In presenting this dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I agree that the Library of the University shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate School when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

Han He

(Signature)

NOTICE TO BORROWERS

Unpublished dissertations deposited in the Emory University Library must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this dissertation is :

Han He
Department of Mathematics and Computer Science
Emory University
Atlanta, GA 30322

The director of this dissertation is :

Jinho D. Choi
Department of Mathematics and Computer Science
Emory University
Atlanta, GA 30322

Users of this dissertation not regularly enrolled as students at Emory University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this dissertation for the use of patrons are required to see that each user records here the information requested.

Name of user	Address	Date	Type of use (Examination only or copying)

Computational Structures as Neural Symbolic Representation

by

Han He
Doctor of Philosophy

Computer Science and Informatics

Jinho D. Choi, Ph.D.
Advisor

Fei Liu, Ph.D.
Committee Member

Liang Zhao, Ph.D.
Committee Member

Song Feng, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Computational Structures as Neural Symbolic Representation

by

Han He
M.S., University of Houston-Clear Lake, 2018

Advisor : Jinho D. Choi, Ph.D.

Abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy
in Computer Science and Informatics

Department of Mathematics and Computer Science

2023

Abstract

Although end-to-end neural models have been dominating Natural Language Processing for both performance and flexibility, critics have recently drawn attention to their poor generalization and lack of interpretability. Conversely, symbolic paradigms such as Abstract Meaning Representation (AMR) are humanly comprehensible but less flexible. In response, we propose **E**xecutable **A**bstract **M**eaning **R**epresentation (**EAMR**) as a reconciliation of both paradigms. EAMR is a neural symbolic framework that frames a task as a program, which interactively gets generated, revised and executed. In our novel definition, execution is a sequence of transforms on AMR graphs. Through a hybrid runtime, EAMR learns the automatic execution of AMR graphs, yet it also allows for the integration of handcrafted heuristics, knowledge bases and APIs. EAMR can be used in many applications such as dialogue understanding and response generation.

Computational Structures as Neural Symbolic Representation

by

Han He
M.S., University of Houston-Clear Lake, 2018

Advisor : Jinho D. Choi, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy
in Computer Science and Informatics

Department of Mathematics and Computer Science

2023

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Jinho D. Choi, for his invaluable guidance, support, and encouragement throughout my dissertation journey. His expertise, patience, and dedication have been crucial to my success, and I could not have completed this work without his mentorship. I am also grateful to the Emory NLP lab for providing the resources and support necessary to complete this dissertation.

I would also like to extend my heartfelt thanks to my committee members, Prof. Fei Liu, Prof. Liang Zhao, and Dr. Song Feng, for their insightful feedback and valuable suggestions that have greatly improved the quality of this dissertation. Their expertise and guidance have been instrumental in shaping my research and helping me to achieve my academic goals.

Special thanks go to those Japanese novelists and musicians, whose artworks motivated and impelled me forward. It was the "Fate" novel series written by Kinoko Nasu and Gen Urobuchi that helped me get out of the lowest point in my life. Whenever I encounter a research problem, a famous quote by the protagonist Emiya (Archer), "What you must imagine is always that you, yourself, are the strongest", always comes to mind. On countless sleepless nights, it was the song "Last Stardust" performed by Aimer that kept me company as I burned the midnight oil. As the theme song for Emiya, its lyrics taught me the idea of searching for hope and light in a world that can often be dark and difficult to navigate. In my daily research, it was the "Fate" original soundtrack composed by Yuki Kajiura that helped me to stay focused on every battle with deadlines. Every time I turn the volume up to the maximum, her melody fills my body with maximum energy too.

Finally, I would like to thank my family and friends for their unwavering support, encouragement, and understanding throughout this process. Their love and encouragement have sustained me through the ups and downs of this journey, and I could not have completed this work without them.

Thank you all for your contributions to my academic and personal growth.

To Emiya, the hero of justice that leads my way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the history of artificial intelligence, two rival paradigms have vied for supremacy. Symbolic paradigm was dominant in the 20th century, while neural approaches have recently become established as the most impactful paradigm [14]. Both paradigms have strengths and weaknesses, and they are complementary to each other. Symbolic approaches explicitly produce intermediate representations which are language-like and amenable to human understanding. However, symbolic representations suffer from the famous symbol grounding problem [18] since they are handcrafted. In contrast, neural representations are learned end-to-end which are flexible while lacking interpretability and robustness. The shortcomings of neural models align with the strengths of symbolic methods, facilitating a hybrid paradigm called neural-symbolic which leverages the advantages of both approaches [2, 17, 26, 40, 9]. However, early neural symbolic methods require an expert design of the

symbolic language in each specific domain, restricting their coverage.

In this dissertation, we aim at designing a novel neural symbolic "runtime" of the Abstract Meaning Representation [6], which is a general semantic framework that represents a sentence as a semantic graph consisting of concepts and relations. We choose AMR instead of others due to its wide coverage of semantic phenomenons and domain-agnostic nature. Thanks to its conceptual and propositional character, AMR is amenable to human understanding and heuristic rules.

Since AMR is designed to be generic, there is no deterministic execution rule or path to manipulate its nodes or relations like a program. In response, we propose the first definition of *execution* for AMR in Section 3.1, which depicts the graph transform of AMR depending on language context and graph topology. Our definition is agnostic of the runtime such that it can be implemented using either neural, symbolic, or hybrid approaches. Specifically, we present a symbolic and a neural baseline followed by our neural symbolic EAMR in Section 3.4. We further propose two novel algorithms to learn EAMR using sole unlabeled text corpora. The execution of EAMR can be learned end-to-end in an unsupervised fashion, yet each execution step is interpretable for humans.

Experiments on DailyDialog [30] show that EAMR performs comparably well with a strong baseline while it demonstrates stronger interpretability. Our proposed EAMR can be applied in many tasks, especially in settings that require incorporating external data sources such as knowledge bases or external APIs. The potential applications of EAMR are discussed in Section 5. We hope EAMR can promote neural model interpretability research with computational linguistic structures.

Chapter 2

Background

2.1 Literature Review

Neural symbolism made its first debut on synthetic visual questions answering, where the representational capacity of deep learning and the compositional linguistic structure of the text are simultaneously exploited to answer synthesized questions paired with images [2, 25, 10]. Later, Gupta et al. [17] propose to parse open-domain questions as executable programs grounded on documents. Wolfson et al. [37] introduce a meaning representation for the decomposition of questions expressed through natural language. Following the same line, Khot et al. [26] propose to solve decomposed questions using existing systems. Most recently, BINDER is proposed to map a question into a programming language whose functions can be either directly executed or indirectly solved by prompting a large language model [9]. The

representations used in these neural symbolic approaches are domain-specific, whereas we take the challenge to harness the wide coverage of semantics via a domain-agnostic representation, AMR.

Our work is also closely related to neural program synthesis, a task employing neural models to automatically translate natural language descriptions to a program that satisfies the user intent [16]. Existing works mostly focus on generating domain-specific codes, such as regular expressions [28], SQL queries [39, 7], and calendar APIs [3]. Different from theirs, our studied AMR is not naturally an executable programming language. Program synthesis research targets improving the accuracy of parsing natural language into programming languages while the execution runtime is deterministic. In our study, the proposed runtime of EAMR can be either deterministic or non-deterministic, and the later can be learned end-to-end while preserving the interpretability of a deterministic one.

The last line of related research is AMR extension for cross-sentence understanding. Initial attempts [31, 13] merge AMR nodes that represent the same entity, formulating it as an AMR coreference resolution task. Recently, graph neural networks are applied to coreference-resolved AMR graphs for dialogue relation extraction and response generation [4]. Beyond coreference

resolution, our EAMR supports much broader types of AMR transforms.

2.2 Past Research

Our past research is a journey challenging the limit of neural models for linguistic structure prediction, such as tagging and parsing. We were the first to challenge state-of-the-art taggers and parsers using non-finetuned BERT [11] embeddings on core NLP tasks [19]. Our BERT models outperform the previously best-performing models by 2% on average (7.5% for the most significant case) on all tasks and datasets. Once finetuned, BERT was further found to boost the performance of multilingual enhanced dependency parsing [20].

Though the accuracy of large pretrained language models (PLMs) is excellent, their speed is relatively slow. To reduce model complexity, we presented three works. In the first work, we propose a novel Levi graph AMR parser [21] by combining tokens, concepts, and labels as one input to a transformer to learn attention matrices which are used to predict all elements in AMR graphs. Our Levi graph decoder reduces the number of decoder parameters by 45% yet gives similar or better performance. In the second one, we challenge multi-task learning (MTL) by sharing one PLM on

5 tasks for faster speed [22]. Surprisingly, our experiments depict that MTL models underperform ones trained individually. To reveal the mysteries of MTL, we propose a dynamic pruning method of attention heads to detect essential heads for each task. Our experiments reveal that all five tasks rely on almost the same set of attention heads, leading to the interference of features. Thus, we propose the *Stem Cell Hypothesis*, likening these talented attention heads to *stem cells*, which cannot be fine-tuned for multiple tasks that are very distinct. Our hypothesis is further validated by a set of novel probing methods. In the most recent work under review, we challenge seq2seq models for linguistic structure generation without external decoders. Our best models perform comparably or better than the state-of-the-art for all tasks, lighting a promising future for seq2seq models for generating non-sequential structures.

In the same line, we improve the efficiency of seq2seq models by unleashing their potential using prompting and constrained decoding [23]. Sequence-to-Sequence (S2S) models have achieved remarkable success on various text generation tasks. However, learning complex structures with S2S models remains challenging as external neural modules and additional lexicons are often supplemented to predict non-textual outputs. We present a systematic

study of S2S modeling using contained decoding on four core tasks: part-of-speech tagging, named entity recognition, constituency and dependency parsing, to develop efficient exploitation methods costing zero extra parameters. In particular, 3 lexically diverse linearization schemas and corresponding constrained decoding methods are designed and evaluated. Experiments show that although more lexicalized schemas yield longer output sequences that require heavier training, their sequences being closer to natural language makes them easier to learn. Moreover, S2S models using our constrained decoding outperform other S2S approaches using external resources. Our best models perform better than or comparably to the state-of-the-art for all 4 tasks, lighting a promise for S2S models to generate non-sequential structures.

These works have established strong taggers and parsers, facilitating our research on neural symbolic approaches. We developed the first interactive calendar assistant [24] as the initial attempt which inspires the EAMR idea in this dissertation.

Chapter 3

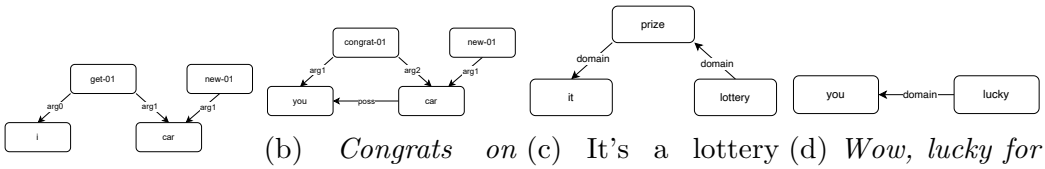
Approach

The following sections are organized as follows: First, we formally define the execution of EAMR. Then, we present two baselines to stimulate ideas. At last, we propose two novel learning algorithms for EAMR to enable end-to-end training.

3.1 Definition

Execution In a document or a dialogue consisting of n utterances, denote the i-th utterances and corresponding AMR graph as x_i and y_i respectively. The execution is then defined as a process that transforms $\mathbf{y}_{<i}$ to y_i. E.g., transforming Figure 3.1a, 3.1b, and 3.1c to Figure 3.1d in a dialogue setting.

Our definition of AMR execution differs from the execution of any programming language in its non-recursive fashion. Thus, the i-th AMR execution would require all past AMR graphs as inputs, while the runtime of a pro-



(a) I got a new car (b) *Congrats on your new car* (c) It's a lottery prize (d) *Wow, lucky for you*

Figure 3.1: AMR graphs for a dialogue. Figure3.1a and 3.1c are user utterances, Figure3.1b and 3.1d are *system responses*.

gramming language would substitute a function with its return value once it gets evaluated. The non-recursive design is necessary for two reasons. Firstly, information in all previous sentences could be referred to again and again. Depending solely on the previous AMR graph is impossible to recall information from several sentences ago. Secondly, the runtime of AMR could be non-deterministic as the response depends on factors missing from the conversation history (e.g., commonsense knowledge, speaker personality), leading to a non-optimal execution path when given only the last AMR graph.

3.2 Baselines

In this section, we present two baselines of EAMR.

AMR-to-AMR Inspired by the success of language models pretrained to predict the next token [35, 29], we consider a seq2seq baseline trained with "language model" objective on a corpus of automatically parsed AMR graphs.

This baseline learns end-to-end AMR execution though its prediction lacks interpretability, which we address in our proposed methods.

Text-to-Text To measure the performance loss caused by using automatically parsed AMR graphs, a vanilla seq2seq baseline is trained to generate the response conditioned on previous contextual turns. Later, the state-of-the-art BART-AMR parser [5] trained on AMR 3.0 is employed to parse the generated responses to AMR graphs for comparison with other models.

3.3 Dialog Context Representation

For a dialogue, multiple training or test instances are created as pairs of context and response. Each instance uses m consecutive utterances with corresponding AMR graphs as the dialogue context and the following utterance with its AMR graph as the response. Given a dialogue context of m utterances and the corresponding AMR graphs, we create textual and graphical representations respectively.

Text Representation A special token `utt-j` is inserted to the front of the $(m - j + 1)$-th utterance to indicate the boundary of utterances. We use descending numbers such that the last utterance is always labeled with `utt-1`.

As the last utterance is usually the most closely related one to the response, an invariant label is assigned to stabilize the textual presentation regardless of the number of context utterances.

Graph Representation Similar to the textual representation, the AMR graph of each utterance is attached to a numbered concept `utt-j` in descending order. Then, all `utt-*` concepts are attached to a root node `multi-utterance` similar to the way AMR groups multiple sentences `snt-*` under a `multi-sentences` root. An exemplar graph representation of the first 3 AMR graphs from Figure 3.1 is illustrated below.

```
(a8 / multi-utterance
:utt3 (a4 / get-01
          :ARG0 (a6 / a5)
          :ARG1 (a3 / car
                    :ARG1-of (a10 / new-01)))
:utt2 (a1 / congratulate-01
          :ARG1 (a12 / you)
          :ARG2 (a2 / car
                    :ARG1-of (a9 / new-01)
                    :poss a12))
:utt1 (a11 / prize
           :mod (a7 / lottery)
           :domain (a5 / it)))
```

3.4 Step-by-Step Execution

The runtime of most programming languages would feature step-by-step execution for debugging purposes. Accordingly, we propose the concept of execution steps to break down a complex execution into a sequence of small steps, where each step is more interpretable to humans and some of them can even be directly guided by heuristics.

Execution Step Given a history of AMR graphs $\mathbf{y}_{<i}$, an *execution step* e_{ij} at step j is defined as an edit action that modifies $\mathbf{y}_{<i}^{j-1}$ into $\mathbf{y}_{<i}^{j}$, where $\mathbf{y}_{<i}^{0} = \mathbf{y}_{<i}$. Each e could be one of the six types: {create, delete, replace} $\times$ {node, edge} and a special `NOP` (no operation) indicating the end of execution. Once the last edit $e_{ik} = \texttt{NOP}$ is predicted, the execution finishes with $y_i \leftarrow \mathbf{y}_{<i}^{k}$. Specifically, the 6 types of edits are described below.

CreateNode(u, c) A new node with variable name being u and concept being c is created.

DeleteNode(u) The node with variable name u together with all its incoming edges and outgoing edges are deleted.

ReplaceNode(u, c) Replace the concept of u with c.

CreateEdge(u, r, v) Create an edge from u to v with the role being r.

DeleteEdge(u, v) Delete the edge from u to v.

ReplaceEdge(u, r, v) Replace the role of the edge from u to v with r.

3.5 Edit Oracle

Intuitively, end-to-end execution is akin to running a program where step-by-step execution can be analogous to step-wise debugging which could provide humans with diagnostic of problems regarding the model and the data. Our step-wise execution can also be related to the rich literature of transition-based parsing [38, 32], which inspires us to propose the following two learning algorithms.

Static Oracle A static oracle is an algorithm producing the optimal gold edit sequence that transforms y^{j-1} to y^j. While prior work has already proposed Graph Edit Distance (GED), a generalization of string edit distance which corresponds to the minimum cost edit sequence between two attributed relational graphs [36, 12], it limits the labels of nodes to a predefined vo-

cabulary. In AMR, the number of concepts is unbounded which urges us to propose a new oracle algorithm, *Smatch Edit*, to approximate the optimal graph edits. To reduce the computation cost, we would prefer shorter edit sequences given that the cost of each edit is uniform. Therefore, we seek to reuse as many nodes and edges as possible by emitting edits according to the optimal node mapping M that maximizes the Smatch score [8] of the context graph and response graph. As depicted in Algorithm 1, new nodes from $y^j - M$ are created, unmapped nodes from $y^{j-1} - M$ are deleted, and the concepts of each pair of matched nodes are made consistent using the one from y^j if they are different. Once node edits are applied, two graphs with identical nodes are obtained. Upon them, edges are compared and edited in a similar way.

where ReplaceNode(u) replaces the concept of u in y^{j-1} with its concept in y^j, and ReplaceEdge(u, v) replaces the role of edge (u, v) in y' with its role in y^j. Though AMR is defined as a connected graph, connectivity is only a necessary property for linearization. Before linearization, a subroutine is applied to maintain the graph connectivity by bridging the largest connected component with other smaller components using an `orphan` edge.

An example of applying a sequence of edits obtained from Smatch score

Algorithm 1: Smatch Edit Oracle

Function `Oracle`(y^{j-1}, y^j)**:**

- edits $\leftarrow$ []
- $M \leftarrow \underset{M'}{\text{argmax}}\, \text{Smatch}(y^{j-1}, y^j)$
- **foreach** $u \in y^j - M$ **do**
  - append CreateNode(u) to edits
- **foreach** $u \in y^{j-1} - M$ **do**
  - append DeleteNode(u) to edits
- **foreach** $u \in M$ **do**
  - **if** *concepts of* u *in* y^{j-1} *and* y^j *are different* **then**
    - append ReplaceNode(u) to edits
- $y' \leftarrow$ apply edits to y^{j-1}
- $E' \leftarrow$ edges of y'
- $E \leftarrow$ edges of y^j
- **foreach** $(u, r, v) \in E - E'$ **do**
  - append CreateEdge(u, r, v) to edits
- **foreach** $(u, r, v) \in E' - E$ **do**
  - append DeleteEdge(u, v) to edits
- **foreach** $(u, v) \in E \cap E'$ **do**
  - **if** *roles of* (u, v) *in* E *and* E' *are different* **then**
    - append ReplaceEdge(u, v) to edits
- **return** edits

to the AMR graph in Figure 3.1a is illustrated below.

Input: AMR graph for "I got a new car".

```
(a2 / get-01
    :ARG0 (a3 / I)
    :ARG1 (a1 / car
              :ARG1-of (a4 / new-01)))
```

Step 1: NodeDeletion(a3)

```
(a2 / get-01
    :ARG1 (a1 / car
              :ARG1-of (a4 / new-01)))
```

Step 2: NodeReplacement(a2, congratulate-01)

```
(a2 / congratulate-01
    :ARG1 (a1 / car
              :ARG1-of (a4 / new-01)))
```

Step 3: NodeInsertion(b4, you)

```
(a2 / congratulate-01
    :ARG1 (a1 / car
              :ARG1-of (a4 / new-01))
    :orphan (b4 / you))
```

Step 4: EdgeReplacement(a2, :ARG2, a1)

```
(a2 / congratulate-01
    :ARG2 (a1 / car
              :ARG1-of (a4 / new-01))
    :orphan (b4 / you))
```

Step 5: EdgeInsertion(a2, :ARG1, b4)

```
(a2 / congratulate-01
    :ARG2 (a1 / car
              :ARG1-of (a4 / new-01))
    :ARG1 (b4 / you))
```

Step 6: EdgeInsertion(a1, :poss, b4)

```
(a2 / congratulate-01
    :ARG2 (a1 / car
              :ARG1-of (a4 / new-01)
              :poss b4)
    :ARG1 (b4 / you))
```

Dynamic Oracle Similar to its spirit in dependency parsing [15], a dynamic oracle in EAMR provides a set of optimal edits for every intermediate graph configuration $\mathbf{y}_{<i}^{j}$ at execution step j. Different from the oracles in dependency

parsing, Smatch Edit guarantees that all configurations can lead to the gold graph including configurations that deviate from the optimal oracle. In such cases, the dynamic oracle will run Smatch Edit again from the given configuration to generate a new edit sequence as a remedy such that the model is less sensitive to error propagation. To provide teaching signals, the model is trained in a predict-then-train loop, where the model is trained on edit sequences that are constantly and dynamically updated by Smatch Edit using the predicted graphs after each step or each epoch.

3.6 AMR Oracle

As our early experiments show poor results for edit prediction, we further propose an extension to the edit-based oracles to directly predict edited AMR graphs. The extension is straightforward: given an edit sequence obtained from either static or dynamic oracle, a sequence of edited AMR graphs can be derived by sequentially applying the edit sequence to the initial AMR graph. Instead of teaching a model to predict the edit sequence, we can alternatively train a model to predict a later graph given an earlier graph in the sequence.

For prediction, an initial context graph is fed in and the generated AMR graph is then fed back as inputs for several iterations till the number of itera-

tions exceeded a predefined number, or the output graph stops transforming. Due to the costly computation, we implement AMR oracle with 2 iterations at most.

Chapter 4

Experiments

Thanks to our generic definition, EAMR can be applied to a wide range of cross-sentence text understanding and generation tasks. In this initial study, we mainly consider dialogue response generation as it is close to the next AMR graph generation in EAMR.

4.1 Dataset

We experiment on the DailyDialog benchmark [30] comprised of conversations about our daily life. Specifically, we perform the dialogue response generation task which aims at generating a system response given the dialogue history.

We parse each utterance to a silver AMR graph using the state-of-the-art parser [5] which scored 84.3 on the AMR 3.0 benchmark [27]. For a long utterance consisting of multiple sentences, we individually parse each sentence and merge them later using `multi-sentence` and `snt-*` nodes since

we observed a significant performance drop on long inputs and the parser is limited to a maximum of 1024 input tokens.

We set the context window size to 7 turns since the average number of turns in DailyDialog is 8. This number is also shown to be effective in training the chatbot Meena on public domain social media conversations [1].

4.2 Metric

Conventionally, BLEU is used as an automatic evaluation metric for dialogue response generation. However, EAMR cannot be directly evaluated using BLEU because node mapping is required to compare two AMR graphs. We tried to employ the state-of-the-art AMR2Text model which scored 49 BLEU [5] to generate responses given our model predictions, but the final performance was largely degraded by error propagation of that model. As an alternative, we propose to evaluate EAMR using Smatch, a metric for semantic feature structures, on predicted AMR graphs and the target AMR graphs.

4.3 Training

We use the seq2seq model as the backbone implementation as it is flexible for new concept generation which the latest Graph Edit Networks [34] is not capable of. Specifically, we finetune the large version of pretrained AMR-BART [5] on several combinations of inputs (text representation, graph representation) and outputs (edits or AMR graphs). Special tokens, including `multi-utterance`, `orphan` etc., are added to the BART vocabulary and get learned from scratch.

Models equipped with static oracles are trained for 33 epochs using learning rate 1e-5 and a copy of weights at the 30th epoch is saved as the initial weight for the dynamic oracle models. Following that, we train dynamic oracle models for 3 additional epochs using a smaller learning rate 5e-6 in a predict-then-train loop as described in Section 3.4. Each finetuning epoch takes 1 hour on an NVIDIA RTX A6000 GPU while generating intermediate AMR graphs on the training data takes 7 hours.

4.4 Results

As shown in Table 4.1, our experimental results are grouped into 3 blocks.

In the first block, we use only one representation per model. Compared

Model	Smatch
text $\rightarrow$ text [4]	34.8
text $\rightarrow$ amr	31.8
amr $\rightarrow$ amr	30.5
amr $\rightarrow$ edit	16.1
amr + text $\xrightarrow{\text{stc}}$ amr	31.5
amr + text $\xrightarrow{\text{dyn}}$ amr	32.6
amr + text $\xrightarrow{\text{stc}}$ amr + text	30.8
amr + text $\xrightarrow{\text{dyn}}$ amr + text	31.4

Table 4.1: Smatch scores of response AMR on DailyDialog. text $\rightarrow$ text is our implementation of seq2seq model evaluated on parsed AMR graphs on generated responses. "stc" and "dyn" means static and dynamic oracle respectively.

to text $\rightarrow$ text, using parsed silver AMR graphs as the source of teaching signals (text $\rightarrow$ amr) hurts the Smatch score by 3 points. Substituting the input text to silver AMR again degrades the Smatch score by another 1.3 points, suggesting that it is challenging to exploit noisy data for learning AMR execution and inspiring us to exploit more features. Unfortunately, amr $\rightarrow$ edit does not perform well, which might be due to the following 2 reasons. Firstly, the intermediate graph after each edit is not explicitly encoded, causing the model to lose track of the current configuration in a long execution sequence. Secondly, the variable names used in edit sequences originate from linearized AMR graphs, and they might not fit well into the

context of edits.

In the second block, we experiment with both the text representation and the graph representation described in Section 3.3. Specifically, we use pairs of special tokens, `<s>` and `<AMR>`, to mark the boundary between utterances and the AMR graph. Compared to amr $\rightarrow$ amr, adding the text representation of dialogue context (amr + text $\xrightarrow{\text{stc}}$ amr) gains 1.0 Smatch improvement, though it still performs 0.3 lower than text $\rightarrow$ amr. It agrees with the finding that learning from noisy data is challenging. Equipped with dynamic oracle, amr + text $\xrightarrow{\text{dyn}}$ amr outperforms its static oracle equivalent by 1.1 points, showing the effectiveness of dynamic oracle on graph level. Similar findings are made in the iterative tagging system designed for Grammatical Error Correction [33].

In the last block, we further experiment with response generation in a multi-task learning setting. However, the performance of neither static nor dynamic oracle gets further improved by the text outputs, which might be due to the left-to-right order of generation. Since the noisy AMR graphs are generated first, the later text generation might be misled which in turn negatively impacts the AMR graph generation.

Chapter 5

Application

In previous sections, we have shown the effectiveness of dynamic oracle in AMR execution. However, how far one execution step proceeds is determined by the model which is still not controllable and not interpretable. In this section, we experiment with human control over the execution steps.

5.1 Controllable Execution

Specifically, we break down the execution into 2 stages: concept summarization and relation generation. Concept summarization is a process of filtering useful concepts from the dialogue context that might be talked about in the later response utterance. Once concepts are filtered, the execution moves on to the next stage which connects concepts with semantic relations.

We implement controllable execution by splitting the edit sequences obtained from Smatch Edit into 2 stages, which we call concept summarization

and relation generation respectively. The first stage contains CreateNode, DeleteNode and ReplaceNode while the second stage contains CreateEdge, DeleteEdge and ReplaceEdge. We call AMR graphs after applying the edits from the summarization stage "concept pool". A seq2seq model with static oracle is trained to transform the initial graph into the concept pool, then into the ultimate response graph in a pipeline fashion. A dynamic oracle model instead predicts a concept pool and then transforms it into the ultimate response graph. The design of a summarization stage is similar to the summary graph proposed by Liu et al. [31]. However, our concept pool is not as a complete AMR graph as theirs because it contains less typology and its structure could also evolve in the later generation stage.

The results of amr + text $\rightarrow$ amr models on DailyDialog are listed in Table 5.1.

Controllable Model	Smatch
amr + text $\xrightarrow{\text{stc}}$ amr	28.2
amr + text $\xrightarrow{\text{dyn}}$ amr	32.0

Table 5.1: Smatch scores of controllable AMR execution on DailyDialog.

The controllable amr + text $\xrightarrow{\text{stc}}$ amr model performs poorly compared to its uncontrollable variant in Table 4.1, which could be attributed to the

error propagation between the 2 execution stages. However, the performance drop of controllable amr + text $\xrightarrow{\text{dyn}}$ amr is less significant, confirming that dynamic oracle is effective in mitigating error propagation.

5.2 Case Study

Using the controllable amr + text $\xrightarrow{\text{dyn}}$ amr model and its outputs from the last section, we present an application scenario where we can control the response generation given a topic keyword or concept. Assuming the topic of the response is given, and it is identical to the root node of the response AMR graph, we replace the root of the "concept pool" with it before performing the relation generation stage. In a control group, we directly replace the roots of the final AMR graphs after relation generation. The results are listed in Table 5.2.

Topic Control Stage	Smatch
no control	32.0
after generation	34.3
after summarization	34.6

Table 5.2: Smatch scores of topic controlled models.

Using "gold" root concepts after summarization outperforms using them after the whole generation by 0.3 points, suggesting that the controllable

amr + text $\xrightarrow{\text{dyn}}$ amr model can adjust some concepts and relations accordingly given the new root. An example of this scenario is illustrated below.

Input: AMR graph for "I got a new car", "Congrats on your new car" and "It's a lottery prize".

```
(a8 / multi-utterance
:utt3 (a4 / get-01
          :ARG0 (a6 / a5)
          :ARG1 (a3 / car
                    :ARG1-of (a10 / new-01)))
:utt2 (a1 / congratulate-01
          :ARG1 (a12 / you)
          :ARG2 (a2 / car
                    :ARG1-of (a9 / new-01)
                    :poss a12))
:utt1 (a11 / prize
           :mod (a7 / lottery)
           :domain (a5 / it)))
```

Concept Pool:

```
(a3 / good-02
    :orphan (a4 / really)
    :orphan (a5 / it)
    :orphan (a6 /
            request-confirmation-91))
```

Replaced Concept Pool:

```
(a3 / lucky
    :orphan (a4 / really)
    :orphan (a5 / it)
    :orphan (a6 /
            request-confirmation-91))
```

Output: You're really lucky, aren't you?

```
(a3 / lucky
    :degree (a4 / really)
    :ARG1 (a5 / you)
    :ARG1-of (a6 /
            request-confirmation-91))
```

Chapter 6

Conclusion

In this dissertation, we propose EAMR, a neural symbolic AMR framework with excessive interpretability. EAMR features a novel Smatch Edit oracle which can be used both statically and dynamically. The dynamic oracle variant of EAMR can be further made controllable using two execution stages: summarize and then generate. EAMR can be readily applied on controllable text summarization and task oriented dialog system through extending the AMR graph with hierarchies of domain related nodes that carry information of keywords and API functions. We hope EAMR can be the starting point for exciting further research of neural symbolic AMR.

The major limitations of EMAR are three folds. Firstly, its interpretability is limited to atomic operations, i.e., node and edge operations. However, the way of human reasoning are more structural which typically requires sub-graph operations as a whole. Secondly, the inference speed of seq2seq

models are slow, which in turn hinders further application of dynamic oracles. Lastly, the AMR graph is agnostic of certain aspects that are crucial for dialogue response generation, such as verb tense and text style.

In the future, we would like to explore a more effective way of encoding AMR graphs and performing graph-to-graph transduction. EAMR will become more attractive if its performance and interpretability can be better balanced.

Appendix A

Hyper-Parameter Configuration

The hyper-parameters used in our models are described in Table A.1.

BART	
name	bart-large
encoder layers	12
decoder layers	12
dropout	0.25
Adam Optimizer	
lr	1e-5
ϵ	1e-8
epochs	33
warm up	0
Generation	
length penalty	1
max length	1024
beams	1

Table A.1: Hyper-parameters settings.

Bibliography

[1] Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

[2] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016. doi: 10.1109/CVPR.2016.12.

[3] Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy Mc-

Govern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. Task-Oriented Dialogue as Dataflow Synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, 09 2020. ISSN 2307-387X. URL `https://doi.org/10.1162/tacl_a_00333`.

[4] Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. Semantic representation for dialogue modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4430–4445, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.342. URL `https://aclanthology.org/2021.acl-long.342`.

[5] Xuefeng Bai, Yulong Chen, and Yue Zhang. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland, May 2022. Association for

Computational Linguistics. doi: 10.18653/v1/2022.acl-long.415. URL https://aclanthology.org/2022.acl-long.415.

[6] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-2322.

[7] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160.

[8] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the*

Association for Computational Linguistics (Volume 2: Short Papers), pages 748–752, 2013.

[9] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding language models in symbolic languages. *ArXiv*, abs/2210.02875, 2022.

[10] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *Conference on Robot Learning*, pages 53–62. PMLR, 2018.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

[12] Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen,

and Horst Bunke. Approximation of graph edit distance based on hausdorff matching. *Pattern Recognition*, 48(2):331–343, 2015. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2014.07.015. URL https://www.sciencedirect.com/science/article/pii/S003132031400274X.

[13] Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. End-to-end AMR coreference resolution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.324. URL https://aclanthology.org/2021.acl-long.324.

[14] Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019.

[15] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL https://aclanthology.org/C12-1059.

[16] Sumit Gulwani, Alex Polozov, and Rishabh Singh. *Program Synthesis*, volume 4. NOW, August 2017. URL https://www.microsoft.com/en-us/research/publication/program-synthesis/.

[17] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *International Conference on Learning Representations (ICLR)*, 2020.

[18] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990. ISSN 0167-2789. doi: https://doi.org/10.1016/0167-2789(90)90087-6. URL https://www.sciencedirect.com/science/article/pii/0167278990900876.

[19] Han He and Jinho Choi. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert, 2020. URL https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS20/paper/view/18438.

[20] Han He and Jinho D. Choi. Adaptation of multilingual transformer encoder for robust enhanced Universal Dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 181–191, Online, July 2020. Association for

Computational Linguistics. doi: 10.18653/v1/2020.iwpt-1.19. URL https://aclanthology.org/2020.iwpt-1.19.

[21] Han He and Jinho D. Choi. Levi graph AMR parser using heterogeneous attention. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 50–57, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.iwpt-1.5. URL https://aclanthology.org/2021.iwpt-1.5.

[22] Han He and Jinho D. Choi. The stem cell hypothesis: Dilemma behind multi-task learning with transformer encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5555–5577, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.451. URL https://aclanthology.org/2021.emnlp-main.451.

[23] Han He and Jinho D. Choi. Unleashing the true potential of sequence-to-sequence models for sequence tagging and structure parsing. *Transactions of the Association for Computational Linguistics*, 2023.

[24] Han He, Song Feng, Daniele Bonadiman, Yi Zhang, and Saab Mansour.

Dfee: Interactive dataflow execution and evaluation kit. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[25] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017.

[26] Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Text modular networks: Learning to decompose tasks in the language of existing models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1264–1279, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.99. URL `https://aclanthology.org/2021.naacl-main.99`.

[27] Kevin Knight, Lauren Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneifer. Abstract meaning representation (amr) annotation release 1.0. *Web download*, 2014.

[28] Nate Kushman and Regina Barzilay. Using semantic unification to

generate regular expressions from natural language. North American Chapter of the Association for Computational Linguistics (NAACL), 2013.

[29] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

[30] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I17-1099.

[31] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A.

Smith. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1114. URL https://aclanthology.org/N15-1114.

[32] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/W06-2933.

[33] Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bea-1.16. URL

https://aclanthology.org/2020.bea-1.16.

[34] Benjamin Paaßen, Daniele Grattarola, Daniele Zambon, Cesare Alippi, and Barbara Hammer. Graph edit networks. In Shakir Mohamed, Katja Hofmann, Alice Oh, Naila Murray, and Ivan Titov, editors, *Proceedings of the Ninth International Conference on Learning Representations (ICLR 2021)*, 2021. URL https://openreview.net/forum?id=dlEJsyHGeaL.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[36] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, 1983. doi: 10.1109/TSMC.1983.6313167.

[37] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *Transactions of the As-*

sociation for Computational Linguistics, 8:183–198, 2020. URL https://aclanthology.org/2020.tacl-1.13.

[38] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France, April 2003. URL https://aclanthology.org/W03-3023.

[39] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

[40] Zhaocheng Zhu, Mikhail Galkin, Zuobai Zhang, and Jian Tang. Neural-symbolic models for logical queries on knowledge graphs. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27454–27478. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhu22c.html.