CHAPTER 2

Background

2.1 Meaning Representations

A meaning representation is a structured way of articulating the individuals, actions, times, locations, causes, and manners involved in an event. This representation can be embodied through various logical forms, including Discourse Representation Theory (DRT; Kamp and Reyle 1993), Abstract Meaning Representations (AMR; Wang et al. 2020; Bevilacqua et al. 2021), Minimal Recursion Semantics (MRS; Horvat et al. 2015), BabelNet Meaning Representation (BMR; Martínez Lorenzo et al. 2022). A meaning representation serves as a connection between intricate linguistic subtleties and our inherent non-linguistic understanding of the world. It can be regarded as a structured format that encapsulates the essence of linguistic input. Throughout this process, we assume that every linguistic structure contains certain content or information suitable for conveying the state of the world. Meaning representations find utility in tasks such as information extraction (Rao et al., 2017; Solawetz and Larson, 2021), machine translation (Song et al., 2019; Li and Flanigan, 2022), sentiment analysis (Marasović and Frank, 2018), and various other applications (Pan et al., 2015; Kapanipathi et al., 2021).

In this dissertation, our focus is on Discourse Representation Structures (DRSs). DRSs are formal meaning representations based on DRT, which are recursive first-order logical frameworks that contain discourse referents (entities introduced in discourse) and capture the interrelationships between them. Compared with alternative meaning representations such as AMR, DRSs exhibit greater semantic expressive power and cover a wide range of
linguistic phenomena, including quantification, negation, reference resolution, comparison, discourse relations, and presuppositions. A conventional depiction of Discourse Representation Structure (DRS), known as the box-format of DRS, offers an intuitive and readable layout but lacks convenience for modeling purposes. Therefore, DRS is often post-processed into a format more suitable for processing by contemporary neural network models. The main focus of this dissertation is on the clause-format DRS (van Noord et al., 2018a) and the variable-free DRS (Bos, 2023), which are available in the Parallel Meaning Bank (PMB; Abzianidze et al. 2017). In this section, we provide a detailed overview of the above formalisms.

2.1.1 Discourse Representation Structures

Discourse Representation Structures (DRSs) are structured representations of meaning that stem from the framework of DRT. DRT is a formal system dedicated to investigating meaning representations through formal semantics. It has been widely studied in the field of linguistic semantics and is particularly apt for compositional semantics. DRT was introduced for dealing with issues in the semantics and pragmatics of anaphora and tense (Kamp, 1981; Geurts et al., 2020; Kamp and Reyle, 1993). Expressions in DRT, called DRSs, have a recursive structure and are usually depicted as boxes. Next, we will introduce the format of the specific dialects of DRS in the PMB that we will be using throughout the whole dissertation.

A basic box in a DRS consists of two components: discourse referents and conditions. An upper part of a box contains a set of discourse referents while the lower part lists a conjunction of atomic or compound conditions over these referents. Discourse referents are also called variables, which are indicators of discourse elements, such as persons or events in a discourse. Concepts, roles, constants and comparison operators are collectively referred to as the conditions, which assert information over these discourse elements. For example, $x_1$ and $x_2$ in Figure 2.1 (a) are two discourse referents that indicate a male person and a dog respectively. The conditions then assert that $x_1$ is a male person named Tom, $x_2$ is a dog, and that $x_1$ played with $x_2$. The concepts are represented by WordNet synsets (Fellbaum, 1998), each containing
2.1. Meaning Representations

three elements: lemma, part-of-speech and sense number. WordNet synsets can cover four parts of speech: noun, verb, adjective and adverb. For example, male.n.02 and dog.n.01 are used to represent the male person and the dog. An event in a DRS has its own discourse referent (\(e_1\)) and the concept invoked in this event is represented using a WordNet synset whose part-of-speech is a verb, such as play.v.05 in Figure 2.1 (a). In addition, the tense is usually introduced by the main verb in the sentence, and it is always modeled as a time period (\(t_1\)) and represented by a WordNet synset time.n.08.

The roles the participants play in the event are represented by VerbNet (Kiper et al., 2008), which are two-place predicates used to relate the event to the event participants. For example, the verb play introduces the role Agent, which means acting on something, while Time is used to denote the tense in which the verb occurs. In addition, constants are also important elements in DRSs, which are used to represent discourse direction (“speaker”, “hearer”), questions (“?”) and names (“Tom”), quantities (“40”) and tense (“now”) (van Noord, 2021). Comparison operators are used to relate and compare discourse referents by the terms, which can be either variables or constants, such as \(t_1 \prec \text{now}\) (temporally precedes) in Figure 2.1 (a), which used to express the past tense. If \(t_1 = \text{now}\), it means that the time period is equal to the constant “now”, which is used to indicate that the event in DRS occurs in the present tense.

A complete DRS is commonly viewed as a set of interconnected boxes that can be nested within each other. When another basic box is nested within a box, the lower part of the full DRS box is also called a complex condition. Complex conditions are used to indicate logical relations between the sets of conditions and can represent their scope. Formally, DRSs and DRS-conditions are defined as follows:

**Syntax of DRSs**

1. If \(U\) is a set of discourse referents, and \(C\) is a set of DRS-conditions, then \((U, C)\) is a DRS;

2. If \(B\) and \(B'\) are DRSs, and \(D\) is a discourse relation, then \(D(B, B')\) is a DRS;
2. Background

Figure 2.1: Example for DRSs for sentence (a) “Tom played with a dog”, (c) “Tom didn’t play with a dog” and (d) “Tom can’t play with a dog”. In (b), we model the presupposition of sentence (a) in a separate box.

3. Nothing else is a DRS.

Syntax of DRS-Conditions

1. If $P$ is a concept, and $x$ a first-order variable, then $P(x)$ is a DRS-condition;

2. If $x$ is a first-order variable, and $y$ is a first-order variable or a constant, and $O$ is a comparison operator, then $xOy$ is a DRS-condition;
Figure 2.2: Example for DRSs for sentence “Tom didn’t play with a dog. He slept in the bed.”

3. If \( R \) is a role, and \( x \) and \( y \) are first-order variables, then \( R(x, y) \) is a DRS-condition;

4. If \( B \) is a DRS, then \( \neg B \), \( \Box B \) and \( \Diamond B \) are DRS-conditions;

5. Nothing else is a DRS-condition.

In addition, the connected boxes can be used to model presupposition, and the presupposition in a separate box outside of the main box of the DRS (Van der Sandt, 1992). For example, the conditions for Tom in figure 2.1 (c) are modeled outside the negation, because the sentence implies that a male named Tom exists, whether he played with a dog or not. Similarly, in Figure 2.1 (b), we give an example for the expression of an affirmative sentence “Tom played with a dog”. The meanings represented in Figure 2.1 (b) and Figure 2.1 (a) are equivalent, the difference is that the presupposition is modeled in a separate box. Additionally, we also give an example in Figure 2.1 (d) where the DRS contains the possibility expression. It can be interpreted as "There exists a male named Tom for whom it is impossible to be an agent in a playing event”.

In principle, the current DRS format can handle multi-sentence documents by using explicit discourse relations to connect different (possibly
nested) boxes. Each box has its own identifier \((b_1, b_2, \text{etc.})\), so the relations between the boxes can be indicated. These relationships draw inspiration from the rhetorical relations found in Segmented DRT (Asher, 1993; Asher and Lascarides, 2003). For example, given a document “Tom didn’t play with a dog. He slept in the bed.”, the discourse relation for this document is \textsc{continuation} (see Figure 2.2). Two sentences separated by periods or commas represent a box respectively. In addition, common discourse relations also include \textsc{contrast} (He might be young, but he’s trustworthy) and \textsc{consequence} (I’m hungry, so I’m going to get something to eat). In brief, DRS is a more expressive formalism compared to other meaning representations, as it explicitly models scope (such as negation and quantification) as well as presuppositions, and has the capability to manage multi-sentence documents. No other semantic representation has previously exhibited the capacity to encompass as many semantic phenomena as DRS does and has available annotated corpora. The DRS data we will be using in this dissertation is from the Parallel Meaning Bank (PMB; Abzianidze et al. 2017), so we will describe this corpus below.

\textbf{Parallel Meaning Bank} This is a semantically annotated parallel corpus, with English serving as the pivot language. Each document within the PMB contains both English text and various representation formats in DRS. Additionally, there are translations of the English text into other languages like Italian, German, Dutch, Chinese, and Japanese. The creation of this corpus relies on cross-lingual projection: automatically produced (and manually corrected) semantic annotations for English sentences are mapped onto their word-aligned translations, assuming that the translations are meaning-preserving. The goal of PMB is to achieve a language-neutral final DRS for a text, which aims to coordinate various semantic phenomena in a single formalism.

There are different layers of semantic annotation in PMB, primarily including: (i) segmentation of the text in sentences and lexical items; (ii) syntactic parsing with Combinatory Categorial Grammar; (iii) universal semantic tagging; (iv) symbolization; and (v) compositional semantic analysis based on Discourse Representation Theory (Abzianidze et al., 2017). These layers
2.1. Meaning Representations

utilize statistical models trained in a semi-supervised manner. Subsequently, the results from these layers are inputted into the rule-based semantic parser Boxer (Bos, 2008), yielding the final DRS. It’s important to note that the objective is to create annotations that encapsulate the most likely interpretation of a sentence. The employed annotation models are all language-neutral, devoid of any ambiguities or under-specification techniques.

By incorporating "Bits of Wisdom" (Bos et al., 2017), human annotators can correct machine-generated output. The annotations are not on the level of the final meaning representation but focus on correcting intermediate layers. These corrections serve as data for training better models and create a gold standard annotated subset of the data. Annotation quality is defined per layer and language at three levels: bronze (fully automatic), silver (automatic with some manual corrections), and gold (fully manually checked and corrected). There are many different release versions of PMB. Specifically, we will be working with PMB release 3.0.0 (Chapter 3 and 5) and PMB release 4.0.0 (Chapter 4, 6, 7 and 8). Detailed descriptions of these data sets will be provided in the respective chapters.

2.1.2 Clause-format DRS

Although DRSs are usually displayed in a recursive box-format, which is intuitive and easy to read, it is difficult for machine learning models to learn. Therefore, DRSs are usually post-processed into clause-format, which is more suitable for machine learning than box-format because it has a simple, flat structure and facilitates partial matching of DRSs, which is useful for evaluation (van Noord et al., 2018b). It is a linear representation of a set of clauses, where each clause is the smallest unit representing discourse referents, DRS-conditions, and discourse structure. This process is executed by applying the labels for DRSs as introduced in Venhuizen (2015) and Venhuizen et al. (2018), and deconstructing the recursive structure of DRS by labeling them with the DRS they are present in. The examples of DRSs in the box notation are presented at the top of Figure 2.3, and the result of translating DRSs to sets of clauses is shown at the bottom of Figure 2.3. In a clausal form, distinct variables are denoted by distinct variable names.
and the reverse holds as well. As a result of this, before transforming a DRS into a clausal form, diverse discourse referents present in the DRS need to be assigned distinct variable names. To enhance the legibility of semantic representation formats, a variety of letters are assigned to variables: the letters x, e, s, and t are respectively employed for discourse referents signifying individuals, events, states, and time, while b is used for variables denoting DRS boxes. This approach facilitates the restoration of the original box notation of a DRS from its clausal form, while significantly simplifying the matching process between clause forms for evaluation (van Noord et al., 2018a).
The box-format of DRS in Figure 2.3 consists of two main boxes \((b2\) and \(b5\)) and two presuppositional boxes \((b1\) and \(b3\)). Note that \(b2\) introduces negation via a condition \(\neg b4\) with a nested box \(b4\). The conditions of \(b4\) represent unary and binary relations over discourse referents that are introduced by \(b4\) and \(b2\) and the presuppositional DRSs \((b1\) or \(b3\)). Conversion from the box notation to the clause form and vice versa is transparent: discourse referents, conditions, and discourse relations in the clausal form are preceded by the label of the box they occur in. Notice that the variable letters in the semantic representations are automatically set and they simply serve for readability purposes. Since each logical operator carries its own scope, the count of these operators serves as a minimum indication of the number of scopes within a meaning representation. Beyond logical operators, scopes are introduced through presupposition triggers such as proper nouns or pronouns.

DRS acts as a scoped meaning representation, integrating word senses, thematic roles, and the list of operators, to form the final product of our semantically annotated corpus. While a DRS in the clause-format is a flat version of the standard box notation, the presence of explicit scopes (boxes) and scopal operators like negation within DRSs contributes to the heightened complexity of their clauses. The DRS clause, which varies in length from three to four, is capable of incorporating three variables and contains two different types of variables for scopes and discourse referents. Let’s first look at the definition of DRS clauses. There are five different types of clauses (we use terms to denote constants or variables):

**Definition of DRS Clauses**

1. If \(b\) is a label for a DRS, and \(x\) is a discourse referent, then "\(b\ \text{REF} \ x\)" is a DRS clause — we call this a **referent clause** for \(x\) in \(b\);
2. If \(b\) is a label for a DRS, \(x\) is a discourse referent, and \(P\) is a concept, then "\(b\ P \ x\)" is a DRS clause — we call this a **concept clause** for \(x\) in \(b\);
3. If \(b\) is a label for a DRS, and \(x\) is a discourse referent, \(t\) is a term, and \(O\) is a comparison operator, then "\(b\ O \ x \ t\)" or "\(b\ O \ t \ x\)" is a DRS clause — we call this a **comparison clause** for \(x\) in \(b\).
4. If $b$ is a label for a DRS, $x$ is a discourse referent, $t$ is a term, and $R$ is a role, then "$bRxt$" or "$bRtx$" is a DRS clause — we call this a role clause for $x$ in $b$;

5. If $b$ and $b'$ are labels for DRSs, and $D$ is a discourse relation, then "$bD\ b'$" is a DRS clause — we call this a discourse clause for $b$;

6. Nothing else is a DRS clause.

Figure 2.3 shows an example of the clause-format of DRS, which strictly follows the formal definition of DRS provided above. For example, the tense-related information ($b\ \text{REF}\ t$) is encoded in a clausal form with three additional clauses, which express a WordNet concept ($b\ \text{time.n.08}\ t$), semantic role ($b\ \text{Time}\ e\ t$) and a comparison operator ($b\ \text{EQU}\ t\ "\text{now}"$). Note that the number of referent clauses (REF clauses) in clause-format is always the same as the number of all upper box references in the full DRS. Based on the clause-format DRS, tree-format of DRS and graph-format of DRS were proposed by Liu et al. (2018) and Fancellu et al. (2019), respectively. However, the conversion process is usually complicated, and the converted structure is not conducive to understanding, making it difficult to reproduce and promote. Consequently, in this dissertation, we will not introduce or utilize these formats. Our focus will remain on working with clause-format DRS in Chapter 3 and Chapter 5.

2.1.3 Variable-free DRS

Variable-free Discourse Representation Structures, also known as Simplified Box Notation (SBN), is a new format of DRS recently proposed by Bos (2023), which is a representational variant of clause-format DRS where explicit variables are eliminated from the meaning representation. Unlike clause-format DRS, which uses variables to distinguish different discourse references and utilizes the co-occurrence of variables to depict their relationships, SBN employs the physical distance between discourse references to represent the connection relationship between them (Bruijn, 1972). The physical distance can be represented by using an index (such as $+1$ and $-2$), which has the
2.1. Meaning Representations

<table>
<thead>
<tr>
<th>x1</th>
<th>e1</th>
<th>t1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>male.n.02(x1)</td>
<td>Name(x1, &quot;tom&quot;)</td>
<td>time.n.08(t1)</td>
<td>t1 &lt; &quot;now&quot;</td>
</tr>
<tr>
<td>hit.v.01(e1)</td>
<td>Theme(e1, x2)</td>
<td>Patient(e1, x1)</td>
<td>Time(e1, t1)</td>
</tr>
<tr>
<td>baseball.n.01(x2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

male.n.02 Name Tom % Tom
time.n.08 < now % was
hit.v.01 Time -1 Patient -2 % hit
Theme +1 % by
baseball.n.01 % a baseball.

Figure 2.4: Positive indices can always be eliminated from variable-free DRS using role inversion, but this comes at the cost of alignment precision (from Bos 2023).

ability to bind in both backward and forward directions, encompassing discourse structure within its scope, as Figure 2.4 shows.

A variable-free DRS comprises sequences of concepts (one-place predicates), relations (two-place predicates), or structural constraints (Bos, 2023). These concepts serve as bindings for first-order entities, serving a dual role: (1) introduce a discourse referent, and (2) establish the type of the discourse referent with a conceptual one-place predicate. In practice, concepts are represented by WordNet and formally ordered based on the order of the words in the phrase, sentence, or text that is under analysis. This alignment between natural language and DRS proves pivotal for manual annotation (ensuring readability and verification) as well as for machine learning techniques in natural language processing. A relation, referred to as a "role", initiates with an uppercase character and establishes a connection between two conceptual predicates. Relations can be ordered in such a way that the predicates come first, followed by the relation instantiated with negative indices. While this imposes a serious constraint on the ordering, it can be efficiently achieved by reversing the roles: for any role R, R(X, Y) and ROf(Y, X) are equivalent. Roles lacking the Of suffix are interpreted as “event X has Y as R”, and roles with the Of suffix represent inverted roles and are rephrased
Figure 2.5: Negation in box-format DRS and corresponding variable-free DRS (from PMB-00/0023).

as Y is R of X. For example, in Figure 2.4, variable-free DRS can be represented as “the hitting event has a baseball as Theme” or “the baseball is the Theme of the hitting event”. From Figure 2.4, we can also observe the influence of role inversion on alignment. Normally, to make the representation of the variable-free DRS more canonical, the role always follows the concept that the part of speech is a verb.

Figure 2.4 shows the translation from a box-format of DRS to variable-free DRS, which is expressed as a basic SBN of a sequence of concepts and relations without any structural constraints. The structural constraints are used to provide information about discourse structure, also known as discourse relations. For a complex SBN, there is at least one structural constraint in the representation. In DRT, these structural constraints are expressed using logical operators and make DRS recursive. The approach adopted by SBN is based on introducing explicit discourse structure markers into the representation, effectively segmenting the sequence of concepts and roles into two distinct discourse units (Bos, 2023). Notably, in the context of SBN, these markers for discourse structure are uniformly capitalized to prevent any potential confusion between concepts and relations. An illustration of SBN with negation is provided in Figure 2.5. In the case of negation, the term "NEGATION" serves as a marker, cleaving the meaning
2.1. Meaning Representations

within the SBN into two distinct segments: the sequence before, and the sequence after the marker. Broadly, when employing $n$ markers within an SBN representation, the resulting division yields $n + 1$ separate discourse units.

Compared with clause-format DRS, variable-free DRS greatly reduces the sequence length of semantic representations, which not only streamlines the alignment between meaning representations and sentences but also facilitates the learning of semantic representations for long texts by machine learning models. Simpler representations are often more readable and perspicuous, not only facilitating further insight into the expressive power of logic languages but also making manual annotation or correction of DRSs less cumbersome. A meaning representation without variables is also easier to transform into a directed acyclic graph structure (as shown in Figure 2.6) and integrate into the syntax-semantics interface, especially to convert the output of a syntactic parser into a formal meaning representation (Poelman et al., 2022). In the upcoming chapters, we will focus on working with variable-free DRS in Chapter 4, 6, 7 and 8.

Figure 2.6: SBN in graphical notation for sentence “A person is not eating at a table”, corresponding to Figure 2.5.
2. Background

2.1.4 DRS vs. Alternative Formalisms

There are several semantically annotated corpora available. In this section, we will provide a comprehensive discussion comparing various alternative meaning representation formalisms.

One of the popular semantic formalism is Abstract Meaning Representations (AMR; Banarescu et al. 2013). An example AMR is shown in Figure 2.7. There are a number of significant differences between AMR and our DRS variants. In general, the main differences between DRS and AMR can be divided into two main categories: "surface representation format" and "intrinsic expressive power". First, DRS concepts and roles are represented using WordNet and VerbNet respectively, while AMR only relies on PropBank (Palmer et al., 2005) for verbs, and nouns have no explicit grounding. Additionally, AMR employs wikification (Cucerzan, 2007) to ground named entities, a component absent in DRS. These distinctions are categorized as differences in the surface representation format. Regarding the differences in intrinsic expressive power, in short, DRS is a more expressive formalism than AMR. DRS can model more semantic phenomena, including quantification, negation, comparison, presupposition and discourse relations. It is worth mentioning that DRS can handle multi-sentence documents through explicit discourse relations, whereas AMR is primarily designed for sentence-level representation, although it can currently handle some multi-sentence situations (O’Gorman et al., 2018). Another difference is that DRS can also explicitly model tense, a feature not found in AMR, where the same meaning representation is used for events in the past and events occurring in the future (van Noord, 2021).

Clause-format DRS closely resembles the triple notation used in AMR. However, due to the inclusion of scope in DRS, their clauses exhibit greater complexity compared to AMR triples. DRS clauses vary in arity, typically ranging from three to four, in contrast to the fixed length of AMR triples. Furthermore, DRS clauses involve two distinct types of variables for scopes and discourse referents, whereas AMR triples employ just one type. Lastly, clause-format DRS tends to be approximately twice as extensive as AMR for conveying the same semantic representation of sentences, both in terms of
the number of clauses and the count of unique variables (van Noord, 2021).

In addition to the two popular meaning representations AMR and DRS mentioned above, there are also novel semantic representation such as Minimal Recursion Semantics (MRS) and BabelNet Meaning Representation (BMR). Here, we provide a concise introduction for them. MRS serves as a descriptive language for first-order object language formulas with generalized quantifiers. It facilitates the straightforward articulation of grammatical constraints governing lexical and phrasal semantics, encompassing the principles of semantic composition (Copestake et al., 1997). Underspecified representations in MRS comprise elementary predications and handle scoping constraints (Niehren and Thater, 2003). To elaborate, elementary predications are object language formulas with placeholders where additional formulas can be inserted, and handle constraints specify the limitations on how these formulas can be interconnected. In contrast to
2. Background

DRS, which primarily focuses on a logical structure to represent semantic information in a logical form, including quantifier scopes, relations, etc., MRS adopts a graphical structure to represent semantic information in a lightweight manner using a set of features and constraints.

BMR is an interlingual formalism designed to transcend language-specific constraints (Martínez Lorenzo et al., 2022). It leverages the extensive multilingual semantic resources provided by BabelNet (Navigli and Ponzetto, 2010) and VerbAtlas (Di Fabio et al., 2019). This emerging language-independent semantic formalism completely abstracts from syntax, serving as a lexical-semantic representation that facilitates integration across diverse languages. Additionally, Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport 2013) emerged as a cross-lingual annotation formalism that connects words in a sentence through language-independent semantic relations. This graph-based semantic formalism, similar to AMR, operates independently of a syntactic foundation. It can be conceptualized as a multi-layered formalism, where each layer specifies the relations it encodes. In addition to the above-mentioned semantic representations, there are also formalisms such as Universal Meaning Representation (UMR; Gysel et al. 2021), currently under development.

2.2 Neural Models

In this part, we give a detailed description of the artificial neural networks that we will be using throughout the dissertation. Neural networks are inspired by the structure and function of the brain and use a large number of artificial neuron connections to model complex relationships between inputs and outputs or to explore patterns in data (Hopfield, 1982). Neural networks automatically transform specific inputs into specific outputs given a set of functions and weights that can be learned by analyzing training examples.

Among the earliest proposed neural models, the Feed-Forward Neural Network (FFNN; Bebis and Georgiopoulos 1994; Jain et al. 1996) occupies a prominent position. It consists of multiple layers of neurons, each layer contains several neurons, and each neuron can receive signals from the neu-
2.2. Neural Models

rons of the previous layer and propagate them to the next layer. An example of this network is shown in Figure 2.8. The layers of an FFNN can be divided into three categories according to their functions: input layer, hidden layer, and output layer (Anderson and McNeill, 1992; Paola and Schowengerdt, 1995; Warner and Misra, 1996). The first layer called the input layer, is used to receive raw data input, which can be a sequence of word embeddings or character embeddings for text. The last layer is the output layer, which is responsible for generating the final output of the network, and its number of neurons usually depends on the type of task. The middle layer contains one or more layers, called hidden layers. The hidden layer is responsible for processing the incoming information, just like the human perceptual nerves, the transmission of information often requires multiple layers of neurotransmission processing. By multiplying the signal obtained by each neuron in the previous layer by the corresponding weight, and summing, the summary information is obtained, and then the information is passed through the nonlinear activation function to obtain the output value.

When starting to train a neural network, all its weights (model parame-
ters) are first randomly initialized (Cao et al., 2018; Yu and Xu, 2014). The neural network is then continuously adjusted to produce output similar to the training data labels by continuously adjusting the parameters. This process of adjusting parameters requires the application of a loss function that measures the loss (gap) between the output value and the target value (Hoffman et al., 2018; Jadon, 2020). For example, in the text generation tasks, we use cross-entropy as a loss function to measure the distance between the predicted distribution (the generation output) and the actual distribution (the desired target). The smaller the loss value, the closer the predicted distribution is to the actual distribution. The loss function serves as the objective function in neural network optimization, and the process of training a neural network involves the minimization of this loss function.

How to minimize the loss function requires the backpropagation algorithm (Rumelhart et al., 1986b,a). To put it simply, it starts from the output layer and uses gradient descent to gradually update the parameters of each layer forward. This direction is opposite to the forward propagation direction, so it is called backpropagation. This method is based on the gradient descent method, and its central idea is to update the parameter values along the opposite direction of the gradient of the objective function to achieve the minimum objective function.

In practice, the neural networks are mostly optimized by the batch stochastic gradient descent algorithm (Robbins and Monro, 1951; Amari, 1967; Bottou et al., 1991). The principle of the stochastic gradient descent algorithm is as follows (Bottou, 1998; Bottou and Bousquet, 2008):

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \beta} \Delta l(x, w_t)$$

(2.1)

where $w_t$ represents the parameters of the model during the $t$th training, $n$ is the batch size, and $\eta$ is the learning rate. $\Delta l(x, w_t)$ is the gradient of the loss function to $w_t$, and $x$ is one of the sample in $\beta$. The batch size represents the number of samples selected by the model for a single training step, and the learning rate represents the step size, which determines the degree of adjustment of the model parameters during each iteration and affects the convergence of the loss function to the minimum value.
Building upon the foundational neural network concepts discussed above, our attention now turns to two types of specific models: the sequence-to-sequence (seq2seq) models and the graph-to-sequence (graph2seq) models. These models are employed in this dissertation, each tailored to distinct DRS format representations. Additionally, we introduce pre-trained neural models (PTM), a widely adopted technique in contemporary Natural Language Processing (NLP) tasks that generally enhances performance considerably.

2.2.1 Sequence-to-Sequence Models

As a basic neural network architecture, FFNN has many limitations when dealing with practical tasks. It treats each input (or feature) as independent of others, ignoring order or correlation in sequence data. Later, Recurrent Neural Networks (RNN; Elman 1990) and its variant Long Short Term Memory (LSTM; Hochreiter and Schmidhuber 1997) networks have been proposed one after another. RNN has a recurrent structure in which information is passed between time steps, making it suitable for processing sequence data, such as NLP tasks. However, RNN has difficulty in effectively capturing long-distance dependencies and suffers from the vanishing gradient problem (Basodi et al., 2020). LSTM uses a specially designed gating mechanism to control the flow of information, which alleviates the shortcomings of RNN in processing long sequences, allowing it to better remember and transfer information. LSTM can handle many NLP tasks well, such as text classification (Zhou et al., 2016; Liu and Guo, 2019), and sentiment analysis (Divate, 2021). To perform text-to-text tasks, such as machine translation or dialogue systems, LSTM is usually used as the core components of seq2seq models, also known as an encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014). Seq2seq models typically use an encoder to process an input sequence and encode it into a fixed-length context vector, and then use a decoder to generate an output sequence. Text often has sequences of varying lengths, and the seq2seq model can handle variable length input and output sequences without having to pre-specify a fixed sequence length (Vengopalan et al., 2015; McCann et al., 2017).
Figure 2.9: Schematic overview of the basic sequence-to-sequence architecture using LSTM. Although the encoder and decoder can contain multiple LSTM layers, only one is depicted here.

An example of the LSTM-based seq2seq architecture is shown in Figure 2.9. In this model, an LSTM layer (or multi-layer LSTM) acts as an encoder, processing the input sequence \( x = (x_1, x_2, ..., x_n) \) and returning its final internal state \((h, c)\) as the context vector. \( h \) and \( c \) represent the hidden state and the cell state respectively, where the hidden state \( h \) can be regarded as the current working memory and the cell state \( c \) acts as a long-term memory cell. Another LSTM layer (or multi-layer LSTM) acts as a decoder, using the intermediate state vectors \((h, c)\) from the encoder as the initial state. \(<BOS>\) and \(<EOS>\) are the special tokens added to the target sequence before it is fed into the decoder. \(<BOS>\) is the starting symbol of the target sequence, which provides a signal to the decoder to start the generation process. Using the hidden state at the current time step, the decoder can generate an output probability distribution that includes the probability of every possible token in the vocabulary. Typically, this is achieved through a fully connected layer and softmax activation function. To generate the actual output token, the decoder employs the beam search method, which involves considering multiple tokens with high probabilities and selecting the most promising one as the output at the current time step. The above steps will be repeated until a special end symbol (such as \(<EOS>\) in Figure 2.9) is generated or the maximum output sequence length is reached (Post and Vilar, 2018; Kang et al., 2022).
Attention Mechanism Since the final state of the above encoder is the only context information passed to the decoder, this means that the decoder must rely on this fixed context vector to generate the entire output sequence. In this case, the models may face challenges if the input sequence is very long or contains basic information distributed at different time steps. These challenges can lead to information loss and performance degradation, primarily due to the limited ability of the encoder to effectively capture such distributed information. Therefore, we discuss here a critical component: the attention mechanism (Bahdanau et al., 2015). The purpose of incorporating the attention mechanism is to transform the behavior of the decoder. Rather than encoding the entire input sequence into a fixed-length context vector, it enables the decoder to dynamically calculate a new context vector based on the currently generated token. This dynamic approach ensures that distinct context vectors are employed at each time step, effectively resolving the issue of information loss.
A schematic overview of the seq2seq architecture with the attention mechanism is shown in Figure 2.10. Unlike in the seq2seq model without attention, the decoder calculates a weighted sum at each time step based on the current context vector and the intermediate state of the encoder to decide which parts of the input sequence it should focus on. This weighted sum, known as attention weights, enables the decoder to dynamically adjust the attention to better capture relevant information in the input sequence. Therefore, the decoder can obtain a different context vector at each time step, which allows the model to better handle long and variable length sequences and to capture important parts of the input sequence more flexibly (Hao et al., 2019; Lin et al., 2022). Mathematically, the context vector is achieved through the following formula (Luong et al., 2015; Sennrich et al., 2017):

\[ c_t = \sum_{i=1}^{T} \alpha_{ti} h_i \]  

(2.2)

\[ \alpha_{ti} = \text{softmax}(e_{ti}) = \frac{\exp(e_{ti})}{\sum_{k=1}^{T} \exp(e_{tk})} \]  

(2.3)

\[ s_t = \tanh(W[s_{t-1}, y_{t-1}]) \]  

(2.4)

\[ e_{ti} = s_t^T W_a h_i \]  

(2.5)

where \( c_t \) represents the context vector at time step \( t \) in the decoder; \( \alpha_{ti} \) is attention weight, which is calculated by normalizing the scores \( e_{ti} \) through the softmax function; \( e_{ti} \) represents the attention scores and \( s_t \) represents the initial hidden state of decoder at time step \( t \); \( y_{t-1} \) represents the previously generated output (often referred to as "previous label/token") at time step \( t - 1 \) and \( \tanh \) is the activation function. Next, the attentional hidden state of the decoder can be calculated as follows (Luong et al., 2015):

\[ \tilde{s}_t = \tanh(W_c[s_t, c_t]) \]  

(2.6)

The attentional vector \( \tilde{s}_t \) is then fed through the softmax layer to produce the predictive distribution formulated as:

\[ o_t = \text{softmax}(V \tilde{s}_t) \]  

(2.7)
2.2. Neural Models

Among them, $V$, $W$, $W_a$ and $W_c$ are weight matrices to be learned during training. The above architecture is the general model that we will be using for our semantic parsing and text generation experiments described in Chapters 3, 4, 5, and 6. Detailed hyper-parameters setting of our experiments can be found in the respective chapters.

2.2.2 Graph-to-Sequence Models

Graph2seq models are a class of neural network models tailored for tasks where the input consists of a graph (such as the semantic graph shown earlier in Figure 2.6) and the desired output is a sequence (such as a natural language sentence). These models employ an encoder-decoder architecture, comprising a graph encoder and a sequence decoder (Yu et al., 2021; Guo et al., 2019). In contrast to traditional seq2seq models, which are primarily designed for sequential data, graph2seq models possess a unique capability. This unique capability arises from the graph encoder’s ability to convert the input graph into a structured graph representation, along with a set of node representations. The sequence decoder then effectively utilizes these representations to generate the target sequence. The primary difference between graph2seq and seq2seq models is in how they handle the input. For input data such as knowledge graphs (Wang et al., 2019; Ribeiro et al., 2020), molecules in chemistry (Reiser et al., 2022) and graph-structured meaning representation (Song et al., 2018; Beck et al., 2018), the conventional sequence encoder of seq2seq may struggle to adequately capture and encode the essential structural information in graph. To address this challenge, graph2seq models introduce a dedicated graph encoder, specifically designed to process and encode the structural information inherent in graph-based input data. A schematic overview of the basic architecture is shown in Figure 2.11. Consequently, in this section, our focus will be on clarifying the underlying principles of graph encoders.

A graph is represented as $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Let $v_i \in V$ be a node and $e_{ij} = (v_i, v_j) \in E$ be an edge pointing from $v_j$ to $v_i$. The neighborhood of a node $v$ is denoted as $N(v) = \{u \in V | (v, u) \in E\}$ (Wu et al., 2020). To effectively model graph-structured
data, the concept of Graph Neural Networks (GNNs; Kipf and Welling 2017; Hamilton et al. 2017) emerged. When dealing with graph data, GNNs fundamentally operate by iteratively aggregating information from neighboring nodes and integrating this aggregated information with the current central node representation during the propagation process. From a network architecture perspective, GNNs employ a stacking mechanism, combining multiple propagation layers. These layers are composed of aggregation and update operations. The formulation of the propagation process in GNNs is as follows (Ribeiro et al., 2020; Wu et al., 2020):

$$n^l_v = \text{Aggregator}_l(h^l_u, \forall u \in N_v)$$  \hspace{1cm} (2.8)

$$h^{l+1}_v = \text{Updater}_l(h^l_v, n^l_v)$$  \hspace{1cm} (2.9)

where $h^l_u$ denotes the representation of node $u$ at $l^{th}$ layer, while $\text{Aggregator}_l$ and $\text{Updater}_l$ correspond to the aggregation and update operations at the $l^{th}$ layer, respectively. During the aggregation step, previous studies have pursued various strategies, including the equal treatment of all neighbors through a meaning-pooling operation (Li et al., 2016; Hamilton et al., 2017) and the discrimination of neighbor importance using an attention mechanism (Veličković et al., 2018). In subsequent update steps, the representation of the central node and the aggregated neighborhood information will be integrated into the updated representation of the central node. After $L$ iterations, a node’s representation encodes the structure information within its
2.2. Neural Models

$L$ hop neighborhood. So far, various strategies have been proposed to enhance the integration of these two representations, including methods such as GRU (Cho et al., 2014), concatenation with nonlinear transformations, and summation. The choice of $\text{Aggregator}_l$ and $\text{Updater}_l$ varies depending on the specific GNN model in use (Kipf and Welling, 2017; Hamilton et al., 2017). In Chapters 6 and 7, we will detail the $\text{Aggregator}_l$ and $\text{Updater}_l$ operations for GNN frameworks that are used for our graph-structured DRS (variable-free) data.

2.2.3 Pre-trained Models

Recently, large-scale pre-trained seq2seq language models have proven to be very useful in many NLP tasks, including semantic parsing (van Noord et al., 2020) and text generation tasks (Bai et al., 2022; Bevilacqua et al., 2021). In this section, we will focus on the BART model (Lewis et al., 2020), which is one of the most popular pre-trained seq2seq language models and is also the model we will mainly cover in Chapter 8. Before that, we will review the development that led to pre-trained language models (PLMs) and briefly introduce the techniques used in other chapters of this dissertation.

In natural language processing tasks, text data usually needs to be converted into the form of numerical vectors, so that neural network models can process and understand it. The quality of this representation not only affects the performance of the underlying model, but also plays a pivotal role in the effectiveness and accuracy of the entire task. Early proposed methods use pre-trained static word vectors, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). While Word2Vec is a significant step, researchers continued to develop new methods for training embeddings from scratch. This approach involves training word embeddings as part of the model parameters during a specific NLP task. This allows embeddings to be tailored to specific tasks and domains, rather than relying on pre-trained embeddings. Training embeddings from scratch requires more resources but provides flexibility to cater to specific task requirements. Subsequently, an important approach in natural language processing, namely PLMs-based methods, gained attention. It focuses on learning context-aware
word embeddings, such as ELMo (Peters et al., 2018), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2019). These models have significantly advanced the field by capturing rich contextual information and achieving state-of-the-art results on a variety of NLP tasks. In the Chapters 3, 5, 6 and 7, we will focus on using embeddings trained from scratch. In Chapter 4, BERT embeddings will be used in the experiments, which are useful when the input to the encoder is natural language, such as parsing tasks (van Noord et al., 2020).

While BERT excels at capturing bidirectional context in text and has achieved remarkable success in various NLP tasks, BART takes a slightly different approach to generating text. BART combines bidirectional and auto-regressive functions to provide a powerful text generation model that can be used for various text generation and sequence generation tasks (Lai et al., 2021; Cui et al., 2021). Unlike BERT, which focuses on understanding and contextualizing language, BART specializes in generating coherent and contextually accurate text. Although the latest GPT models (Brown et al., 2020; OpenAI, 2023) have surpassed BART in terms of overall performance and functionality, BART still has advantages in some practical applications due to its relatively small model size. In Chapter 8, we focus on the BART model to improve performance on multilingual semantic parsing and text generation tasks.

**Fine-tuning** An important technique that goes hand in hand with pre-trained models is fine-tuning (Dai and Le, 2015; Howard and Ruder, 2018). Fine-tuning refers to further training and adjusting some or all model parameters based on the pre-trained model for specific tasks and data fields. Since the pre-trained model is usually trained on large-scale data, it has certain versatility and generalization capabilities. The goal of fine-tuning is to better adapt the pre-trained model to a specific task with limited training data on a relatively small data set, thereby improving the model’s performance on that task. Fine-tuning can be regarded as a bridge between pre-training and task-specific training, enabling the model to better adapt to new tasks and effectively saving computing resources and time. The classic fine-tuning approach involves continuing training of a pre-trained model
with a small amount of task-specific data. During this process, the weights of the pre-trained model are updated to better fit the task. The amount of fine-tuning required depends on the similarity between the pre-training corpus and the task-specific corpus. If the two are similar, only a small amount of fine-tuning may be needed. If the two are not similar, more fine-tuning may be needed. In Chapters 3 and 8, this fine-tuning technique is used to improve the performance of tasks.

2.2.4 Neural Semantic Parsing and Generation

In this section, we will describe previous semantic parsing and meaning-to-text generation methods based on neural network models. We will discuss approaches to semantic parsing and meaning-to-text generation in separate parts, with particular interest for DRS parsing and generation systems. While the existing research on DRS is relatively limited, there is valuable inspiration that can be drawn from semantic parsing and generation endeavors in other semantic formalisms, particularly the widely adopted AMR formalism. Compared with DRS meaning representations, AMR has rich and mature corpus resources and an extensive body of research. Therefore, it is essential to conduct a comprehensive review and categorization of the relevant literature in this domain.

Semantic Parsing Most of the previous efforts for semantic parsing can be broadly categorized into two groups: rule-based methods and neural network-based approaches. Rule-based methods typically involve applying a set of rules to the syntactic analysis of a sentence, aiming to generate a formal meaning representation. However, these rules are manually crafted, tailored to specific domains, and demand substantial domain expertise during their design (van Noord, 2021). A classic parser related to DRS is Boxer (Bos, 2008), which is based on rules and statistical methods. However, in this thesis, our focus will be on neural network-based methods because of their transformative impact on the field.

Indeed, neural models have emerged as the predominant approaches in this domain and typically yield the best performance. Moreover, there are
Various techniques have been incorporated into neural-based systems, including adding linguistic features (van Noord et al., 2019), using character-level neural networks (van Noord et al., 2018b), and incorporating the embeddings of PLM (van Noord et al., 2020). In Chapter 3 and Chapter 4, respectively, we employ the latter two technologies as our baseline models for English semantic parsing, acknowledging their efficacy in improving overall parsing performance. In addition to the seq2seq model mentioned earlier, there are two other research directions: tree-based methods (Liu et al., 2018, 2019a) and graph-based methods (Fancellu et al., 2019; Fu et al., 2020). These two methods convert box-format DRS into tree-based representation and graph-based representation respectively, and then use structure-aware decoders to replace the original sequential decoder to improve the performance. Most of the above studies have focused on English texts, and are now gradually extending their coverage to include other Latin languages. It is worth noting that Fancellu et al. (2019) is an initial attempt at multilingual DRS parsing. They employ a training approach that trains parsers for each language from scratch, combined with a more complex graph data transformation process. Due to the relatively intricate conversion process employed by both tree-based and graph-based methods, making reproduction challenging, we excluded these methods from subsequent comparative experiments. In Chapter 3 and Chapter 4, our parsing efforts shift focus to Chinese, a language that has not been explored in prior research. Chapter 8 further expands our parsing scope to cover English, German, Dutch, and Italian, which are consistent with the languages studied by Fancellu et al. (2019). Notably, our study integrates parsers for these four languages into a unified multilingual parsing model.

Regarding AMR parsing, a range of neural model techniques are employed. These techniques can be divided into three main categories: transition-based methods (Fernandez Astudillo et al., 2020; Zhou et al., 2021), sequence-to-graph methods (Zhang et al., 2019b; Cai and Lam, 2020a) and sequence-to-sequence methods (Xu et al., 2020; Bevilacqua et al., 2021). Neural transition-based parser operates by sequentially processing a sentence from left to right, incrementally constructing the graph. This construction involves alternately inserting new nodes or establishing new
edges. In this way the parsing process can be viewed as predicting a sequence of actions, the performance of which relies heavily on the effective modeling of the parser state at each decision step. Sequence-to-graph AMR parsers operate by jointly deciding a new node and its connections to existing nodes at each time step. This innovative approach enables the model to dynamically expand the graph structure in a coherent manner, capturing the relationships between nodes as it processes the input sequence. The sequence-to-sequence parsing linearizes the AMR graph and transforms the parsing task into a sequence-to-sequence transduction. A distinctive feature is its use of shared vocabulary, equal treatment of concepts and relational predictions (Cai and Lam, 2020a).

Most previous research on DRS parsing has predominantly relied on the seq2seq model. This preference stems from the fact that the available DRS corpus suitable for neural network models is primarily structured in a clause-format. Unlike AMR, the application of graph models to clause-format DRS is less convenient. The variable-free DRS, however, offers a promising avenue to address this limitation. While transition-based and graph-based parsing approaches are not employed in this thesis, exploring the related work in AMR parsing serves as a source of inspiration for future endeavors for DRS parsing. Lately, the performance of AMR parsing has seen substantial improvement, largely attributed to the successful implementation of pre-training techniques. For example, Zhang et al. (2019b) and Cai and Lam (2020a) use pre-trained language model BERT for sentence encoding, the same technique used in the DRS parsing (van Noord et al., 2020); Bevilacqua et al. (2021) finetune BART for AMR parsing, which extend a Transformer encoder-decoder model pretrained on English text denoising to also work with AMR.

**Meaning-to-text Generation** Much like DRS parsing, previous efforts in the generation task can be roughly divided into rule-based methods (Basile and Bos, 2011) and neural network-based methods (Liu et al., 2021a). However, in contrast to DRS parsing, the field of DRS-to-text generation has only recently garnered attention from NLP researchers (Basile and Bos, 2011; Narayan and Gardent, 2014; Basile, 2015). A recent study by Liu
et al. (2021a) introduced a novel approach where DRS is transformed into trees, with each box representing a subtree and conditions within the box corresponding to children of the subtree. Subsequently, they presented a DRS-to-text model based on tree-LSTMs. Despite achieving notable results, this method encounters challenges similar to those observed in tree-based DRS parsing methods. Specifically, the large number of rules required to be converted into the DRS tree format creates difficulties with reproducibility.

Due to the vast amount of annotated corpora for AMR, there has been a discernible imbalance in the research focus, with a greater emphasis on AMR-to-text compared to DRS-to-text. Approaches to AMR-to-text generation can be categorized into two primary classes: graph-to-sequence models (Beck et al., 2018; Damonte and Cohen, 2019) and sequence-to-sequence models (Konstas et al., 2017). Graph-to-sequence models employ a graph encoder to encode an AMR graph, with a sequence decoder for generating sentence. Such work aims to preserve structural information in AMR data via graph encoders (Song et al., 2020; Wang et al., 2020). Similar work is also applied to other graph structured data, such as Knowledge Graph (KG; Ribeiro et al. 2020; Song et al. 2020) and structured query language (SQL; Xu et al. 2018) query. In contrast, sequence-to-sequence models linearize an AMR graph into a sequence with bracket representation, treating it as a seq2seq task using either randomly initialized or pretrained models for encoding (Ribeiro et al., 2021b; Bevilacqua et al., 2021; Procopio et al., 2021). Moreover, AMR research has gradually expanded from English to non-English languages, including not only AMR parsing (Blloshmi et al., 2020) but also multilingual AMR-to-text generation tasks (Sobrevilla Cabezudo and Pardo, 2019; Ribeiro et al., 2021a). Such research, like that of DRS, is also based on the theory that meaning representation is language-neutral, although some preliminary studies showed that AMR has limitations as an interlingua due to distinctions in the underlying ontologies (Xue et al., 2014).

In Chapter 5, we employ an LSTM-based seq2seq model for the DRS-to-text generation task, a neural model similar to the one utilized in the study by Konstas et al. (2017). In Chapter 6, the focus shifts to an explicit comparison of the factors influencing the performance difference between sequence-to-sequence models and graph-to-sequence models in DRS-to-text
generation tasks. In Chapter 8, we adopt finetune BART technique for both DRS parsing and DRS-to-text generation, which is similar to the work of Bevilacqua et al. (2021) on AMR parsing. The distinction lies in our integration of multi-tasking (parsing and generation) and multi-language support (four languages) within a unified model. The closest to our work, Bai et al. (2022) propose a monolingual framework based on AMR, where the pre-training and fine-tuning share the same data format to facilitate knowledge transfer between them.
PART II
Neural Semantic Parsing