ARL-TR-9449 ● APR 2022

DEVCOM ARMY RESEARCH LABORATORY

Dialogue-AMR Parsing Pipeline

by Mitchell Abrams, Claire Bonial, and Clare Voss

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9449 ● APR 2022

Dialogue-AMR Parsing Pipeline

Mitchell Abrams
Institute of Human and Machine Computing

Claire Bonial and Clare Voss
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
April 2022	Technical Report	June 2018–October 2021

4. TITLE AND SUBTITLE
Dialogue-AMR Parsing Pipeline

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Mitchell Abrams, Claire Bonial, and Clare Voss

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLC-IT
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9449

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Claire Bonial 0000-0002-3154-2852

14. ABSTRACT

This research forms part of a larger project focused on natural language understanding (NLU) in the development of a two-way human–robot dialogue system in the search and navigation domain. We leverage Abstract Meaning Representation (AMR) to capture and structure the semantic content of natural language instructions in a machine-readable, directed, a-cyclic graph. Two key challenges exist for NLU in this task: 1) how to effectively map AMR to a constrained robot-action specification within a particular domain and 2) how to preserve necessary elements for general understanding of human language with the goal that our robot may expand its capabilities beyond a single domain. To address these challenges, we establish a two-step NLU approach in which automatically obtained AMR graphs of the input language are converted into "Dialogue-AMR" graphs, which is a new version of AMR that is augmented with tense, aspect, and speech act information. Here, we detail both rule-based and classifier-based methods to transform AMR graphs into Dialogue-AMR graphs, thereby bridging the gap from unconstrained natural-language input to a fixed set of robot actions.

15. SUBJECT TERMS
meaning representation, dialogue, natural language processing, graph-to-graph conversion, human–robot interaction, Military Information Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Claire Bonial
Unclassified	Unclassified	Unclassified	UU	28	19b. TELEPHONE NUMBER (Include area code) 301-394-1431

Contents

List of Figures

1. Introduction

This report presents a "parsing" (automatic conversion into a structured parse) pipeline for a new version of Abstract Meaning Representation (AMR), "Dialogue-AMR," which is a meaning-representation framework for dialogue. This research and work are situated in the context of a human–robot dialogue system in the search and navigation domain.

This conversion system is part of a two-step Natural Language Understanding (NLU) pipeline for human–robot dialogue. As depicted in Fig. 1, the first step captures the core semantics from unconstrained language input in the form of an AMR graph, which we will refer to as "Standard-AMR". The second step—specifically, the graph-to-graph (G2G) transformation process—converts the Standard-AMR graph to a Dialogue-AMR. A Dialogue-AMR contains information critical for human–robot communication and robot execution.

This conversion system aims to be effective for automatically producing Dialogue-AMR for human–robot dialogue and scaling to new domains that can also benefit from Dialogue-AMR, such as the Minecraft blocks-world domain in which the dialogue is focused on building structures from blocks in a virtual environment.

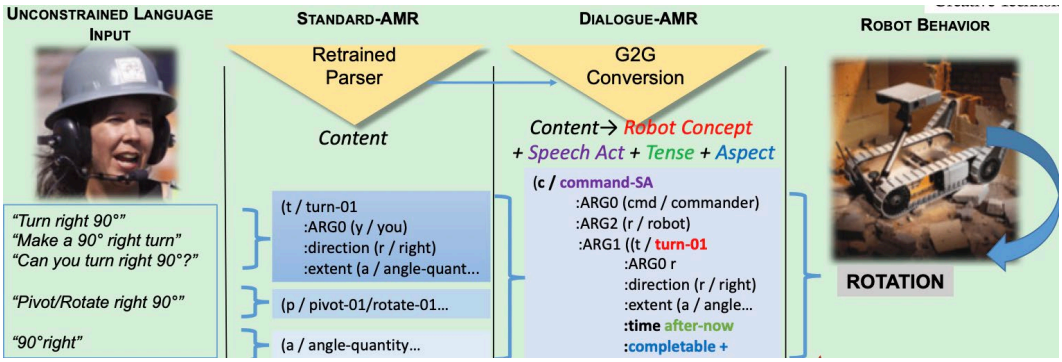

Fig. 1 Dialogue-AMR conversion

1.1 Introducing AMR

To facilitate natural language understanding (NLU) for dialogue and various Natural Language Processing (NLP) tasks, we leverage the AMR. It captures the propositional content of natural language in a machine-readable, direct, acyclic graph (Barnescu et al. 2013).

AMR is edge-labeled and leaf-labeled, where leaves are labeled with concepts (`d / dog`) and edges are relations (roles).

AMR can appear in PENMAN notation: a human-readable tree-like format. Figure 2 is an example of a sentence and its corresponding AMR graph in the PENMAN representation.

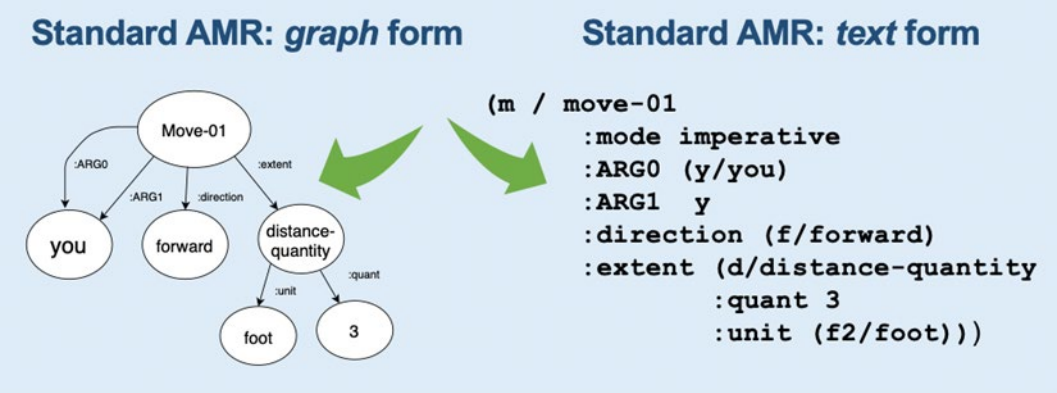

Fig. 2 AMR represented in graph form and text form

To walk through the anatomy of the graph, the edges here—`ARG0`, `ARG1`—are *relations* of `move`. Each node in the graph has concepts (`you`, `forward`, `distance-quantity`) with a variable (`y`, `f`, `d`). For the :`ARG0` relation in this example, ***y*** is an instance of ***you***.

A distinguishing feature of the AMR framework is its connection to the PropBank inventory, where predicates correspond to specific frames and senses. In PropBank, each predicate contains a specific sense number (*run-01*, *run-02*, etc.) with a role inventory (core roles). For this graph, the root action is *kick*, but specifically in the `move-01` PropBank sense, with the core argument roles, `:ARG0` and `:ARG1`—the move and thing moved, respectively.

While other meaning representation frameworks exist for NLU, AMR offers many advantages for our problem space.

First, AMR smooths over syntactic idiosyncrasies and maps multiple utterances to the same representation, generalizing across parts of speech and etymologically related words. This is important for handling unconstrained language input, where the same sentence can be expressed in many ways. For instance, the simple commands, *make a right, turn right,* and *rotate right,* all map to the same representation of a turning event—even with the use of alternative syntax or word choice. Additionally, this is critical for the human–robot dialogue domain where the goal is to take unconstrained language input and distill it to a fixed set of robot behaviors.

Another advantage of leveraging AMR is that accurate parsers exist, making it possible to automatically obtain an AMR without manual annotation. This is

important for live dialogue that requires a relatively quick interpretation of an utterance in an end-to-end system.

AMR allows many aspects of meaning to be stored in a simple data structure. This is attractive for the "situated" dialogue domain, where a human or system can pinpoint specific actions and direction specification with respect to the current physical "situated" environment and check for missing information or whether a robot can execute an action.

Another strength of AMR is that actions and concepts can be linked to PropBank Framesets with assigned sense numbers (e.g., *move-01*, *move-02*, and *turn-01*) and core arguments. This specific feature aids interpretation and transparency, especially with words that are used for a variety of meanings. In the light verb construction, *take a right,* the verb *take* has different possible meanings. However, AMR will assign a specific sense for this representation. In this case, *turn* is turn-01.

For the reasons stated, AMR is a promising framework for NLU. There are, however, a few drawbacks for using this meaning representation for human–robot dialogue and other NLP tasks.

AMR in its original form, or Standard-AMR, as we will refer to it, fails to represent *tense* and *aspect*. There is no distinguishing, for instance, between an event that happens in the past, present, or future (e.g., *I turned*, *I'm turning*, or *I will turn*). In the robot-navigation domain, tense is critical for coordinating when actions are taking place.

Other pragmatic information is also neglected in Standard-AMR framework, such as speech act information—or, roughly, the speaker's intent. For the sentence, *Can you move forward three feet*, while the core semantic content is preserved, it is unspecified whether this is a question of ability or a polite request. Explicitly knowing what utterances are *command*s might be important for human–robot navigation. Additionally, knowing what sentences are a *question*s can help in a question–answer task.

To leverage the strengths of Standard-AMR, and fill the gaps that are critical for successful human–robot communication, we created Dialogue-AMR (Bonial et al. 2020).

1.2 Dialogue-AMR

Dialogue-AMR serves as a refinement and expansion of Standard-AMR, handling *tense*, *aspect*, and *speaker intent*—all critical components for situated dialogue and understanding.

1.2.1 Augmented Components

To understand the anatomy of a Dialogue-AMR and its differences to Standard-AMR, in Fig. 3 we compare a) a Standard-AMR b) Dialogue-AMR template, and c) a filled-in Dialogue AMR for this sentence:

"Move forward three feet"

(a)
```
(m / move-01
     :ARG0 (y / you)
     :ARG1 y
     :direction (f/forward)
     :extent (d/distance-quantity
             :quant 3
             : unit (f2/foot))
```

(b)
```
(c / command-00
      :ARG0-speaker
      :ARG2-addressee
      :ARG1 (g / go-02 :completable +
         :ARG0-goer
         :ARG1-extent
         :ARG3-start point
         :ARG4-end point
         :path
         :direction
         :time (a2 / after
             :op1 (n / now)))
```

(c)
```
(c / command-00
     :ARG0-speaker
     :ARG2-addressee
     :ARG1 (g / go-02 :completable +
         :ARG0 a
         :ARG1 (d/distance-quantity
                :quant 3
                : unit (f2/foot))
         :ARG3 (h / here)
         :direction (f/forward)
         :time (a2 / after
             :op1 (n / now)))
```

Fig. 3 AMR and text with corresponding Dialogue-AMR skeletal template and filled-in template

We created Dialogue-AMR primarily for coverage of human–robot dialogue—so, for this example, we will use a typical *command* from our domain: *Move forward three feet*. Figure 3b represents a skeletal structure of the Dialogue-AMR representation of this sentence, and Fig. 3c offers the filled-in representation with different colors denoting the augmented components.

1.2.2 Speech Acts

Every Dialogue-AMR has a top anchor node corresponding to a "speech act" type—speech acts label what a speaker is trying to do with an utterance in the conversational context. This allows us to capture pragmatic information from the language of dialogue participants. The speech-act inventory currently consists of

36 types that were determined through an initial study of the Human–Robot Dialogue Corpus. Each speech act is marked with a 00 role set number or discussed in this report using the -SA ending to denote "Speech Act."

Under each top-level speech act relation are three main arguments. One of these arguments—`:ARG1`—is the main action or content of the speech act. This might be the action of a command or assertion or the content of a question. For the example sentence in Section 1.2.1, the speech act is a command (denoted *command-00*) and the main action of the command is *go-02.* The speech act and main action determine the template of the Dialogue-AMR.

(c / command-00	→	command:move	*move forward 3 feet*
		command:turn	*turn right 45 degrees*
		command:send-image	*send image*
		command:repeat	*do that again*
		command:cancel	*cancel that*
		command:stop	ok stop there

The remaining arguments—`:ARG0` and `:ARG2`—strictly belong to the *speaker* and *addressee* roles. We explicitly mark the participants in dialogue as the meaning of a sentence can change depending on who is saying what to whom. This will also determine the participant variables that are encoded deeper in the Dialogue-AMR structure. For instance, if there is a movement action we need to know what is being moved. In the example in Fig. 3, the addressee (a robot) is the moving thing and marked by the `:ARG0` role under *go-02* (`:ARG0 a`).

1.2.3 Tense and Aspect

In situated dialogue, we need to know *when* an action occurs relative to speech time. Equally, we need aspectual information such as the telicity of an event—whether an action has a bound endpoint and is completable or ongoing. This semantic information helps establish common ground and updates the participants on the location or progress of an action. We address these needs by adding tense and aspect to Dialogue-AMR.

We base our tense and aspect schema on Donatelli et al. (2018) with some modifications, using three temporal categories and five categories for aspectual annotation. The categories are summarized in Fig. 4:

TEMPORAL ANNOTATION	ASPECTUAL ANNOTATION
:time 1. (b / before :op1 (n / now)) 2. (n / now) 3. (a / after :op1 (n / now))	:stable +/- :ongoing +/- :complete +/- :habitual +/- :completable +/-

Fig. 4 Temporal and aspectual annotations

The temporal annotations relate to before, after, and during speech. In our example (Fig. 3c), temporal annotations are situated at the bottom of the Dialogue-AMR graph (denoted in purple) and sit under the action predicate (*go-02*). As this sentence is a command to move, the movement action will take place in the future, so the Dialogue-AMR receives the *future* label `(a/after :op1(n/now))`.

The aspectual annotations allow us to express information on the boundness of an event, such as whether an event has a clear endpoint or is in progress. A common aspectual annotation in our domain, `:completable +/-`, is a category we added to our schema to signal whether a future event has a clear end-goal that is executable for a robot. The example in Fig. 3c has a `:completable +` label denoted in red, which signals this moving event is completable because the distance to move forward is specified. If the sentence was only *move forward*, the label would alternatively be `:completable` since there is no specification for an end-point of travel.

1.2.4 Robot Concepts

Robot-concept relations are a fixed set of relations that correspond to robot behaviors. This is important for our domain because we need to represent a robot's limited action specification and normalize unconstrained natural language to one of these relations, as illustrated in Fig. 5.

Input	AMR	Dialogue-AMR
Turn left 90 degrees. *Make a left turn.*	turn-01	turn-01
Rotate left.	rotate-01	
90 degrees left.	:angle-quantity...	
Pivot 90 left.	pivot-01	

Fig. 5 Unconstrained natural-language input compared with a normalized representation in the Standard-AMR and, finally, mapped to a single robot-concept relation in the Dialogue-AMR

A command to turn left can be issued in many ways. Standard-AMR partially tames the syntactic variation from natural language but still preserves lexical items. For a turning event, this can be boiled down to some common actions: *turn*, *rotate*, and *pivot*. Dialogue-AMR takes another step toward smoothing over this variation by mapping these all to one robot-concept relation representing a rotation behavior: `turn-01`.

1.3 Parsing-Pipeline Approach

We have outlined the Dialogue-AMR schema and its suitability to the human–robot dialogue domain, especially with its critical additions of speaker intent, tense, and aspect. Obtaining Dialogue-AMR and integrating this representation into a dialogue system presents another challenge.

Here, we describe our two-step NLU parsing pipeline that transforms a Standard-AMR interpretation of a natural language utterance into Dialogue-AMR.

While there exist automatic AMR parsers that produce Standard-AMR graphs from text input, there are no Dialogue-AMR parsers. Thus, we are presented with two main directions: 1) build a Dialogue-AMR parser from scratch or 2) leverage existing resources.

At this stage, it is challenging to build a Dialogue-AMR parser because of a low-resource problem—we do not have enough training data in the form of manually annotated Dialogue-AMR graphs to train a parser from-scratch. Since the Dialogue-AMR schema has significant structural modifications and additions, we also cannot create Dialogue-AMR from training a Standard-AMR parser on in-domain data. This would only improve the Standard-AMR parser in our pipeline.

A more viable option is to use existing resources. Since Dialogue-AMR is closely related to a Standard-AMR, we considered the idea of a *graph-to-graph transformation*—converting a Standard-AMR graph into a Dialogue-AMR graph.

One possibility is to create a neural G2G to perform this conversion process. However, this would also require much training data in the form of input graphs and output graphs. Since Dialogue-AMR is a relatively new schema, there are not enough expert annotators to do this quickly and accurately. Additionally, within the schema's speech-act taxonomies some categories have little training examples found in the domain data—such as *questions* or *requests*, among other categories. Therefore, we would need to consider collecting more training data for an accurate end-to-end parser.

A final obstacle for a parser that converts Standard-AMR graphs into Dialogue-AMR graphs is that not all required information for a Dialogue-AMR can be gleaned from the Standard-AMR alone. We can take advantage of Standard-AMR as an initial interpretation of a sentence and the core propositional content, but it lacks the speaker's intent, tense, and aspect that we added to the Dialogue-AMR schema. Therefore, we hypothesized that the original text would provide extra linguistic information to fill these gaps. Tense (event time) might be signaled by morphological or other grammatical elements encoded in the text (e.g., -ed, will, and -ing). This information is lost in a Standard-AMR, which smooths over morphological and syntactic information. Speaker intent can also be predicted by the presence of certain words or syntactic structures. For the sentence, *Do you have any arms*, the *Do you* constructions can signal a speaker asking a question.

To that end, we present in Fig. 6 our G2G conversion approach, which takes both the sentence text and Standard-AMR graph as input and outputs a Dialogue-AMR.

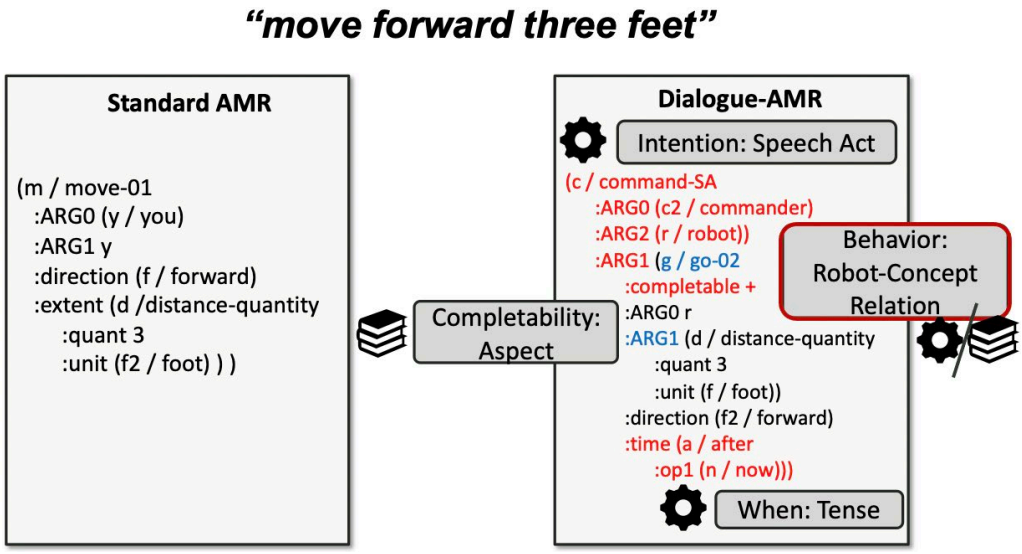

Fig. 6 Comparison of Standard-AMR and Dialogue-AMR with main augmented components: *speech act*, *tense*, *aspect,* and *robot-concept relation*

This approach relies on both rule-based and classifier-based methods to construct a Dialogue-AMR and perform a G2G transformation. At a high level, we use the original Standard-AMR graph of a sentence to preserve the core propositional content and use rules to manipulate the graph structure and perform slot-filling functions.

A group of classifiers predict information we cannot collect from a Standard-AMR graph alone, namely, **speaker intent** (roughly corresponding to the *speech act*) and **tense** (*when* the main action takes place). **Aspect** is assigned by rule-based combination of slot elements.

Additionally, we offer two distinct strategies for the determining the **robot-concept** (robot behavior): 1) a keyword-based approach and 2) a classifier-based approach. We restrict the root action of a Dialogue-AMR to a set of actions that a robot can execute.

2. Standard-AMR Parsing

An essential first step of our G2G conversion process is acquiring a Standard-AMR graph as an initial semantic interpretation. We therefore need an effective Standard-AMR parser to automatically produce a Standard-AMR for our NLU pipeline.

While there exist a variety of Standard-AMR parsers, we chose to focus on two open-source Standard-AMR parsers that serve as better starting points for our human–robot dialogue domain: one from Johns Hopkins University (Zhang et al. 2019) and one from Saarland University (Lindemann et al. 2019). Both parsers make use of Bidirectional Encoder Representations from Transformers (commonly, BERT) embeddings (Devlin et al. 2019) and are evaluated on Standard-AMR release data, which makes them more amendable to comparison against each other and our retrained models.

First, we trained the parsers on the AMR 2.0 and 3.0 corpora, which are the Standard-AMR releases, and on the Standard-AMR portion of the Dial-AMR corpus (Bonial et al. 2020). These AMR releases serve as a good baseline and mostly comprise text sources such as *The Wall Street Journal* and web discussion forums. However, since these data releases are not dialogue specific or contain instruction-heavy language, as is typical in our domain, we leverage the DialAMR corpus to train and evaluate on human–robot dialogue.

The DialAMR corpus includes 1122 instances of the Situated Corpus of Understanding Transactions (SCOUT[*]) and is annotated with Standard-AMR and corresponding Dialogue-AMR. The Standard-AMR subset of the DialAMR corpus, used for retraining the parsers, includes over 800 Standard-AMRs.

In total, we compare the two parsers—from Johns Hopkins and Saarland universities—trained on combinations of this data, yielding eight unique parsers. We evaluated these parsers on a *continuous-trial* data set—a 20-min experimental trial from SCOUT containing 304 utterances and representing a full human–robot interaction. The results are summarized in Fig. 7.

[*] SCOUT provides navigation domain language data with over 80 h of dialogue (Marge et al. 2016; 2017).

Parser	Training	P	R	F
JHU	Zhang et al.	AMR 2.0	.47	.77	.58
		2.0 + DialAMR	.73	.77	.75
		AMR 3.0	.52	.80	.63
		3.0 + DialAMR	.88	.89	.89
Saarland U	Lindemann	AMR 2.0	.53	.77	.63
		2.0 + DialAMR	**.92**	.94	**.93**
		AMR 3.0	.55	.81	.65
		3.0 + DialAMR	.91	**.95**	**.93**

Fig. 7 Evaluation (including precision [P], recall [R], and the harmonic mean of precision and recall, f-score [F]) of two AMR parsers and eight variants trained on combinations of AMR releases 2.0 and 3.0 and the DialAMR corpus

The best-performing parser, and therefore the parser to be integrated into the full NLU parsing pipeline, was the Saarland parser trained on AMR 3.0 and the DialAMR corpus. Training with the DialAMR corpus substantially boosts the performance of all parser varieties and especially the Saarland parsers. This is due to the highly repetitive instructional language in the SCOUT corpus, with people often saying phrases like *Move forward* and *Take a picture*. When adding 800 instances of the DialAMR training data, we reached an F-score of 0.93, which is a very high f-score for semantic parsing, where the ceiling (best) parsing performance is established by measuring how well expert human annotators can reliably reproduce the agreed-upon parses. The ceiling performance of humans is between 0.78 and 0.95, depending upon the complexity of the language, so our automatic parser performance is on par with human performance.

As part of our efforts to scale up to new domain, we then tested our best parser on instructional dialogue from Minecraft. Minecraft is a game where participants collaboratively build structures in a virtual environment, and the Minecraft Corpus (Narayan–Chen et al. 2019; Bonn et al. 2020) contains collaborative human–human dialogue in building structures. So, this domain is like the human–robot search and navigation domain in that it contains instructional language, albeit with different actions and behaviors.

For this evaluation, we tested our parser on 100 sequential instances of the Minecraft corpus and compared this with the manually assigned gold standard for this corpus. The performance drops to an overall F-score of 57, and all scores are about 35 points lower than the performance on the SCOUT data from the human–robot dialogue domain. Despite the similarity of instruction-giving domains, scaling to a new domain would require us to retrain the Standard-AMR parser.

3. Graph-to-Graph Conversion

Here, we outline the G2G conversion system from input to Dialogue-AMR output following the example utterance *Move forward three feet*. We also point to snippets of the code that relate to steps of the system to facilitate understanding and improving the code (see Fig. 8).

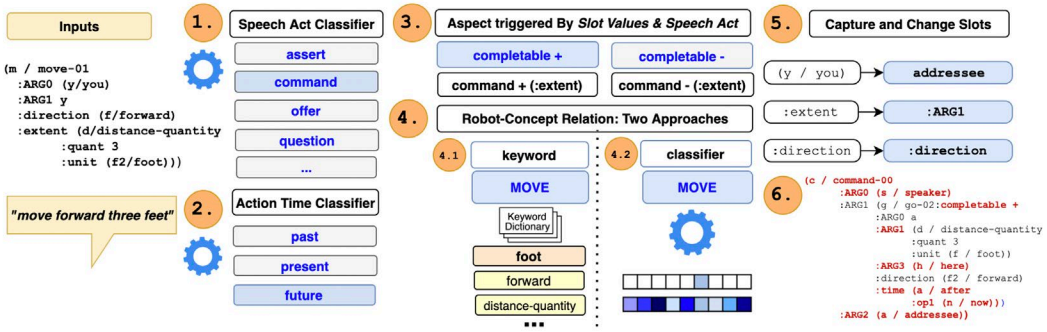

Fig. 8 Conversion system and main steps with typical example *Move forward three feet*: 1) speech act is classified as a *command*, 2) tense (action time) is classified as future event, 3) rule-based approach assigns: completable + as the aspect, 4) robot-concept is classified as a movement behavior (either through rule-based or classifier-based approach), and 5) graph is finalized through capturing and changing the appropriate slots

Our conversion system has five main sections, which leverage both rule-based and classifier-based methods. Figure 8, presenting the pipeline visually, can be read from left to right. On the left, we start with the two pieces of input, a Standard-AMR graph, and a text utterance. The Standard-AMR is provided either manually or automatically with an AMR parser. For our pipeline, we implement the Saarland AMR parser trained on AMR 3.0 and the DialAMR corpus (see Section 2).

3.1 Speech Act Classification

Following the diagram, the first step of the transformation process employs a Support Vector Machine (SVM) with token unigram features to predict the *speech act* (speaker intent) from the original text. The classifiers in this system are implemented in the scikit-learn library (Pedregosa et al. 2011). The pickled classifier accesses the weights stored in the *cl_weights* folder (see Fig. 9).

```
# ------------------------------------------------------------
# speech act classification
# ------------------------------------------------------------

    # utterance
    raw_utterance = utterance
    utterance = [utterance]

    # for GloVe embeddings from sci-kit learn
    # utterance = utterance

    # pickled data
    if args.domain == "botl":
        sa_pkl_filename = os.path.abspath("cl_weights/botl_domain/sa_pickle_model.pkl")
        sa_pkl_vec = os.path.abspath("cl_weights/botl_domain/sa_pickle_vec.pkl")

    if args.domain == "minecraft":
        sa_pkl_filename =
os.path.abspath("cl_weights/minecraft_domain/sa_pickle_model.pkl")
        sa_pkl_vec = os.path.abspath("cl_weights/minecraft_domain/sa_pickle_vec.pkl")

    # Load vectorizor
    with open(sa_pkl_vec, 'rb') as file:
        sa_vectorizer = pickle.load(file)

    # Vectorize test data
    # change to input string: must be iterable
    sa_test_X = sa_vectorizer.transform(utterance).toarray()
    # Load model
    with open(sa_pkl_filename, 'rb') as file:
        sa_pickle_model = pickle.load(file)

    # Make predictions
    speech_act_classification = sa_pickle_model.predict(sa_test_X)
    …
```

Fig. 9 Snippet of code for speech act classifier

After classification, the speech act label is stored as a slot of be added to the Dialogue-AMR and used for decision-making processes downstream. In the example displayed in Fig. 8, *Move forward three feet* is classified as a command and is given the label *command-00*.

3.2 Action Time Classification

The second step of the conversion is also a classification step. Here we predict the action time or, roughly, the *tense* of the action—whether the natural language utterance pertains to a *past*, *present*, or *future* action.

While it makes sense to use an off-the-shelf classifier for this three-label classifications, we built our own classifier to handle the specific challenges in our domain. While some features in the language correspond to traditional tense labels (e.g., *-ed* endings typically marking past tense or *will* marking future tense), this was not always the case. For instance, a common shorthand in our domain for a command to take and send an image is *"image"*, which refers to a *future* action. A traditional off-the-shelf tense classifier would not work in this case because *"image"* does not inherently signal future tense or provide any helpful morphological information. Additionally, our classifier takes the entire utterance as

input and not just a verb or the main action; not every utterance has a verb and other content and function words can help predict the action time (see Fig. 10).

```
# ------------------------------------------------------------------------
# tense classification
# ------------------------------------------------------------------------
    # print(pathlib.Path('tense_pickle_vec.pkl').resolve())
    # pickled data
    pkl_filename = os.path.abspath("cl_weights/pickle_model.pkl")
    pkl_vec = os.path.abspath("cl_weights/pickle_vec.pkl")

    # Load vectorizor
    with open(pkl_vec, 'rb') as file:
        tense_vectorizer = pickle.load(file)

    # Vectorize test data
    # change to input_string: iterable
    test_X = tense_vectorizer.transform(utterance).toarray()

    # Load model
    with open(pkl_filename, 'rb') as file:
        pickle_model = pickle.load(file)

    # Make predictions
    tense_classification = pickle_model.predict(test_X)

    # print("prediction", tense_classification)
    if tense_classification == 0:
        tense_cl = "(b/before :op1 (n/now))"
        tense_label = "past"
        # print(tense_label)

    if tense_classification == 1:
        tense_cl = "(n/now)"
        tense_label = "present"
        # print(tense_label)

    if tense_classification == 2:
        tense_cl = "(a/after :op1 (n/now))"
        tense_label = "future"
        # print(tense_label)
```

Fig. 10 Snippet of code for tense classifier

After classification, the action time (tense) is stored as a slot to be added to the Dialogue-AMR and used for determining aspect assignment downstream.

3.3 Aspect Assignment

The third step of the conversion is aspect assignment. We capture aspectual information with rule-based methods, rather than classifier-based methods, since we noticed consistent patterns associated with speech act, tense, and certain arguments. Therefore, we set up rules where particular combinations of speech act labels, tense labels, and the presence or absence of certain arguments of the robot's main action trigger an aspectual label. For instance, commands use the `:completable ±` label, indicating if the main action is goal-oriented and requires an end-point for execution.

One rule for a common *move* and *turn* command: If an argument conveying a clear end-point is present in the AMR—realized as the `:extent` slots (*move forward **three feet***) or the `:destination` slot (move to ***the room***)—the Dialogue-AMR received the `:completable +` label (see Fig. 11).

```
# inventory for completable slots for "go-02" actions and "turn-01" actions
    go_completable_slots = [":extent", ":destination", ":path", ":ARG4"]
    turn_completable_slots = [":direction", ":location", ":destination", ":ARG1",
":extent"]

    # arguments in amr to check for completable actions
    arguments = get_args(amr)

    ## assign apsect slot
    if sa == "command-00":
        if action == "go-02":
            completable = (any(x in arguments for x in go_completable_slots))
            if completable:
                aspect_role = ":completable"
                aspect_constant = "+"
            else:
                aspect_role = ":completable"
                aspect_constant = "-"
        elif action == "turn-01":
            completable = (any(x in arguments for x in turn_completable_slots))
            if completable and ":extent" in arguments:
                aspect_role = ":completable"
                aspect_constant = "+"
            else:
                aspect_role = ":completable"
                aspect_constant = "-"
        else:
            aspect_role = ":completable"
            aspect_constant = "+"
```

Fig. 11 Snippet of code demonstrating sample of aspect classification rules

Another rule requires a combination of speech act and tense information. For assertions within the present tense, the aspect label is `:ongoing + :complete −` label (an ongoing action); for assertions in the past tense, the aspect label is `:ongoing - :complete +` (past completed that is complete). See Fig. 12.

```
elif sa == "assert-00":
        if 'ready-02' in amr:
            aspect_role = ":stable"
            aspect_constant = "+"
        else:
            if tense_label == "present":
                aspect_role = ":ongoing"
                aspect_constant = "+"
                aspect_role_2 = ":complete"
                aspect_constant_2 = "-"
                #amr_slots["aspect"] = ":ongoing + :complete -"
            else:
                aspect_role = ":ongoing"
                aspect_constant = "-"
                aspect_role_2 = ":complete"
                aspect_constant_2 = "+"
```

Fig. 12 Snippet of code demonstrating aspect classification rules for assertions

3.4 Robot-Concept Relation Classification

In the fourth step, we classify the robot-concept relation and rely on two approaches: one keyword-based and one classifier-based.

3.4.1 Keyword-Based Approach

Our keyword-based approach works with a built-in dictionary where each robot behavior contains a set of keywords. A list of keywords is extracted from the utterance text (the tokens in the sentence) and the relations in the Standard-AMR and checked against the keyword dictionary of similar actions. If one is found, it triggers the robot-concept relation it maps to. This approach is an extended version of an earlier keyword approach (Abrams et al. 2020), which extracted the top root relation of the Standard-AMR as the keyword, instead of tokens and AMR slots. We discovered some problems in this earlier approach. The same root relation could correspond to multiple robot-concept relations. For instance, the top root relation in a Standard-AMR for the *move* and *go*, typically `move-01` and `go-02`, can refer to both a movement and turning behavior. Our earlier model encountered this overlap and we incorrectly classified utterances like *Move right 45 degrees* (rotation behavior).

To combat these issues, we created our current keyword dictionary informed by a data-driven analysis. We examined utterance tokens, AMR relations, and arguments that are truly unique to robot concepts, based on our analysis. For example, the predicate `move-01` may not map strictly to a movement behavior, but the token, *feet*, does. Similarly, we found the token *degrees* to map strictly to a rotation behavior. After gathering data and analyzing the data, we removed keywords that signaled multiple relations and added our new keywords.

The keyword dictionary is stored as a separate json file and accessed within dialamr.py. Two version exist in the code for the *bot language* and the *Minecraft* domain and can be accessed through flags on the command line when running the G2G conversion system with the keyword approach (see Fig. 13).

```
{
...
"turn-01": ["turn-01",
            "degrees",
            "turn-01",
            "angle-quantity",
            "pivot-01",
            "rotate-01"],
"help-01": ["help-01",
            "help"],
"instruct-01": ["instruct-01"],
…
}
```

Fig. 13 Sample of key-word dictionary constructed in a json file for keys *turn-01*, *help-01,* and *instruct-01*

Following the example text of *Move forward three feet*, the keyword-based approach would determine this to be a movement behavior (mapped to `go-02`). *Feet* or *foot*, in this case, would be the unique keyword that belongs to this relation.

3.4.2 Classifier-Based Approach

For robot-concept classification (Fig. 14), we employ a classifier-based approach to have a more robust option, especially when scaling to new domains and handling novel utterances in our domain. The classifiers are SVMs with simple unigram features. Additionally, while we experimented with different vectorization methods, we implemented one-hot encoding into the model.

```
# ------------------------------------------------------------------------
# robot-concept classification
# ------------------------------------------------------------------------

    # pickled data
    if args.approach == "cl":
        if args.domain == "botl":
            robot_concept_pkl_filename =
os.path.abspath("cl_weights/botl_domain/dialrel_pickle_model.pkl")
            robot_concept_pkl_vec = os.path.abspath("cl_weights/botl_domain/dialrel_pickle_vec.pkl")

        if args.domain == "minecraft":
            robot_concept_pkl_filename =
os.path.abspath("cl_weights/minecraft_domain/dialrel_pickle_model.pkl")
            robot_concept_pkl_vec =
os.path.abspath("cl_weights/minecraft_domain/dialrel_pickle_vec.pkl")


        # Load vectorizor
        with open(robot_concept_pkl_vec, 'rb') as file:
            robot_concept_vectorizer = pickle.load(file)

        # Vectorize test data
        # change to input_string: iterable
        #test_X = robot_concept_vectorizer.transform(utterance)

        # for one-hot vectorization
        test_X = robot_concept_vectorizer.transform(utterance).toarray()

        # Load model
        with open(robot_concept_pkl_filename, 'rb') as file:
            robot_concept_pickle_model = pickle.load(file)

        # Make predictions
        robot_concept_classification = robot_concept_pickle_model.predict(test_X)
```

Fig. 14 Snippet of code for robot-concept classification

The classifier should be more robust to the keyword-based approach but can be potentially misled by words in the utterance that tend to overlap with other robot-concept relations. For a simple example like *Move forward three feet*, the model should predict a movement behavior (`go-02`) from the text as input.

3.5 Capture and Change Slots

In the final step of the transformation process (labeled as Step 5 in the pipeline, see Fig. 8), we form the Dialogue-AMR by capturing slots that we need to keep from the Standard-AMR to preserve the core meaning, change slots as necessary, and append the new slots gathered from the previous steps. Overall, this final step encompasses smaller steps that use different methods.

To form a Dialogue-AMR graph, we first start with a core AMR graph and add slots to "build" it up. This core AMR graph is typically just the robot-concept relation:

```
(g /go-02)
```

We collect slots from the Standard-AMR that represent the essential content of the utterance that should sit under the robot-concept relation (e.g., `:direction (f2/forward)`. These slots might have to be renamed for a specific Dialogue-AMR, such as `:extent` to `:ARG1`.

Once a minimal AMR is formed, the remaining steps rely on the Penman python package to perform the heavy lifting of graph formation and manipulation. The minimal AMR (string in PENMAN notation) is formed and *decoded* into a graph. The remaining slots can be appending to the graph in the form of triples (e.g., `[('t', ':instance', 'turn-01')]`).

The remaining slots are added to the graph through the `add_triple()` and `add_top_triple()` functions. The slots include the speech act, tense, and aspect information determined through previous steps. Additionally, the speaker and addressee roles are added to the top of the graph, right under the speech act. These arguments are always set as `:ARG0` and `:ARG2` for speaker and addressee, respectively. But the speaker and addressee names can be changed and set beforehand.

The completed graph is then *encoded*—to use the terminology from the Penman package—from a graph back into an AMR string in PENMAN notation. When encoding the graph, the Penman package checks that variables are unique to their relation and co-refer when they need to, especially since new variables can be added that coincide with other ones. If a new slot (in the form of a triple) is added to the graph (`('s', ':instance', 'speaker')`), the variable will be reassigned a unique identifier when the variable is already present (`('s2', ':instance', 'speaker'), ('s', ':instance', 'spin-01')`).

4. Conclusions

We have walked through the Dialogue-AMR conversion system and have outlined, in detail, various components of the pipeline. We have explained the need for a semantic representation suitable for natural language understanding in the human–robot navigation domain; AMR offers a useful framework but is insufficient in several critical areas of meaning. Our solution is Dialogue-AMR, an augmented version of AMR with *speech act*, *aspect*, and *tense* information. The conversion system ultimately facilitates a two-step NLU process—converting text and Standard-AMR into a Dialogue-AMR. We also provided details of a full pipeline, including the AMR parser and the rule-based and classifier-based steps of the conversion system. Some snippets of code are provided to explain how the step is implemented.

5. References

Abrams M, Bonial C, Donatelli L. Graph-to-graph meaning representation transformations for human-robot dialogue. Proceedings of the Society for Computation in Linguistics 2020. Association for Computational Linguistics; 2020. p. 250–253.

Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koeh P, Palmer M, Schneider N. Abstract meaning representation for sembanking. Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse; 2013. p. 178–186.

Bonial C, Donatelli L, Abrams M, Lukin SM, Tratz S, Marge M, Artstein R, Traum D, Voss C. Dialogue-AMR: abstract meaning representation for dialogue. Proceedings of the 12th Language Resources and Evaluation Conference; 2020; Marseille, France. European Language Resources Association. p. 684–695.

Bonial C, Donatelli L, Lukin SM, Tratz S, Artstein R, Traum D, Voss C. Augmenting abstract meaning representation for human-robot dialogue. Proceedings of the First International Workshop on Designing Meaning Representations. Association for Computational Linguistics; 2019. p. 199–210.

Bonn J, Palmer M, Cai Z, Wright-Bettner K. Spatial AMR: expanded spatial annotation in the context of a grounded Minecraft corpus. Proceedings of the 12th Language Resources and Evaluation Conference. European Language Resources Association; 2020. p. 4883–4892.

Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 4171–4186.

Donatelli L, Regan M, Croft W, Schneider N. Annotation of tense and aspect semantics for sentential AMR. Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG); 2018. p. 96–108.

Lindemann M, Groschwitz J, and Koller A. Compositional semantic parsing across graphbanks. In Proc of ACL; 2019. p. 4576–4585.

Marge M, Bonial C, Byrne B, Cassidy T, Evans AW, Hill SG, Voss C. Applying the Wizard-of-Oz technique to multimodal human-robot dialogue. In RO-MAN 2016: IEEE International Symposium on Robot and Human Interactive Communication; 2016.

Marge M, Bonial C, Foots A, Hayes C, Henry C, Pollard K, Artstein R, Voss C, Traum D. Exploring variation of natural human commands to a robot in a collaborative navigation task. Proceedings of the First Workshop on Language Grounding for Robotics; 2017. p. 58–66.

Narayan-Chen A, Jayannavar P, Hockenmaier J. Collaborative dialogue in Minecraft. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics; 2019. p. 5405–5415.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–2830.

Zhang S, Ma X, Duh K, Van Durme B. AMR parsing as sequence-to-graph transduction. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics; 2019.

List of Symbols, Abbreviations, and Acronyms

AMR	Abstract Meaning Representation
G2G	graph to graph
JHU	Johns Hopkins University
NLP	Natural Language Processing
NLU	natural language understanding
SCOUT	Situated Corpus of Understanding Transactions
SVM	Support Vector Machine

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
2 (PDF)	DEVCOM ARL FCDD RLC IT C BONIAL C VOSS