	-	18.0	80.0	41.0	60.0
Wang et al. [53]	64.7	71.0	66.0	75.0	-	14.0	80.0	36.0	59.0
Flanigan et al. [42]	67.0	69.0	68.0	79.0	75.0	45.0	83.0	42.0	60.0
Damonte et al. [57]	64.0	69.0	65.0	83.0	70.0	47.0	83.0	41.0	57.0
Ballesteros & Al-Onaizan [56] (JAMR)	65.9	71.0	66.0	80.0	-	45.0	82.0	46.0	59.0
Ballesteros & Al-Onaizan [56] + Label (JAMR)	67.0	72.0	68.0	81.0	79.0	46.0	82.0	48.0	64.0
Ballesteros & Al-Onaizan [56] + Label	68.3	73.0	69.0	79.0	78.0	62.0	82.0	51.0	66.0
Ballesteros & Al-Onaizan [56] + Label + POS[1]	69.0	74.0	70.0	80.0	79.0	62.0	83.0	51.0	67.0
Ballesteros & Al-Onaizan [56] + Label + POS + DEP[2]	69.4	75.0	70.0	81.0	79.0	65.0	83.0	52.0	67.0
Ballesteros & Al-Onaizan [56] + Label + POS + DEP + NER[3]	69.8	75.0	70.0	83.0	79.0	62.0	83.0	52.0	67.0
Ballesteros & Al-Onaizan [56] + Label + POS+ DEP + NER + Concepts	70.9	76.0	71.0	83.0	79.0	66.0	84.0	54.0	69.0
Ballesteros & Al-Onaizan [56] + Label + POS + DEP + NER + Concepts + BERT	72.9	78.0	73.0	83.0	78.0	67.0	84.0	58.0	72.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention	69.8	75.0	70.0	80.0	78.0	63.0	83.0	53.0	68.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS	70.4	75.0	71.0	80.0	79.0	64.0	83.0	53.0	68.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP	70.7	75.0	71.0	80.0	79.0	62.0	83.0	53.0	68.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP + NER	70.8	76.0	71.0	83.0	79.0	64.0	83.0	53.0	68.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP + NER + Concepts	71.8	77.0	72.0	82.0	78.0	66.0	84.0	56.0	70.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP + NER + Concepts + BERT	73.1	78.0	74.0	82.0	79.0	66.0	84.0	58.0	72.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP + NER + Concepts + BERT + Smatch	73.6	78.0	74.0	84.0	79.0	64.0	85.0	59.0	72.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + BERT	73.4	78.0	74.0	83.0	79.0	64.0	84.0	57.0	71.0
Ballesteros & Al-Onaizan [56] + Label (JAMR) + Attention + POS + DEP + NER + Concepts + BERT + Smatch + RL[4]	75.5	80.0	76.0	83.0	80.0	67.0	86.0	56.0	72.0
van Noord & Bos [106]	71.0	74.0	72.0	79.0	65.0	62.0	82.0	52.0	66.0
Guo & Lu [63]	69.8	74.0	72.0	78.0	71.0	57.0	84.0	49.0	64.0
Groschwitz et al. [59] (PD[5])	71.0	74.0	72.0	78.0	71.0	57.0	84.0	49.0	64.0
Groschwitz et al. [59] (FTD[6])	70.0	74.0	70.0	77.0	71.0	55.0	84.0	46.0	62.0
Vilares & Gómez-Rodríguez [61]	64.0	68.0	65.0	83.0	70.0	47.0	83.0	44.0	57.0

[1] Part of Speech
[2] Dependency Trees
[3] Named Entity labels
[4] Reinforcement Learning
[5] Projective Decoder
[6] Fixed-Tree Decoder

Welch et al. [65] (Retrained NER)	65.0	71.0	66.0	77.0	-	14.0	80.0	36.0	58.0
Welch et al. [65] (Oracle NER)	69.0	74.0	70.0	79.0	-	15.0	85.0	37.0	61.0
Lyu & Titov [78]	74.4	77.0	76.0	86.0	76.0	58.0	86.0	52.0	70.0
Naseem et al. [64]	75.5	80.0	76.0	83.0	80.0	67.0	86.0	56.0	72.0
Zhang et al. [73]	76.3	79.0	77.0	78.0	86.0	75.0	85.0	60.0	70.0
Gu et al. [66]	67.0	72.0	68.0	82.0	-	47.0	85.0	42.0	61.0
Zhou et al. [67] (small)	81.7	85.4	82.2	88.9	78.7	67.5	88.9	70.6	80.7
Zhou et al. [67] (base)	81.8	85.5	82.3	88.5	78.8	69.7	88.7	71.1	80.8
Bai et al. [75]	85.4	88.3	85.8	91.5	81.4	74.0	91.2	73.5	81.5

TABLE 8
FEATURES OF VARIOUS PARSERS.

Parser	Category	Features					Pipeline
		POS	DEP	NER	SRL	Other	
Flanigan et al. [37]	Graph-based	✓	✓	×	×	No	Yes
Artzi et al. [36]	Grammar-based	✓	×	×	×	No	Yes
Peng et al. [34]	Grammar-based	×	×	×	×	No	No
Werling et al. [41]	Graph-based	✓	✓	✓	×	No	Yes
Wang et al. [47]	Transition-based	✓	✓	✓	×	No	Yes
Wang et al. [53]	Conversion-based	✓	✓	✓	✓	No	Yes
Pust et al. [77]	Other methods	×	×	✓	×	Word[1]	Yes
Flanigan et al. [42]	Graph-based	✓	✓	✓	×	No	Yes
Zhou et al. [43]	Graph-based	✓	✓	✓	✓	No	No
Barzdins & Gosko [68]	Seq2Seq-based	✓	✓	✓	×	No	Yes
Goodman et al. [55]	Transition-based	✓	×	✓	×	No	Yes
Foland & Martin [45]	Seq2Seq-based	×	×	✓	×	No	Yes
Wang & Xue [50]	Transition-based	✓	✓	✓	✓	No	Yes
Damonte et al. [57]	Transition-based	✓	✓	✓	✓	No	No
Buys & Blunsom [70]	Seq2Seq-based	✓	×	✓	×	No	No
Ballesteros & Al-Onaizan [56]	Transition-based	✓	✓	×	×	No	No
Peng et al. [58]	Transition-based	✓	✓	×	×	No	No
Konstas et al. [69]	Seq2Seq-based	×	×	✓	×	Giga[2]	No
van Noord & Bos [72]	Seq2Seq-based	✓	×	×	×	Silver[3]	No
Groschwitz et al. [59]	Transition-based	✓	×	✓	×	No	No
Vilares & Gómez-Rodríguez [61]	Transition-based	✓	✓	✓	×	No	No
Guo & Lu [63]	Transition-based	✓	×	×	×	No	No
Lyu & Titov [78]	Other methods	✓	×	✓	×	No	Yes
Gu et al. [66]	Transition-based	✓	✓	✓	×	No	No
Zhou et al. [67]	Transition-based	×	×	✓	✓	No	No
Bai et al. [75]	Seq2Seq-based	×	×	×	×	No	No

TABLE 9
THE EVALUATION OF PARSERS ON DIFFERENT AMR DATASETS BASED ON F1 SCORE.

Parser	F1 score (%)			
	LDC2014T12 (Newswire section)	LDC2014T12	LDC2015E86	LDC2017T10
Flanigan et al. [37]	59.0	58.0	-	-
Artzi et al. [36]	67.0	-	-	-
Peng et al. [34]	-	-	52.0	-
Werling et al. [41]	62.0	-	-	-
Wang et al. [47]	70.0	66.5	-	-
Wang et al. [53]	-	66.5	67.3	-
Pust et al. [77]	-	67.1	-	-
Flanigan et al. [42]	-	66.0	67.0	-
Zhou et al. [43]	71.0	66.0	-	-
Barzdins & Gosko [68]	-	-	67.2	-
Goodman et al. [55]	70.0	-	64.0	-
Foland & Martin [45]	-	-	70.7	-
Wang & Xue [50]	-	68.1	68.1	-
Damonte et al. [57]	-	64.0	64.0	-
Buys & Blunsom [70]	-	-	-	61.9
Ballesteros & Al-Onaizan [56]	69.0	64.0	-	-

[1] WordNet for concept identification
[2] 20M unlabeled Giga-word
[3] 100k additional training pairs created by using CAMR and JAMR

Peng et al. [58]	-	-	64.0	-
Konstas et al. [69]	-	-	62.1	-
van Noord & Bos [72]	-	-	68.5	71.0
Groschwitz et al. [59]	-	-	70.2	71.0
Vilares & Gómez-Rodríguez [61]	-	-	64.0	-
Guo & Lu [63]	74.0	68.3	68.7	69.8
Lyu & Titov [78]	-	-	73.7	74.4
Gu et al. [66]	67.0	62.0	-	-
Zhou et al. [67] (small)	-	78.2	-	81.7
Zhou et al. [67] (base)	-	78.5	-	81.8
Bai et al. [75]	-	-	-	85.4

TABLE 10
THE EVALUATION OF SOME AMR GENERATING METHODS BASED ON BLEU.

Method	BLEU
Flanigan et al. [88]	22.10
Pourdamghani et al. [92]	26.90
Song et al. [91]	23.00
Song et al. [90]	25.62
Konstas et al. [69]	22.00
Gruzitis et al. [52]	18.82
Schick [96]	27.40
Fan & Gardent [93]	29.70
Jin & Gildea [97]	31.20

In general, AMR provides a noteworthy level of abstraction of the utterance propositional content. Although, it cannot capture the force, the tense and the aspect of it.

Reentrancy is the attribute that made AMR as a graph, not a tree; it means the same concept can participate in multiple relations. Parsing a sentence to a graph would need more complex approach, so it brings some challenges for both decoding and learning processes.

Finally, data sparsity in AMR means that many of AMR concepts are word sense-disambiguated lemmas, which were drawn from Propbank, and since the AMR Bank is not big, a lot of concepts in development or test set just occurred a few times or even never appeared in the training set. Managing this challenge forces the learning algorithm to find the sharing features among similar concepts. Generally, researchers address this issue using deep learning and neural methods to produce a bi-directional LSTM-based concept identifier upon a re-designed concept set.

Generating natural language from AMR is a completely complicated task. Because, its graph is abstracted away from their associated surface forms and the process needs training data, which has been extensively annotated by human agents. It can be said that this is still a preferred purpose, even if it is far away, because AMR eliminates all ambiguity from statements. In this regard, it can be mentioned that this aspect of AMR is a disadvantage too, as it makes all misunderstanding catastrophic. The other reason is the ambiguous aspects of natural language itself. Mostly, it happens in cases that even human agents cannot completely resolve ambiguities, or in cases that an ambiguous statement obtains importance.

D. AMR features and future direction

Here, we define some main questions about existing features of AMR and its future direction.

What is difference between AMR and other meaning representation methods?

What AMRs have in common with traditional meaning representation techniques is that logical conjunction is often implicit. Also, scope is not explicitly displayed in AMRs, thus the negation representation in AMR is completely different from previous approaches.

The common denominator between AMR and some other meaning representation methods is the implicitness of the logical connections. However, in terms of expressiveness, the power of AMR is lower. In AMR, scope is not presented clearly; therefore, the negation representation in AMR is basically different from other meaning representation methods. Unlike other representations, AMR is able to represent various aspects of information structure, for example, the role inversion process could change the concept domain. Albeit the presentation of negation would lead to some changes in meaning. For example, as Discourse Representation

Structures (DRSs) [18] in a clausal structure is very similar to the triple notation of AMRs, and both of these representations try to model natural language meaning, it would be instructive to compare these two methods. The primary difference between these representations is that DRSs have explicit scopes and scope operators like negation. According to the existence of scope this method, their clauses are more complicated in comparison with AMR triples. The length of DRS clauses is not constant, and it can be three or four, unlike AMR, which the length of its triples is always three. Furthermore, DRS clauses include two different kinds of variables, for scopes and discourse referents, while there is only one type of AMR triples. Besides, DRS model considers the tense of sentences; in contrast, AMR does not represent tense. Generally, in DRS method the tense connected information is encoded in a clausal form with three extra clauses, which specify a WordNet concept, semantic role and a comparing operator. As every logical operator includes a scope, their number represents a lower boundary for the number of scopes in the semantic representations. In many applications of natural language processing like information retrieval, tense is not needed as a significant parameter for development. Therefore, AMR could perform faster in these applications. However, if in one application the tense should be processed too, a syntactic parser could be added to it.

Is it possible to share AMR in different languages?

Although AMR model extremely focused on English, it abstracts away from morphological and syntactical features that distinguish different natural languages. Therefore, it has the potential to be generalized to some similar or even all other natural languages. It seems that machine translation-based systems are not suitable in this area, and instead it is better to concentrate on projection-based parsers. A review of related works illustrate that recent parsers can overcome divergence, which exists in the translation process. Besides, the approach for concept identification should be accurate enough to attain acceptable outcomes. Thus, in many cases the main reason for reaching the sub-optimal results by parsers, which use Smatch metric, is the existence of notable noise sources in the annotation projection method. It must be considered that it is not related to inconsistency in AMR among different languages. Therefore, having a global method for AMR parsing is possible and producing appropriate datasets in this regard would be counted as an important step.

Is it possible to provide a theoretical semantic model for AMR?

AMRs could be translated into FOL methodically, so this representation would have an indirect theoretical model interpretation. Indeed, AMRs do not have recurrent variables, so they can be mapped into a selectable part of FOL. This comprises the polarity flag existed in AMR in order to show negation. The mapping process is solely indicative and can be simply developed, for instance in Prolog program format.

Is it necessary to expand the AMR language?

In pursuance of handling weaknesses that currently AMR has, like quantifier scope and projection phenomena, the AMR language must be expanded. More research is still needed in this area.

Is it possible to have universal quantifier in AMR?

Using multiple polarity relations, the universal quantifier can be added to the initial AMR. Although only one universal quantifier can be applied in AMR.

VI. CONCLUSION

In this paper, we reviewed related works about the AMR model, its syntax and main abilities. Furthermore, we reviewed the existing methods for generating natural language from AMR and parsing it to AMR. Afterwards, we explained how they have been applied in NLP tasks. Besides, we talked about the standard datasets and metrics that can be used for taking advantages of AMR graphs in different applications. Finally, we discussed some common challenges of working with AMR structure. Generally, AMR is constructed based on graph representation, abstraction and framesets.

As discussed, the existed AMR corpus was created manually by human annotators and contains thousands of sentences already. In addition, AMRs can be displayed as conjunction of logical triples. In this regard, the previously introduced Smatch score can be applied to evaluate AMR parsing accuracy. In general, the AMR parsing and generation tasks have caught considerable amount of attention, as they are beneficial for vital NLP applications. However, there are still a lot of rooms for further improvements and there could be many pathways for future AMR-related works. One of the main ones is enlarging the AMR Bank corpus, which could result in better outcomes in NLU and NLG tasks. For example, in both AMR parsing and generation tasks, semi-supervised algorithms and human-collaborated approaches could enhance the performance level.

Future research on graph-based AMR parsing may expand AMR to constitute more linguistically motivated constraints and higher order attributes. The other work could improve the prototypes for AMR-based systems. Undoubtedly, the AMR model will change in future, so finally it may contain more relations, quantification, or entity normalization. Besides, an acquisitive list of more abstract frames can also be imagined.

REFERENCES

[1] Marcus, Mitchell P.; Santorini, Beatrice; Marcinkiewicz, Mary Ann, "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313-330, 1993.

[2] Abolghasemi, Majid; Dadkhah, Chitra; Tohidi, Nasim, "HTS-DL: Hybrid Text Summarization System using Deep Learning," in *The 27th International Computer Conference, the Computer Society of Iran*, Tehran, Online, 2022.

[3] Tohidi, Nasim; Dadkhah, Chitra; Rustamov, Rustam B., "Optimizing the performance of Persian multi-objective question answering system," in *16th International Conference on "Technical and Physical Problems of Electrical Engineering"*, Istanbul, Online, 2020.

[4] Tohidi, Nasim; Hasheminejad, Seyed Mohammad Hossein;, "MOQAS: Multi-objective question answering system," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 4, pp. 3495-3512, 2019.

[5] J. Flanigan, "Parsing and Generation for the Abstract Meaning Representation," Carnegie Mellon University, Pittsburgh, 2018.

[6] U. Weinreich, "On the Semantic Structure of Language," in *On Semantics*, Philadelphia, University of Pennsylvania Press, 2018, pp. 37-96.

[7] Basile, Valerio; Bos, Johan; Evang, Kilian; Venhuizen, Noortje, "A platform for collaborative semantic annotation," in *In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics.*, 2012.

[8] Butler, Alistair; Yoshimoto, Kei., "Banking Meaning Representations from Treebanks.," *Linguistic Issues in Language Technology*, vol. 7, no. 6, pp. 1-22, 2012.

[9] Abend, O.; Rappoport A., "UCCA: A Semantics-based Grammatical Annotation Scheme.," in *In Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, 2013.

[10] Böhmová, Alena; Hajič, Jan; Hajičová, Eva; Hladká, Barbora, The Prague dependency treebank, vol. 20, Springer, 2003, p. 103–127.

[11] Uchida, H.; Zhu, M.; Senta, T. D., "an electronic language for communication, understanding and collaboration.," UNL: Universal Networking Language, IAS/UNU Tokyo, 1996.

[12] Banarescu, Laura; Bonial, Claire; Cai, Shu; Georgescu, Madalina; Griffitt, Kira; Hermjakob, Ulf; Knight, Kevin; Koehn, Philipp; Palmer, Martha; Schneider, Nathan, "Abstract Meaning Representation for Sembanking," in *In proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, Sofia, Bulgaria, 2013.

[13] Tohidi, Nasim; Dadkhah, Chitra, "Abstract Meaning Representation Applications: A Short Review," in *The 1st Conference on Artificial Intelligence and Smart Computing*, Online, Semnan, Iran, 2022.

[14] J. Bos, "Expressive Power of Abstract Meaning Representations," *Computational Linguistics*, vol. 42, pp. 527-535, 2016.

[15] Cai, Shu; Knight, Kevin, "Smatch: an Evaluation Metric for Semantic Feature Structures," in *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, 2013.

[16] Knight, Kevin; Badarau, Bianca; Baranescu, Laura; Bonial, Claire; Bardocz, Madalina; Griffitt, Kira; Hermjakob, Ulf; Marcu, Daniel; Palmer, Martha; O'Gorman, Tim; Schneider, Nathan, "Abstract Meaning Representation (AMR) Annotation Release 3.0," Linguistic Data Consortium, Philadelphia, 2020.

[17] Jurafsky, Daniel; Martin, James H., Speech and Language Processing, Upper Saddle River, NJUnited States: Prentice Hall, 2019.

[18] J. Bos, "Economical Discourse Representation Theory," in *International Workshop on Controlled Natural Language*, Verlag Berlin Heidelberg, 2010.

[19] C. Wang, "Abstract Meaning Representation Parsing," PhD thesis, Brandeis University, 2018.

[20] Li, Bin; Wen, Yuan; Song, Li; Qu, Weiguang; Xue, Nianwen, "Building a Chinese AMR Bank with Concept and Relation Alignments," *Linguistic Issues in Language Technology*, vol. 18, no. 1, 2019.

[21] Wein, Shira; Schneider, Nathan, "Classifying Divergences in Cross-lingual AMR Pairs," in *The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, Punta Cana, Dominican Republic, 2021.

[22] Pustejovsky, James; Lai, Ken; Xue, Nianwen, "Modeling Quantification and Scope in Abstract Meaning Representations," in *The First International Workshop on Designing Meaning Representations*, Florence, Italy, 2019.

[23] O'Gorman, Tim; Regan, Michael; Griffitt, Kira; Hermjakob, Ulf; Knight, Kevin; Palmer, Martha, "AMR Beyond the Sentence: the Multi-sentence AMR corpus," in *The 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, 2018.

[24] Anikina, Tatiana; Koller, Alexander; Roth, Michael, "Predicting Coreference in Abstract Meaning Representations," in *The Third Workshop on Computational Models of Reference, Anaphora and Coreference*, Barcelona, Spain (online), 2020.

[25] Fu, Qiankun; Song, Linfeng; Du, Wenyu; Zhang, Yue, "End-to-End AMR Coreference Resolution," in *The 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online, 2021.

[26] Lai, Kenneth; Donatelli, Lucia; Pustejovsky, James, "A Continuation Semantics for Abstract Meaning Representation," in *The Second International Workshop on Designing Meaning Representations*, Barcelona, Spain (online), 2020.

[27] Bonn, Julia; Palmer, Martha; Cai, Zheng; Wright-Bettner, Kristin, "Spatial AMR: Expanded Spatial Annotation in the Context of a Grounded Minecraft Corpus," in *The 12th Language Resources and Evaluation Conference*, Marseille, France, 2020.

[28] Stein, Katharina; Donatelli, Lucia, "Representing Implicit Positive Meaning of Negated Statements in AMR," in *The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, Punta Cana, Dominican Republic, 2021.

[29] Williamson, Gregor; Elliott, Patrick; Ji, Yuxin, "Intensionalizing Abstract Meaning Representations: Non-Veridicality and Scope," in *The Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, 2021.

[30] Donatelli, Lucia; Regan, Michael; Croft, William; Schneider, Nathan, "Annotation of Tense and Aspect Semantics for Sentential AMR," in *The Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, Santa Fe, New Mexico, USA, 2018.

[31] M. F. Bakal, "Graph-to-Graph Translations To Augment Abstract Meaning Representation Tense And Aspect," University of Michigan, 2021.

[32] Szubert, Ida; Damonte, Marco; Cohen, Shay B.; Steedman, Mark, "The Role of Reentrancies in Abstract Meaning Representation Parsing," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online, 2020.

[33] Chiang, David; Andreas, Jacob; Bauer, Daniel; Hermann, Karl Moritz; Jones, Bevan; Knight, Kevin, "Parsing graphs with hyperedge replacement grammars," in *In Proceedings of the 51st Meeting of the Association of Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, 2013.

[34] Peng, Xiaochang; Song, Linfeng; Gildea, Daniel, "A synchronous hyperedge replacement grammar based approach for AMR parsing," in *In Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, Beijing, China, 2015.

[35] Peng, Xiaochang; Gildea, Daniel, "UofR at SemEval-2016 task 8: Learning synchronous hyperedge replacement grammar for AMR parsing," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[36] Artzi, Yoav; Lee, Kenton; Zettlemoyer, Luke, "Broad-coverage CCG semantic parsing with AMR," in *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[37] Flanigan, Jeffrey; Thomson, Sam; Carbonell, Jaime; Dyer, Chris; Smith. Noah A., "A Discriminative Graph-Based Parser for the Abstract Meaning Representation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland, 2014.

[38] Jones, Bevan; Andreas, Jacob; Bauer, Daniel; Hermann, Karl Moritz; Knight, Kevin, "Semantics-Based Machine Translation with Hyperedge Replacement Grammars," in *In Proceedings of COLING.*, 2012.

[39] Pourdamghani, Nima; Gao, Yang; Hermjakob, Ulf; Knight, Kevin, "Aligning English Strings with Abstract Meaning Representation Graphs," in *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[40] Xue, Nianwen; Bojar, Ondřej; Hajič, Jan; Palmer, Martha; Urešová, Zdeňka; Zhang, Xiuhong, "Not an Interlingua, But Close: Comparison of English AMRs to Chinese and Czech," in *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, 2014.

[41] Werling, Keenon; Angeli, Gabor; Manning, Christopher D., "Robust Subgraph Generation Improves Abstract Meaning Representation Parsing," in *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China, 2015.

[42] Flanigan, Jeffrey; Dyer, Chris; Smith, Noah A.; Carbonell, Jaime, "CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016a.

[43] Zhou, Junsheng; Xu, Feiyu; Uszkoreit, Hans; QU, Weiguang; Li, Ran; Gu, Yanhui, "AMR parsing with an incremental joint model," in *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016.

[44] Foland, William; Martin, James H., "CU-NLP at SemEval-2016 task 8: AMR parsing using LSTM-based recurrent neural networks," in *In Proceedings of the 10th InternationalWorkshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[45] Foland, William; Martin, James H., "Abstract meaning representation parsing using LSTM recurrent neural networks," in *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, 2017.

[46] Rao, Sudha; Vyas, Yogarshi; Daumé III, Hal; Resnik, Philip, "CLIP@UMD at SemEval-2016 Task 8: Parser for Abstract Meaning Representation using Learning to Search," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[47] Wang, Chuan; Xue, Nianwen; Pradhan, Sameer, "Boosting transition-based AMR parsing with refined actions and auxiliary analyzers," in *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China, 2015a.

[48] Goodman, James; Vlachos, Andreas; Naradowsky, Jason, "UCL+Sheffield at SemEval-2016 Task 8: Imitation learning for AMR parsing with an alpha-bound," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016b.

[49] Puzikov, Yevgeniy; Kawahara, Daisuke; Kurohashi, Sadao, "M2L at SemEval-2016 task 8: AMR parsing with neural networks," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[50] Wang, Chuan; Xue, Nianwen, "Getting the Most out of AMR Parsing," in *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, Copenhagen, Denmark, 2017.

[51] Nguyen, Khoa; Nguyen, Dang, "UIT-DANGNTCLNLP at SemEval-2017 Task 9: Building Scientific Concept Fixing Patterns for Improving CAMR," in *In Proceedings of the 11th InternationalWorkshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, 2017.

[52] Gruzitis, Normunds; Gosko, Didzis; Barzdins, Guntis, "RIGOTRIO at SemEval-2017 Task 9: Combining Machine Learning and Grammar Engineering for AMR Parsing and Generation.," in *In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, 2017.

[53] Wang, Chuan; Xue, Nianwen; Pradhan, Sameer, "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*, Denver, Colorado, 2015b.

[54] Brandt, Lauritz; Grimm, David; Zhou, Mengfei; Versley, Yannick, "ICL-HD at SemEval-2016 Task 8: Meaning Representation Parsing - Augmenting AMR Parsing with a Preposition Semantic Role Labeling Neural Network," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[55] Goodman, James; Vlachos, Andreas; Naradowsky, Jason, "Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, 2016a.

[56] Ballesteros, Miguel; Al-Onaizan, Yaser, "AMR Parsing using Stack-LSTMs," in *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017.

[57] Damonte, Marco; Cohen, Shay B.; Satta, Giorgio, "An Incremental Parser for Abstract Meaning Representation," in *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Valencia, Spain, 2017.

[58] Peng, Xiaochang; Wang, Chuan; Gildea, Daniel; Xue, Nianwen, "Addressing the Data Sparsity Issue in Neural AMR Parsing," in *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Valencia, Spain, 2017.

[59] Groschwitz, Jonas; Lindemann, Matthias; Fowlie, Meaghan; Johnson, Mark; Koller, Alexander, "AMR dependency parsing with a typed semantic algebra," in *The 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, 2018.

[60] Groschwitz, Jonas; Fowlie, Meaghan; Johnson, Mark; Koller, Alexander, "A constrained graph algebra for semantic parsing with AMRs," in *The 12th International Conference on Computational Semantics (IWCS)*, 2017.

[61] Vilares, David; Gómez-Rodríguez, Carlos, "A Transition-Based Algorithm for Unrestricted AMR Parsing," in *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana, 2018.

[62] Wang, Chuan; Li, Bin; Xue, Nianwen, "Transition-Based Chinese AMR Parsing," in *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana, 2018.

[63] Guo, Zhijiang; Lu, Wei, "Better Transition-Based AMR Parsing with a Refined Search Space," in *The 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[64] Naseem, Tahira; Shah, Abhishek; Wan, Hui; Florian, Radu; Roukos, Salim; Ballesteros, Miguel, "Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning," in *The 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019.

[65] Welch, Charles; Kummerfeld, Jonathan K.; Feng, Song; Mihalcea, Rada, "World Knowledge for Abstract Meaning Representation Parsing," in *The Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018.

[66] Gu, Min; Gu, Yanhui; Luo, Weilan; Xu, Guandong; Yang, Zhenglu; Zhou, Junsheng; Qu, Weiguang, "From text to graph: a general transition-based AMR parsing using neural network," *Neural Computing and Applications*, 2020.

[67] Zhou, Jiawei; Naseem, Tahira; Astudillo, Ramón Fernandez; Florian, Radu, "AMR Parsing with Action-Pointer Transformer," in *the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, 2021.

[68] Barzdins, Guntis; Gosko, Didzis, "RIGA at SemEval- 2016 Task 8: Impact of Smatch Extensions and Character-Level Neural Translation on AMR Parsing Accuracy," in *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, 2016.

[69] Konstas, Ioannis; Iyer, Srinivasan; Yatskar, Mark; Choi, Yejin; Zettlemoyer, Luke, "Neural AMR: Sequence-to-Sequence Models for Parsing and Generation," in *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017.

[70] Buys, Jan; Blunsom, Phil, "Robust Incremental Neural Semantic Graph Parsing," *arXiv*, p. 1215–1226, 2017.

[71] Misra, Dipendra Kumar; Artzi, Yoav, "Neural shiftreduce ccg semantic parsing," in *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016.

[72] van Noord, Rik; Bos, Johan, "Dealing with Coreference in Neural Semantic Parsing," in *In Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, 2017a.

[73] Zhang, Sheng; Ma, Xutai; Duh, Kevin; Van Durme, Benjamin, "AMR Parsing as Sequence-to-Graph Transduction," in *The 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019.

[74] Dozat, Timothy; Manning, Christopher D., "Deep Biaffine Attention for Neural Dependency Parsing," in *arXiv:1611.01734*, 2016.

[75] Bai, Xuefeng; Chen, Yulong; Zhang, Yue, "Graph Pre-training for AMR Parsing and Generation," *arXiv:2203.07836*, vol. 3, pp. 1-15, 2022.

[76] Lewis, Mike; Liu, Yinhan; Goyal, Naman; Ghazvininejad, Marjan; Mohamed, Abdelrahman; Levy, Omer; Stoyanov, Veselin; Zettlemoyer, Luke, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *The 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020.

[77] Pust, Michael; Hermjakob, Ulf; Knight, Kevin; Marcu, Daniel; May, Jonathan, "Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation," in *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[78] Lyu, Chunchuan; Titov, Ivan, "AMR Parsing as Graph Prediction with Latent Alignment," in *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, 2018.

[79] Gildea, Daniel; Jurafsky, Daniel, "Automatic Labeling of Semantic Roles," in *In 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, 2000.

[80] V. Klimeš, "Analytical and Tectogrammatical Analysis of a Natural Language," Ph.D. thesis, Charles University, Prague, 2006a.

[81] V. Klimeš, "Transformation-based tectogrammatical analysis of czech," in *In International Conference on Text, Speech and Dialogue*, Berlin, Heidelberg, 2006b.

[82] V. Klimeš, "Transformation-based tectogrammatical dependency analysis of english," in *In International Conference on Text, Speech and Dialogue*, Berlin, Heidelberg, 2007.

[83] Bohnet, Bernd; Wanner, Leo; Mille, Simon; Burga, Alicia, "Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer," in *In Proceedings of the 23rd International Conference on Computational Linguistics*, Beijing, China, 2010.

[84] Langkilde, Irene; Knight, Kevin, "Generation that exploits corpus-based statistical knowledge," in *In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, Montreal, Quebec, Canada, 1998.

[85] Wong, Yuk Wah; Mooney, Raymond, "Generation by inverting a semantic parser that uses statistical machine translation," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, Rochester, New York, 2007.

[86] Belz, Anja; White, Michael; Espinosa, Dominic; Kow, Eric; Hogan, Deirdre; Stent, Amanda, "The first surface realisation shared task: Overview and evaluation results," in *In Proceedings of the 13th European workshop on natural language generation*, Nancy, France, 2011.

[87] I. Langkilde, "Forest-based statistical sentence Generation," in *In proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

[88] Flanigan, Jeffrey; Dyer, Chris; Smith, Noah A.; Carbonell, Jaime, "Generation from abstract meaning representation using tree transducers," in *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, 2016b.

[89] Pelja Paul, N.; Revathy, P.; Sini, G.M.; Binu, R., "Automatic AMR Generation for Simple Sentences Using Dependency Parser," *Procedia Technology*, vol. 24, pp. 1528-1533, 2016.

[90] Song, Linfeng; Peng, Xiaochang; Zhang, Yue; Wang, Zhiguo; Gildea, Daniel, "AMR-to-text Generation with Synchronous Node Replacement Grammar," in *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada, 2017.

[91] Song, Linfeng; Zhang, Yue; Peng, Xiaochang; Wang, Zhiguo; Gildea, Daniel, "AMR-to-text generation as a Traveling Salesman Problem," in *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016.

[92] Pourdamghani, Nima; Knight, Kevin; Hermjakob, Ulf, "Generating english from abstract meaning representations," in *In Proceedings of the 9th International Natural Language Generation conference*, Edinburgh, UK, 2016.

[93] Fan, Angela; Gardent, Claire, "Multilingual AMR-to-Text Generation," in *The 2020 Conference on Empirical Methods in Natural Language Processing*, Online, 2020.

[94] Mille, Simon; Carlini, Roberto; Burga, Alicia; Wanner, Leo , "FORGe at SemEval-2017 Task 9: Deep sentence generation based on a sequence of graph transducers," in *In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, 2017.

[95] Lampouras, Gerasimos; Vlachos, Andreas, "Sheffield at SemEval-2017 Task 9: Transition-based language generation from AMR," in *In Proceedings of the 11th InternationalWorkshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, 2017.

[96] T. Schick, "Transition-Based Generation from Abstract Meaning Representations," Master's thesis, 2017.

[97] Jin, Lisa; Gildea, Daniel, "Generalized Shortest-Paths Encoders for AMR-to-Text Generation," in *The 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online), 2020.

[98] Ballesteros, Miguel; Bohnet, Bernd; Mille, Simon; Wanner, Leo, "Data-driven sentence generation with non-isomorphic trees," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado, 2015.

[99] May, Jonathan; Priyadarshi, Jay, "Semeval-2017 task 9: Abstract meaning representation parsing and generation," in *In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, 2017.

[100] Knight, Kevin; Badarau, Bianca; Baranescu, Laura; Bonial, Claire; Bardocz, Madalina; Griffitt, Kira; Hermjakob, Ulf; Marcu, Daniel; Palmer, Martha; O'Gorman, Tim; Schneider, Nathan, "Abstract Meaning Representation (AMR) Annotation Release 3.0," Linguistic Data Consortium, Philadelphia, 2020.

[101] Zettlemoyer, Luke S.; Collins, Michael, "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars," in *In UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, 2005.

[102] Papineni, Kishore; Roukos, Salim; Ward, Todd; Zhu, Wei-Jing, "Bleu: a Method for Automatic Evaluation of Machine Translation," in *The 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, 2002.

[103] Allen, James F.; Swift, Mary; Beaumont, Will De, "Deep semantic analysis of text," in *In Proceedings of the 2008 Conference on Semantics in Text Processing*, 2008.

[104] Dridan, Rebecca; Oepen, Stephan, "Parser Evaluation using Elementary Dependency Matching," in *In Proceedings of the 12th International Conference on Parsing Technologies*, Dublin, Ireland, 2011.

[105] Opitz, Juri; Parcalabescu, Letitia; Frank, Anette, "AMR Similarity Metrics from Principles," *Transactions of the Association for Computational Linguistics*, vol. 8, p. 522–538, 2020.

[106] van Noord, Rik; Bos, Johan, "Neural semantic parsing by character-based translation: Experiments with abstract meaning representations," *arXiv:1705.09980*, 2017b.

[107] Song, Li; Dai, Yuling; Liu, Yihuan; Li, Bin; Qu, Weiguang, "Construct a Sense-Frame Aligned Predicate Lexicon for Chinese AMR Corpus," in *The 12th Language Resources and Evaluation Conference*, Marseille, France, 2020.