TU WIEN Informatics

Angleichen von Sätzen an ihre formale Bedeutungsdarstellung im Kontext von Discourse Representation Structures

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Christian Obereder, BSc.
Matrikelnummer 11704936

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Univ. Ass. Gábor Recski, PhD

Wien, 2. Mai 2024

Christian Obereder	Gábor Recski

TU WIEN Informatics

Aligning sentences to their formal meaning representation in the context of Discourse Representation Structure Parsing

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Christian Obereder, BSc.
Registration Number 11704936

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Ass. Gábor Recski, PhD

Vienna, May 2, 2024

Christian Obereder	Gábor Recski

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Christian Obereder, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. Mai 2024

Christian Obereder

Acknowledgements

First and foremost I want to thank my supervisor, Dr. Gábor Recski, for supporting me throughout this thesis and for always taking the time to give me feedback, discuss new ideas and share his expertise with me.

I am also grateful to my partner Yana for helping me write this work by providing their perspectives on various approaches, proof-reading this thesis and listening to the problems I encountered.

My gratitude also goes to my friends, who keep my life fun and balanced. Thank you for always trying to stay calm during boardgames, to be on time, to stay awake during meetups, and to beat me at ping-pong. It inspires me to never give up.

Lastly, I want to thank my parents for their support and encouragement during my studies and while writing this work.

Kurzfassung

Discourse Representation Structures (DRS) (Kamp et al., 2011) sind eine Möglichkeit, einen Satz in natürlicher Sprache als eine formelle Bedeutungsrepräsentation darzustellen. Für maschinelles Lernen können DRS als eine eindimensionale Liste an Klauseln dargestellt werden. In den vergangenen Jahren wurden Systeme zur automatisierten Erstellung von DRS, aus Sätzen in natürlicher Sprache, basierend auf neuronalen Netzen entwickelt. Diese Systeme werden auf Daten aus der Parallel Meaning Bank (Abzianidze et al., 2017) trainiert und erzielen hohe Performanz. Die Daten aus der Parallel Meaning Bank bestehen jedoch nicht nur aus Sätzen und ihren zugehörigen DRS, sondern beinhalten auch eine Zuweisung zwischen den beiden. Für jede Klausel im DRS gibt es ein Wort im Satz, welches für diese Klausel am relevantesten ist. Moderne neuronale Systeme, die automatisiert DRS generieren, ignorieren jedoch diese Zuweisung, welche in den Trainingsdaten vorhanden ist, und produzieren ausschließlich DRS. Eine solche Zuweisung wäre jedoch sehr nützlich, da sie es ermöglicht, einzelnen Wörter die Information, welche in DRS-Klauseln vorhanden ist, zuzuweisen. Diese Arbeit beschäftigt sich damit, ein bereits existieren neuronales sequence-to-sequence System zur Erstellung von DRS (van Noord et al., 2020) zu erweitern, sodass es auch die besprochene Zuweisung generiert, ohne die Architektur des bestehenden Modells stark zu verändern. Zu diesem Zwecke wird mit einem Ansatz experimentiert, welcher die beschriebene Zuweisung aus dem Aufmerksamkeitsmechanismus eines sequence-to-sequence Systems ausliest, sowie einem Ansatz, welcher die Zuweisung als Teil der zu generierenden Sequenz in einem solchen System betrachtet. Außerdem wird eine Kombination der beiden genannten Systeme, welche die beste Performanz unter den beschriebenen Ansätzen zeigt, entwickelt. Schlussendlich werden noch DRS mit fehlerhafter Zuweisung, welche von genannten Systemen produziert wurden, manuell inspiziert, um Einblicke darin zu erlangen, welche Arten von Fehlern produziert werden und wie die beschriebenen Systeme noch verbessert werden können.

Abstract

Discourse Representation Structures (DRS) (Kamp et al., 2011) are a way of formally representing the meaning of a sentence. DRS Parsing is the task of automatically generating DRS from a given sentence, which is often done using machine learning techniques. Current state-of-the-art approaches employ sequence-to-sequence models, where the input sequence is the natural language sentence and the output sequence is DRS. For that purpose, DRS can be represented in a machine-readable way, as a flat list of clauses. In recent years, neural methods for parsing DRS using data from the Parallel Meaning Bank (Abzianidze et al., 2017) have shown promising performance. However, the data in this corpus consists not only of sentences and their corresponding DRS, but also an alignment between the two, describing which tokens of the input sentence are most relevant for a given clause in the DRS. State-of-the-art neural DRS parsers do not include this alignment in their output, instead only producing pure DRS. However, using DRS in downstream NLP applications such as Named Entity Recognition (NER), Relation Extraction (RE), or Open Information Extraction (OIE) requires that DRS clauses produced by a parser be aligned with words of the input sentence.
This work expands an existing neural sequence-to-sequence DRS parser (van Noord et al., 2020) so that it is capable of producing alignment alongside DRS, while making minimal changes to the underlying architecture. For the purpose of producing this alignment, an approach based on the attention-scores generated by the cross-attention-mechanism in an Encoder-Decoder model and an End-to-End approach are considered, with a combination of these approaches ultimately achieving the best overall performance in terms of alignment accuracy. Furthermore, a qualitative analysis of alignment errors produced by these approaches is provided, giving insights into the nature of such errors.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

Formal meaning representations provide the possibility of encoding the semantics of a sentence in a formal, machine-readable way. Semantic Parsing is the task of generating such formal meaning representations from natural language input. Discourse Representation Structures (DRS) (Kamp et al., 2011) is one such meaning representation that is grounded in formal logic and can express a large number of semantic phenomena. DRS parsing is the task of mapping sentences to DRS. In recent years, large annotated corpora, containing sentences and corresponding DRS, have become available, such as the Parallel Meaning Bank (PMB) (Abzianidze et al., 2017) and the Groningen Meaning Bank (Bos et al., 2017). A sample DRS from the PMB for the short sentence *The eagle is white* is shown in Figure 1.1. It first displays the DRS in clause-format alongside an alignment of each clause to the sentence, and then the DRS in box-format, which is used for its better human-readability. This means DRS can be represented as a graph, like in the box-format, or as a string or flat list, like in the clause-format. The example shows discourse-variables *x1, t1 and s1* that represent concepts from the sentence and predicate-functions that are used to model the available information. *x1* for instance represents the *eagle* from the corresponding sentence, or *t1* represents a point in time in the present, as the sentence is in present tense.
The large corpora containing DRS gave rise to neural DRS parsers (Fancellu et al., 2019; Liu et al., 2018; van Noord et al., 2018b), which generally manage to outperform previous DRS parsers that are based on symbolical, statistical, rule-based or mixed approaches (Johnson and Klein, 1986; Wada and Asher, 1986; Bos, 2015, 2008; Le and Zuidema, 2012a). Among such neural DRS parsers, both End-to-End approaches, which treat DRS parsing as a string-transformation task between a sentence-string and a DRS-string (clause-format), and approaches that have an explicit model of the structure of DRS can be found.

The DRS in the PMB contain an alignment between DRS and natural language input, i.e, each clause of the DRS is assigned usually one but potentially any number of tokens from the input sequence that can be considered most relevant for the given clause out of all the tokens in the input sequence. While some current neural DRS parsers do make use of this alignment for the task of generating DRS, they do not explicitly produce this alignment in their output, instead producing only the DRS itself. However, such an alignment is necessary to use the information contained in a DRS on a word- or phrase-level. Indeed, if one aims to make use of the information provided by DRS in other NLP tasks, e.g., Relation Extraction or Named Entity Recognition, being able to align specific clauses to specific parts of the sentence would be crucial. This work aims to explore ways to produce this alignment with a high accuracy and adapts an existing DRS parser accordingly.

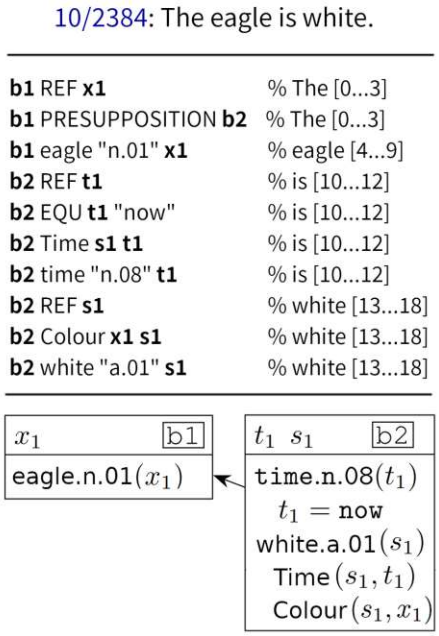


Figure 1.1: DRS for the sentence *The eagle is white.*, first in clause- and then in box-format from the 3.0.0 release of the PMB. The clause-format also shows the alignment of each clause to the input sequence. The box-format shows a box *b1* with a discourse-variable *x1* representing the *eagle* from in input sentence. Box *b2* contains discourse-variables *t1*, representing a point in time in the present, and *s1*, representing the adjective *white*. *b2* also sets the introduced discourse-variables in relation to each other.

As such, this thesis considers the following research questions (RQs):

1. **How can DRS that include an alignment to the input sequence be generated, and what is the impact of the choice of alignment generation method on the alignment accuracy? How does it affect performance**

in terms of F1-Score on the pure DRS generation task? The aim of this question is to develop methods for generating the alignment and explore the performance of different alignment generation methods in terms of alignment accuracy, meaning the percentage of correctly predicted alignments on matched DRS clauses. Furthermore, it seeks to investigate how the additional task of generating the alignment affects performance on the underlying DRS parsing task in terms of F1-Score.

2. **What are the characteristics of alignment errors? Can any shortcomings in the proposed systems be identified based on the errors they produce?** The aim of this question is to provide a qualitative error analysis of incorrect alignments that may reveal what kind of phenomena the proposed methods are struggling with during alignment and what approaches could be taken to address them.

This work makes the following contributions:
1) it extends an existing End-to-End neural DRS parser employing an Encoder-Decoder architecture with cross-attention (van Noord et al., 2020) so that it also produces the desired alignment. To that end, the following approaches are used, and their performance is compared: **1a)** For each generated DRS clause, attention-scores of the cross-attention-mechanism are taken into account and the token of the input-sequence with the largest attention-score is generated as part of the output alongside the DRS clause as its alignment. **1b)** For this approach, the target sequences in the training data consists not only of pure DRS but also include the alignment. This means a model is trained in an End-to-End fashion to generate alignment as part of the target sequence, whereas 1a) is trained in an End-to-End fashion to only generate pure DRS and the alignment is extracted from the attention-mechanism. **1c)** Approaches 1a) and 1b) are combined, with alignment being generated as in 1a) and the model being trained as in 1b). This proposed systems can predict alignment on top of DRS with very high accuracy, with 1c) achieving an accuracy of over 98%.
2) A qualitative analysis of alignment errors is performed by manually inspecting erroneous alignments and assigning them to a number of error-categories. This is done for 1b) and 1c) separately, giving insight into the overall alignment errors and the differences in alignment errors these systems produce.
For 1a), we also experiment with different scoring-functions for the cross-attention-mechanism, and find that alignment produced by a scoring-function that uses learnable weights (e.g. general / bilinear attention) produces better alignment than when using a scoring-function based on a static calculation (e.g. dot-product attention). All software used in our experiments is released under an MIT license and is available on GitHub[1].

[1] https://github.com/GitianOberhuber/Neural_DRS_alignment

1.2 Thesis Organization

This work is organized as follows:
Chapter 1 provides and introduction to the topic and explains which problem we attempt to solve and states our research questions. It also briefly summarizes the contribution this work provides.
Chapter 2 explains the necessary background, including an explanation of DRS, the alignment, and the corpora containing them, as well as an overview of existing DRS parsers and a detailed explanation of the Encoder-Decoder DRS parser that this work is based on and the underlying principles of such a model. Lastly, DRS pre- and post-processing is also discussed.
Chapter 3 explains the methods for generating alignment on top of DRS that this work proposes. They include an Alignment through Attention approach, where a pure DRS parser is trained, and the alignment is extracted from the cross-attention mechanism, an End-to-End approach where a sequence-to-sequence (seq2seq) model is trained to produce alignment alongside DRS as the output sequence, and a third approach that combines the previous two.
Chapter 4 provides design- and technical details on the performed experiments and their implementation.
Chapter 5 explains the metrics used for evaluating DRS and its alignments, and describes the challenges of evaluating performance on such a task that is comprised of two goals (DRS parsing and alignment generation). It also includes the results of our experiments and a discussion of them.
Chapter 6 describes the strategy used in a manual inspection of alignment errors produced by the proposed systems. The results of this manual inspection are then shown and discussed.
Chapter 7 describes the contributions of this work and summarizes our response to the research questions.

CHAPTER 2

Background

DRS is a formalism that represents a scoped meaning representation which can model a number of semantic phenomena such as negation, modals, quantification, and presupposition. It is part of Discourse Representation Theory (DRT) (Kamp et al., 2011), which was introduced in the early 90s and has since then received a lot of attention (Pereira and Shieber, 1987; Asher and Lascarides, 2003; Muskens, 1996; Bjerva et al., 2014). DRS are complex, recursive structures that represent meaning using variables and predicate functions, and they can be translated into formal logic (Blackburn and Bos, 2005). For the purpose of displaying a DRS, the more human-readable box-format and the more machine-readable flat clause format are often used. DRS parsing is the task of automatically generating DRS from natural language input. In the past, DRS parsers employed mainly symbolical and statistical methods (Le and Zuidema, 2012b; Wada and Asher, 1986; Johnson and Klein, 1986). Most noteworthy among such non-neural parsers is Boxer (Bos, 2008, 2015), which used a combination of rule-based and statistical models.

In recent years, the Parallel Meaning Bank (PMB) (Abzianidze et al., 2017) and Groningen Meaning Bank (Bos et al., 2017), which are annotated corpora containing English sentences and their corresponding DRS, have been released. The DRS contained in the PMB were initially generated by Boxer and then manually corrected by expert linguists, elevating them to gold-quality. The alignment this work is using for training and evaluation was therefore also generated using Boxer.

The availability of such corpora enabled the development of neural DRS parsers and facilitated a Shared Task on Discourse Representation Structure Parsing in 2019 (Abzianidze et al., 2019; Fancellu et al., 2019; Liu et al., 2018). For this shared task, the goal was pure DRS parsing, so none of the participating parsers explicitly produce an alignment. One of the participating neural approaches is the parser devised by Noord et al., who use an Encoder-Decoder architecture (Sutskever et al., 2014; Cho et al., 2014b) with cross-attention (Bahdanau et al., 2014; Luong et al., 2015). They experiment with

different features and input representations such as word- and character-embeddings, additional linguistic information such as part-of-speech (POS) tags and lemmas and a varying number of encoders (van Noord et al., 2018b, 2019). Their model that uses a combination of BERT-embeddings (Devlin et al., 2018) with semantic tags and learned character-embeddings in two encoders currently achieves the highest performance among the neural parsers (van Noord et al., 2020). One of the models proposed in their work, which uses only BERT in a single encoder and neither semantic tags nor character-embeddings, is used as a base for this work, meaning it is the pure DRS parser that is being expanded to also produce alignment. While this BERT-only model was chosen for the sake of simplicity, there is no apparent reason why the alignment generation methods proposed in this work should not be applicable to any of their other models.

2.1 Discourse Representation Structures

This section gives an overview of DRS, first providing a step-by-step explanation of an example and then giving a more concise, formal definition.

An example DRS (without alignment) along with the corresponding natural language sentence, from van Noord et al. (2018b), is displayed in Figure 2.1. It shows a DRS corresponding to the sentence *He played piano and she sang* first in the flat clause-format (without alignment) and then in box-format. Both formats contain the same amount of information, however, the box-format is more easily readable. As such, the following explanation of this DRS will always refer to the DRS in box-format unless stated otherwise. Figure 2.1 shows a number of boxes or DRS which are identified by the box-variable (e.g. *b0*) in the top right of the box. The top-left of each box lists the discourse-variables which are used in that box (e.g. *x1* or *t1*). These variables have arbitrary names, but there is a distinction between box-variables and discourse-variables. The main body of a DRS contains either conditions or nested DRSs with a discourse relation over these DRSs. In the example, the latter is true for *b0*, while the former is the case for all other shown DRSs.
Consider first the five DRS that are indicated by an arrow: *b2, b3, b4, b6* and *b7*. These DRS are presuppositional, elementary DRS that contain basic conditions. They are called elementary DRS because they contain only discourse-variables and conditions, and their conditions are called basic because they consist of a predicate function over a discourse-variable. The arrow pointing towards these boxes indicates that they are presuppositional DRS. For example, box *b2* contains a discourse-variable *x1* and a predicate function *male.n.02()* that is parameterized by *x1*. This box therefore expresses that *x1* is male. Specifically, this predicate function is grounded in WordNet (Miller, 1995), a large English lexical database, and it represents the second (.02) definition of *male* as a noun (.n) found on WordNet, which reads *male, male person (a person who belongs to the sex that cannot have babies)*. Similarly, the other boxes *b3, b4, b6* and *b7* contain WordNet senses for piano, time, and female. Apart from theses basic conditions, *b*4 and *b*7 also contain a complex condition, representing how their discourse variables

00/3008: He played the piano and she sang.

```
b0 DRS b1                b0 DRS b5
b2 REF x1                b6 REF x3
b2 male "n.02" x1        b6 female "n.02" x3
b1 REF e1                b5 REF e2
b1 play "v.03" e1        b5 sing "v.01" e2
b1 Agent e1 x1           b5 Agent e2 x3
b1 Theme e1 x2           b5 Time e2 t2
b3 REF x2                b7 REF t2
b3 piano "n.01" x2       b7 TPR t2 "now"
b4 REF t1                b7 time "n.08" t2
b4 time "n.08" t1        b0 CONTINUATION b1 b5
b4 TPR t1 "now"          b1 Time e1 t1
```

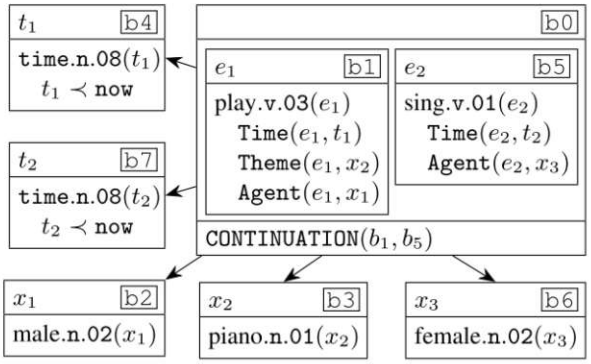


Figure 2.1: DRS in box- and clause-format for the sentence *He played piano and she sang* from the 2.2.0 release of the PMB, taken from van Noord et al. (2018b). Boxes $b2, b3, b4, b6$ and $b7$ are presuppositional (indicated by the arrow), elementary DRS with basic conditions, though $b4$ and $b7$ also contain a complex condition. $b0$ is the main DRS and also a segmented DRS, containing two more elementary DRS $b1$ and $b5$, as well as the discourse relation *CONTINUATION*, over $b1$ and $b5$. 00/3008 is the ID of the sample in the PMB.

$t1$ and $t2$, which are described by the eighth WordNet sense for the noun *time*, are less than a constant *now*. The conditions are called complex because they consist of logical operators on discourse-variables. Regardless of whether a condition is basic or complex, a DRS containing only discourse-variables and conditions on these discourse-variables is called a basic DRS. With this being established, it can be seen which of the mentioned boxes correlate to what part of the input sequence *He played the piano and she sang*: Informally speaking, $b2$ represents *He*, $b3$ represents *the piano*, $b6$ represents *she*, and $b4$ and $b7$ represent some point in time past (we have two boxes here because the sentence does not necessitate that things are happening at the same point in time in the past). Though not as straight forward as for the previous boxes, $b4$ and $b7$ can be thought of as correlating to the verbs *played* and *sang*, as they indicate the tense of the sentence.
Box $b0$ is the main DRS, indicated by the arrows going out from it, and also a segmented

DRS. It is called a segmented DRS because it contains nested boxes as well as a discourse relation (*CONTINUATION(b1, b5)*) over these boxes. A discourse relation takes as arguments box-variables and describes how they are linked, e.g., that *b5* is a continuation of *b1*. It contains two further boxes *b*1 and *b*5, which are again elementary DRS with basic conditions. However, these conditions include not only word senses from WordNet, but also thematic roles from VerbNet (Bonial et al., 2011). While similar, these two concepts distinguish themselves in that WordNet senses take exactly one argument and assign a word sense to a variable, while VerbNet roles always take two arguments and assign a role over the variables, For example *Agent(e1, x1)* in box *b1* expresses that *x1* is the agent performing *e1*. Apart from that, *b1* and *b5* are again elementary DRS, which are nested within *b0*. With this, it can again be informally said that *b*1 corresponds to *played* and *b*5 corresponds to *sang*. Lastly, the *and* in the sentence can be said to be represented by the discourse relation *CONTINUATION(b1, b5)*. This association between boxes and words of the input sentence is reflected in the alignment that this thesis is concerned with, as can be seen in Figure 2.2 in Section 2.2, which provides more detailed information on the alignment.
Throughout this work, any example DRS-clauses that are used to explain or illustrate something will be taken from the sample DRS in Figure 2.1 if possible.

The version of DRS that is being used in the PMB (and subsequently in this work) uses a number of extensions to the original DRS proposed in DRT (Kamp et al., 2011): presuppositions are modeled according to (Van der Sandt, 1992) and Projective DRT (Venhuizen et al., 2018) and non-logical symbols are resolved using WordNet (Miller, 1995) and VerbNet (Bonial et al., 2011). DRS are recursive structures that are usually displayed in the box-format, a connected graph of boxes (see Figure 2.1), where each box corresponds to a DRS. Due to the popularity of this representation, a DRS is also often referred to as a box and vice versa. The following is a more concise and formal definition of DRS: A DRS contains a main box (or DRS), and potentially a number of other DRSs that are considered presuppositional, all of which are labeled with a box variable. DRSs use discourse variables to represent the objects under discussion, as well as predicate functions, logical operators and discourse relations to describe these variables and how they relate to each other. Predicate functions take as arguments either one or two discourse variables or constants, and are used to model the information available on discourse variables. Logical operators, sometimes also called DRS operators, can take both discourse variables and box variables as arguments and are used to model concepts such as equality, negation, or comparison. Discourse relations take box-variables as arguments, thus being able to describe a relation between DRSs.
A DRS can either be an elementary DRS or a segmented DRS. Elementary DRS contain a set of discourse variables and a set of conditions, where a condition can be either a basic condition or a complex condition. A basic condition is a predicate function over discourse variables or constants, while a complex condition is a logical operator over discourse- or box-variables. A segmented DRS contains a number of nested boxes and a discourse relation over those boxes. A more precise, recursive definition of these concepts

is given in van Noord et al. (2018a).
Following this, given an example with two discourse-variables *x1* and *x2* and box-variables *b1*, *b2* and *b3*, all of which are arbitrarily named, a clause in a DRS will take one of four forms:

- ***b1 <Thematic Role> x1 x2*** ; a predicate-function is used to express a relation between $x1$ and $x2$ inside a box *b1*, for example: *b1 Theme x1 x2* would express that *x2* is the theme of *x1*, or *b1 Agent x1 x2* would express that x2 is the agent of x1 (e.g. as in *x2* is performing *x1*).
- ***b1 <word>.<part-of-speech>.<sense> x1*** ; a predicate-function to express an attribute of *x1* inside a box *b1* using word senses from WordNet (an English, lexical database), for example: *b1 male "n.02" x1* expresses that *x1* is male as per the second noun (n.02) definition found in WordNet.
- ***b1 <Logical/DRS-operator> x1 (x2)*** a logical- or DRS-operator over one or two discourse variables inside a box *b1*, for example *b1 EQU x1 x2* would express that discourse-variables *x1* and *x2* are equal, or *b1 REF x1* would be used to introduce a new discourse-variable, as all discourse-variables in a DRS need to be explicitly introduced.
- ***b1 <Logical/DRS-operator> b2*** a logical- or DRS-operator over a box variable *b2* in a box *b1*, for example *b1 NOT b2* would express that *b2* (and its contents) is negated.
- ***b1 <Discourse Relation> b2 b3*** ; a Discourse Relation over two boxes *b2* and *b3* inside a box *b1*, for example *b0 CONTINUATION(b2, b3)* would express that *b2* and *b3* are nested within *b1* and that *b3* (and its content) continues *b2* (and its content).

2.2 Parallel Meaning Bank & Alignment

2.2.1 Parallel Meaning Bank

The Parallel Meaning Bank (PMB) (Abzianidze et al., 2017)[1] is a multilingual corpus containing sentences in English, German, Italian, and Dutch alongside extensive semantic annotations, such as segmentation, syntactic parsing with Combinatory Categorcial Grammar, semantic tagging, and most importantly for the purposes of this thesis, DRS. Each of these is considered to be one layer of annotation. The English sentence is then translated and aligned to the other languages of the corpus. The semantic information available for the English sentence is then mapped to the translated languages based on this token-level alignment, and the assumption is made that such a mapping is meaning-preserving.

[1] https://pmb.let.rug.nl/

However, only the raw English sentence and the corresponding DRS are used in this work. It is also important to note that the PMB contains one possible DRS for the corresponding input sequence, not the only possible DRS for such an input sequence. These DRS are generated by the DRS parser Boxer (Bos, 2008, 2015), a system that uses a statistical approach and Combinatory Categorical Grammar (CCG) (Steedman, 2001). These automatically generated annotations (as well as those of the other annotation-layers in the PMB) can then be manually corrected by expert linguists on a wiki-like platform (Basile et al., 2012). Based on the amount of manual annotations available, data in the PMB is categorized as one of three disjoint groups: bronze-standard for entirely unannotated data (which corresponds to the output of the underlying system for automatic generation), silver-standard which is partially corrected (there exists least one manual correction) and gold-standard which is fully manually corrected. van Noord et al. (2018b) find that use of silver-standard data can prove beneficial, as their DRS parser that is trained on silver + gold-data and fine-tuned on gold-data outperforms a model trained only on gold-data.
The PMB has been growing steadily since its inception, with multiple stable versions having been released thus far. The First Shared Task on Discourse Representation Structure Parsing (Abzianidze et al., 2019) used the 2.2.0 release of the PMB, with some recent works also operating on the 3.0.0 release (van Noord et al., 2020) and 4.0.0 release (Poelman et al., 2022). This thesis uses the data provided in the 3.0.0 release. Table 2.1 shows the number of samples for each annotation layer, as well as the average number of clauses in a DRS per annotation-layer.

Category	**# samples**	**average # clauses**
Gold train	4.597	15.09
Gold dev	682	
Gold test	650	
Silver	67.965	25.76
Bronze	120.665	20.20

Table 2.1: Number of samples per annotation-level for the 3.0.0 release of the Parallel Meaning Bank (PMB). Gold means the output of the underlying DRS parser that serves as a basis for the DRS in the PMB has been fully manually corrected, silver means partially corrected, bronze means no manual corrections at all. Also shows the average number of clauses in a DRS per annotation-level.

2.2.2 Alignment

The DRSs found in the PMB also contain the alignment to the natural language sentence this thesis is interested in. For each clause in the DRS in clause-format, there can be a number of tokens of the input sequence which that clause is aligned to. Usually one clause is aligned to exactly one token, however, in rare cases one clause is aligned to multiple input tokens or no input token. As such, it constitutes an n:n relation between input

```
%%% He played the piano and she sang .
b2 REF x1               % He [0...2]
b2 male "n.02" x1       % He [0...2]
b1 REF e1               % played [3...9]
b1 REF t1               % played [3...9]
b1 Agent e1 x1          % played [3...9]
b1 TPR t1 "now"         % played [3...9]
b1 Theme e1 x2          % played [3...9]
b1 Time e1 t1           % played [3...9]
b1 play "v.03" e1       % played [3...9]
b1 time "n.08" t1       % played [3...9]
b3 REF x2               % the [10...13]
b3 piano "n.01" x2      % piano [14...19]
b6 DRS b1               %
b6 DRS b4               %
b6 CONTINUATION b1 b4   % and [20...23]
b5 REF x3               % she [24...27]
b5 female "n.02" x3     % she [24...27]
b4 REF e2               % sang [28...32]
b4 REF t2               % sang [28...32]
b4 Agent e2 x3          % sang [28...32]
b4 TPR t2 "now"         % sang [28...32]
b4 Time e2 t2           % sang [28...32]
b4 sing "v.01" e2       % sang [28...32]
b4 time "n.08" t2       % sang [28...32]
% . [32...33]
```

Figure 2.2: DRS with alignment in clause-format sentence *He played piano and she sang* from the 2.2.0 release of the PMB. The alignment, which comes after the %-character when present, consists of a token from the input sequence followed its start- and end-index.

tokens and DRS clauses. The alignment (if present) is always given as a 'comment' beside the DRS-clause, meaning the DRS-clause and alignment are separated by a %-character. Structurally, the alignment consists of the exact token from the input sequence, followed by the start- and end-index of that token within the input sequence. An example of a DRS with alignment from the PMB release 2.2.0 is shown in Figure 2.2. The very first clause, *b2 REF x1* has an alignment *He [0...2]* with the alignment token *He* and its start- and end-indices within the input sequence 0 and 2. An intuitive explanation of the alignment for this sample, describing why certain clauses can be thought of as belonging to certain tokens from the input sequence, is given in Section 2.1. Table 2.2 shows statistics on the alignment, displaying the number of clauses that have no alignment, clauses that have exactly one aligned token, and clauses that have more than one aligned tokens, for the gold- and silver-data of the PMB 3.0.0 release. It can be seen that the vast majority of clauses have exactly one aligned input token, while about 3% of clauses have multiple alignments and less than 0.5% of clauses have no alignment for the gold-data, though this number increases to 1.86% on the partially manually corrected silver-data. Sentences in the PMB are explicitly tokenized. In cases where multiple tokens describe a single concept, a '~' is used, e.g. *New York* or *10 a.m* would be *New~York* and *10~a.m.* Such tokens can also be found in the alignment, where *New~York* would be a single

alignment with a size of eight. About 3% of all alignments consist of multiple elements connected by '~' (3.3% for the gold-data and 3.47% for the silver-data).

	No alignment		One alignment		Multiple alignment	
	Count	Percentage (%)	Count	Percentage (%)	Count	Percentage (%)
Train	311	0.31	98 263	96.51	3246	3.19
Dev	34	0.27	12 280	96.71	384	3.02
Test	23	0.17	13 075	96.48	454	3.35
Silver	44 020	1.86	2 236 946	94.69	82 000	3.47

Table 2.2: Counts and fraction of DRS-clauses that have no-, exactly one-, or multiple aligned tokens from the input sequence in the PMB 3.0.0 release. Figures are listed for the train-, dev- and test-sets of the gold-data (i.e. fully manually corrected data), as well as for the silver data (i.e. partially manually corrected data), for which there is no dev- or test-set.

2.3 DRS parsing

DRS parsing is the task of automatically generating DRS from natural language input. To that end, a number of systems that use different approaches have been proposed. Boxer (Bos, 2008, 2015), is a statistical and rule-based system. The output that Boxer produced was used as a basis for the creation of annotated corpora (Abzianidze et al., 2017; Bos et al., 2017), which recent neural DRS parsers make use of. Poelmann et al. use an approach based on Universal Dependency graphs and apply a number of transformations based on rules learned from the PMB-data (Poelman et al., 2022). Their approach generates the UD graph using an existing UD-parser and applies a number of graph transformations. A DRS is then created by connecting boxes and labeling nodes and edges in the transformed UD graphs. Evang constructs word-meaning pairs using the training data and employs them in a transition-based parser. They use stacked LSTMs to encode parsing states and a statistical model to make transition decisions (Evang, 2019). Liu et al. represent DRS as a tree and make use of a structure aware approach that first generates the overall structure of the DRS and then fills in conditions and variables in later steps using a number of encoder-decoder components (Liu et al., 2018, 2019b). They experiment with both Transformer- and biLSTM-setups. Furthermore, their work looks at the task of predicting DRS on a document-level, as opposed to sentence-level like most other DRS parsers. They also improve their work by adding a supervised attention mechanism which learns an alignment between sentences and tree-nodes, using it as a feature employing it in a copying-mechanism which can copy parts of the input based on this alignment (Liu et al., 2019a). Noord et al. use the clause-format of DRS as it is found in the PMB and train a sequence-to-sequence model with attention using character-level input representation. They also apply a number of pre-processing-steps such as rewriting the arbitrary variable names of DRS using a

relative naming scheme and post-processing which fixes certain potential issues of the generated DRS, as their End-to-End model has no way to ensure well-formed output (van Noord et al., 2018b). They improve this approach, using largely the same model as before, by employing BERT-embeddings (Devlin et al., 2018) for their input representations and adding further information such as character-embeddings, part-of-speech tags (POS) and lemmas in one or multiple encoders (van Noord et al., 2020). Their best performing model uses BERT-embeddings, learned character-embeddings, as well as semantic tags in two encoders. To the best of our knowledge, this is the currently best performing DRS parser among those mentioned (and among those who evaluate their approach in clause-format on the PBM-data, thus facilitating comparison to Noord et al.), though it manages to outperform the model of Liu et al. by only about 1.5 points in terms of F-Score.

2.4 Encoder-Decoder Architectures

In machine learning, input- and output-representations are usually expected to be vectors of fixed length. Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986; Werbos, 1990) enable the usage of inputs of variable length, as they can be unrolled for as many timesteps as there are elements in the input sequence. However, input and output still need to have the same dimensions even when employing an RNN, which is problematic, as for many tasks such as translation from one language to another, both input and output should ideally be sequences of arbitrary length. Encoder-Decoder architectures (Cho et al., 2014b; Sutskever et al., 2014) are a family of models that make use of RNNs such as the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to solve the task of parsing sequences whose lengths are not known a priori. This is achieved by having one RNN (the encoder) process the input sequence, one step at a time, and output a large, fixed-size vector (often called the "Context Vector"), which is then passed to the decoder. The decoder (which is another, independent RNN) then generates the output sequence, taking the context-vector provided by the encoder into account. An illustration of the described Encoder-Decoder architecture from Cho et al. (2014b) can be seen in Figure 2.3. The following is a detailed description of their proposed *RNN Encoder-Decoder*.

Formally, an RNN has a hidden state h and an output o. Given a sequence $x = \{x_1, ..., x_T\}$, the RNN processes this sequence one step t at a time, using the current input x_t as well its hidden state at that timestep h_t to compute the new hidden state using the formula

$$h_t = f(x_{t-1}, h_{t-1}), \tag{2.1}$$

with f being a non-linear function depended on what RNN is being used, (Sutskever et al., 2014) for example use LSTM. The output of an RNN that is trained to predict the next element in a sequence at each timestep is given as

$$p(x_t|\{x_1, ...x_{t-1}\}), \tag{2.2}$$

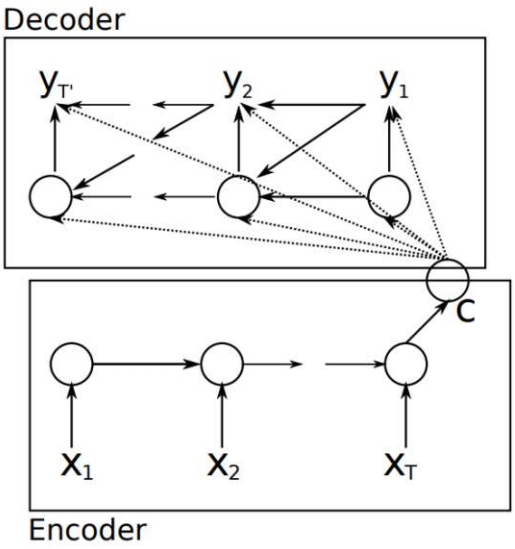


Figure 2.3: Schematic overview of encoder-decoder architecture based on RNNs. The encoder-block reads the input sequence and encodes it into a context-vector c, which is used by the decoder-block to generate the target-sequence, taken from Cho et al. (2014b).

meaning it learns a probability distribution over that sequence. It is therefore possible to calculate the probability of some sequence x as

$$p(x) = \prod_{t=1}^{T} p(x_t|\{x_1, ...x_{t-1}\}). \tag{2.3}$$

In the sequence-to-sequence (seq2seq) setting that Encoder-Decoder architectures are usually applied to, the task is generating an output sequence $y = \{y_1, ..., y_{T'}\}$ from the input sequence $x = \{x_1, ..., x_T\}$, where T' may be different from T. Both encoder and decoder are RNNs, so in order to avoid confusion, the hidden-state of the encoder at a timestep t will be referred to as h_t, while hidden-states of the decoder will be called s'_t. Furthermore, timesteps of the decoder will be t'. The encoder is an RNN that processes the entire input sentence and, using the calculated hidden-states $\{h_1, ..., h_T\}$, outputs a high-dimensional, fixed-sized vector c, which can be regarded as a summary vector of the input sequence, using

$$c = q(\{h_1, ..., h_T\}), \tag{2.4}$$

where q is again a non-linear function. $q(\{h_1, ..., h_T\}) = h_T$ is often being used, meaning c will simply be the very last encoder hidden-state. The idea behind it is that any given hidden-state is dependent on the previous input and hidden-state, meaning that the last hidden-state h_T can be viewed as a summary of the entire sequence. As such, only the hidden-states produced by the encoder RNN are of relevance, while their outputs are not relevant. The encoder therefore encodes a summary of the input sequence x into the context-vector c and passes it on to the decoder, which is again an RNN. The

decoder is trained to predict the next word $y_{t'}$ in the output sequence similarly to how it is described in Eq. 2.2 and Eq. 2.3. However, it uses not only its previous prediction and hidden-state to do so, but also takes c into account: the hidden-state of the encoder RNN is calculated using

$$s'_t = f(y_{t'-1}, s_{t'-1}, c), \tag{2.5}$$

and the probability of the next token $y_{t'}$ is given as

$$p(y'_t|\{y_1, ...y_{t'-1}\}, c) = g(s'_t, y_{t'-1}, c), \tag{2.6}$$

where g is a non-linear, potentially multi-layered function that outputs a probability distribution over the vocabulary. As such, Eq. 2.3 is also updated to include c and the probability of the output sequence y is given as

$$p(y) = \prod_{t'=1}^{T'} p(y'_t|\{y_1, ..., y_{t-1}\}, c).$$

In summary, the decoder is tasked with outputting the next element in a sequence based on its previous hidden-state, previous prediction, and information from the input sequence as encoded in the context-vector. Lastly, assuming x to be an input sequence from the training set and y to be the corresponding target sequence, encoder and decoder are jointly trained to maximize the conditional probability that y is generated given x:

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log_{p_0}(y_n|x_n),$$

where θ are the model's parameters.

2.5 Attention and alignment

The Encoder-Decoder architecture described in Section 2.4 handles sequence-to-sequence (seq2seq) tasks by using an encoder to encode the input sequence as a fixed-sized, high-dimensional context vector and a decoder that generates the target sequence while making use of this vector. This means that the information in the input sequence, regardless of its length, has to be condensed into this single fixed-sized vector, which is a potential bottleneck. Indeed, it has been shown that performance of encoder-decoder models decreases quickly with increasing length of the input sentence in the case of neural machine translation (Cho et al., 2014a). Attention (Bahdanau et al., 2014; Luong et al., 2015) addresses this issue by allowing the decoder to focus on those parts of the input sequence that are most relevant for any individual output element. Instead of using a constant context-vector throughout the sequence generation, attention calculates a new context-vector for each predicted element, by computing a weight for each hidden-state of the encoder based on how relevant it is for the current prediction, and the aggregating these weighted hidden-states into a context-vector. This means that an attention-mechanism also needs to learn an alignment between the input- and output-sequences to some extent

in order to be able to calculate a meaningful context-vector. Attention was first applied to the task of neural machine translation, i.e., the task of translating from one language to another, where it was found that attention indeed finds a linguistically plausible alignment between the two sequences. This means that attention-mechanism have been observed to learn to jointly align and translate. Figure 2.4 shows a schematic overview of an attention-mechanism. The rest of this section is dedicated to a formal explanation of attention.

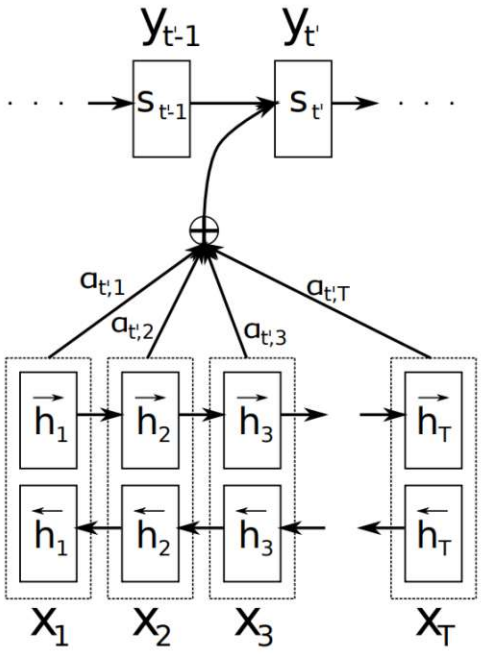


Figure 2.4: Schematic overview of an attention-mechanism in an encoder-decoder architecture, taken from Bahdanau et al. (2014) but slightly changed to account for differences in notation. A context vector for timestep t' is computed by calculating a weighted sum over all encoder hidden-states. The weight for each hidden-state is dependent on how well that hidden-state and the previous hidden-state of the decoder match.

An Encoder-Decoder architecture with attention adapts Eq. 2.6 by using a dynamic context-vector $c_{t'}$ at each timestep t':

$$p(y'_t|y_1, ..., y_{t'-1}, x) = g(y_{t'-1}, s_{t'}, c_{t'}),$$

where x is the input sequence, and hidden-state $s_{t'}$ is calculated as

$$s_{t'} = f(s_{t'-1}, y_{t'-1}, c_{t'}).$$

Another change made to the architecture from Section 2.4 is the use of bidirectional RNN (Schuster and Paliwal, 1997) in the encoder. This means that the input sequence

that is to be encoded is read both forwards and backwards, meaning once from x_1 to x_T and once the other way around. This results in forward-hidden-states $\{\overrightarrow{h_1}, ..., \overrightarrow{h_T}\}$ and backwards-hidden-states $\{\overleftarrow{h_T}, ..., \overleftarrow{h_1}\}$. The full hidden-state of the encoder at a timestep t is then given as the concatenation of that specific forwards- and backwards-hidden-state:

$$h_t = [\overleftarrow{h_t}; \overrightarrow{h_t}].$$

This makes information from the full input sequence available at any timestep t, since $\overrightarrow{h_t}$ will contain information on the entire input sequence up to element t, while $\overleftarrow{h_T}$ will contain the information after element t. This means that the information in h_t can be thought of as representing parts of the full input sequence with a focus on the element of the input sequence at position t.
The context-vector $c_{t'}$ is then calculated as

$$c_{t'} = \sum_{t=1}^{T} \alpha_{t't} h_t,$$

meaning it is a weighted sum over the encoder hidden-states. The weight $\alpha_{t't}$ represents how relevant h_t is for y'_t, the output token generated by the decoder at timestep t'. It is calculated as

$$\alpha_{t't} = \frac{exp(e_{t't})}{\sum_{k=1}^{T} exp(e_{t'k})}, \tag{2.7}$$

with $e_{t't}$ being an alignment score generated by an alignment model a

$$e_{t't} = a(s_{t'-1}, h_t),$$

which represents how well the encoder hidden-state of position t matches with previous decoder hidden-state $s_{t'-1}$. Bahdanau et al. for instance use a feed forward network to parameterize the alignment model a (Bahdanau et al., 2014).
The above explanations describe attention as it was first introduced by Bahdanau et al.. Luong et al. explore these ideas and slightly change certain aspects in their experiments, such as using the current decoder hidden state $s_{t'}$ instead of the previous one $s_{t'-1}$ when calculating attention-scores, or not using a bidirectional RNN for the encoder (Luong et al., 2015). Most impotently, they experiment with different scoring functions a such as dot-product attention

$$a = h_t^\intercal s_{t'}, \tag{2.8}$$

general attention, with W being a learned matrix of weights

$$a = h_t^\intercal W s_{t'}, \tag{2.9}$$

and concat attention, with v being a learned vector of weights

$$a = v^\intercal tanh(W[h_t^\intercal; s_{t'}]).$$

Interestingly, dot-product attention only involves a simple mathematical operation and does not use learned weights.

2.6 DRS Representation for seq2seq models

DRS can be represented in many possible ways. For human-readability, the box-format shown in Figure 2.1 is commonly used. However, for the purpose of representing a DRS in a machine-readable way, e.g. for use in a DRS parser, a number of other formats are available. Section 2.3 discusses multiple DRS parsers that cast a DRS to a graph or a tree. The PMB on the other hand contains DRS in the form of a flat list of clauses (also shown in Figure 2.1), which is the form that is used by the seq2seq DRS parsers proposed by Noord et al. (van Noord et al., 2019, 2020). Most of these representations face the problem that individual DRS-clauses contain box- and discourse-variables which are arbitrarily named, meaning one could produce any number of semantically equivalent DRS with different variable names.
Noord et al. identify this as an issue for their seq2seq model and devised a relative naming scheme for the clause format. The list of clauses is read top to bottom and each variable is renamed based on when it was introduced, which is done for box-variables and discourse-variables separately. Figure 2.5 shows an example of this for the sentence *Tom isn't afraid of anything*, from van Noord et al. (2018b). It displays the normal DRS in clause format on the left and the rewritten DRS on the right. The first clause of the rewritten DRS (at the very top) contains a *[NEW]* and *<NEW>* to indicate a new (not before encountered) box- and discourse-variable respectively. The following two clauses then refer to these newly introduced variables with *[0]* and *<0>* respectively. In the lower half of the DRS, a number of rewritten clauses that point to previously and subsequently introduced variables can be see. For example, *[0] Time <0> <-1>*, which refers back to the previously introduced discourse-variable (-1) or *[0] Stimulus <0> <1>*, which refers to the discourse-variable that will be introduced next.

This naming scheme eliminates the need for seq2seq models to learn how to deal with such arbitrary variable names in the input. (van Noord et al., 2018b) show that indeed, rewriting variable names as described above can slightly increase performance of a DRS parser. In their experiments, they observe an increase of between 0.7 and 1.0 points of F-Score. After training a parser on DRS in the representation, the output DRS will also use the representation and need to be translated back to normal DRS with absolute variable names.

2.7 Noord et al. Seq2Seq pure DRS Parser

This section describes the neural seq2seq DRS parser created by Noord et al. (van Noord et al., 2020), which this work will use as a base for producing a model that can generate DRS that includes an alignment to the input sequence. While they propose multiple different models, we chose the model that uses only BERT-embeddings as input and also only a single encoder. The model uses a standard Encoder-Decoder architecture with attention as described in Section 2.4 and Section 2.5. An overview of this model is displayed in Figure 2.6. Encoder-Decoder models are trained on pairs input- and output-sequences. The natural language input sequence is embedded using

```
b1 REF x1                    [NEW] REF ⟨NEW⟩
b1 male "n.02" x1            [0] male "n.02" ⟨0⟩
b1 Name x1 "tom"             [0] Name ⟨0⟩ "tom"
b2 REF t1                    [NEW] REF ⟨NEW⟩
b2 EQU t1 "now"              [0] EQU ⟨0⟩ "now"
b2 time "n.08" t1            [0] time "n.08" ⟨0⟩
b0 NOT b3                    [NEW] NOT [NEW]
b3 REF s1                    [0] REF ⟨NEW⟩
b3 Time s1 t1                [0] Time ⟨0⟩ ⟨-1⟩
b3 Experiencer s1 x1         [0] Experiencer ⟨0⟩ ⟨-2⟩
b3 afraid "a.01" s1          [0] afraid "a.01" ⟨0⟩
b3 Stimulus s1 x2            [0] Stimulus ⟨0⟩ ⟨1⟩
b3 REF x2                    [0] REF ⟨NEW⟩
b3 entity "n.01" x2          [0] entity "n.01" ⟨0⟩
```

Figure 2.5: DRS in flat clause format (left) raw and rewritten with relative variable names (right) for the sentence *Tom isn't afraid of anything*, take from van Noord et al. (2018b). *[NEW]* indicates the first time a new box-variable is encountered and *<NEW>* is the indicator for discourse-variables.

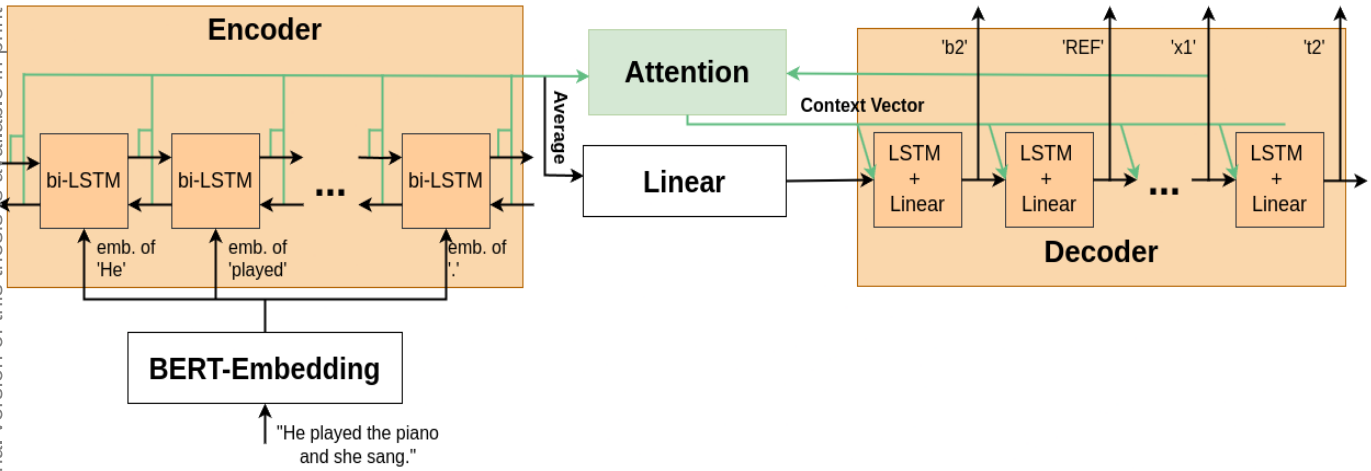


Figure 2.6: Schematic overview of the seq2seq neural DRS parser that this work extends. It is Encoder-Decoder model with BERT-embeddings for representing the input sequence, an encoder based on biLSTM, a cross-attention-mechanism, and a decoder using a unidirectional LSTM and a linear layer at each timestep. The first decoder hidden-state is initialized by a linear layer, which receives an average of all encoder hidden-states as input.

a frozen BERT-embedding layer. The output sequence is embedded using trainable, non-contextual, pre-trained GloVe-embeddings (Pennington et al., 2014) and then fed to the decoder. During inference time, the decoder operates in an autoregressive fashion, receiving its previous output as input. As a scoring function for the attention-mechanism, dot-product attention (see Eq. 2.8) is used. The encoder uses a bidirectional LSTM, while the decoder only employs a forward-reading LSTM. Apart from that, certain small changes based on Nematus, a Toolkit for Neural Machine Translation (Sennrich et al., 2017), are made. These are: all encoder hidden-states are averaged and fed to an extra linear layer that is inserted between encoder and decoder. The output of that linear layer is then used to initialize the decoder hidden-state. The initial decoder hidden-state is therefore defined as

$$s_0 = tanh(W c_{tok}),$$

with c_{tok} being the average of encoder hidden-states defined as

$$c_{tok} = (\sum_{t=1}^{T} h_t)/T)$$

Also, an additional linear layer is added after each encoder state. The target sequence is presented not at DRS-clause-level but rather on an element-level, meaning that a DRS-clause, for example *b2 REF x1*, is not produced as a whole at a given decoder state, but rather that the individual elements *b2*, *REF* and *x1* are produce in subsequent timesteps.
While not technically a part of the model, Noord et al. employ several post-processing steps. Since the described model generates DRS in an End-to-End fashion, thus having no explicit model of the underlying problem, it is not guaranteed that the produced output is syntactically and semantically valid DRS. As such, the validity of a generated DRS is verified using a (non machine learning) clausal form checker that checks if a given DRS is semantically interpretable. The checker distinguishes between box- and discourse-variables and performs a number of checks, such as finding whether the DRS are well-formed or not (for example there is no box-variable where only a discourse-variable should be and vice versa), checking that discourse-variables have been explicitly introduced and inspecting if segmented DRS actually contain the boxes that their discourse relation is parameterized with. Lastly, it also checks if there is a main DRS (with the other DRS being either presuppositional or nested within the main DRS) and makes sure there exist no loops in the subordinate relations. If a DRS fails these checks, it is considered invalid, and is replaced with a dummy-DRS that will always count as incorrect for the purposes of evaluation (as described in Chapter 5). While our work is concerned with the task of parsing DRS with alignment, these validation steps can still be applied to the DRS-part of each clause, meaning we also replace invalid DRS with dummy DRS (and a dummy alignment).

CHAPTER 3

Method

This chapter explains the approaches for extending the DRS parser discussed in Section 2.7 so that it also produces the alignment from DRS-clauses to input sequence, which is found in the training data provided by the PMB (see Section 2.2). The underlying model architecture is not changed in any significant way, instead the alignment is generated based on the attention-mechanism or by changing the output sequence in the training data to include alignment. The methods proposed in this work are: 1.) an approach based on the capability of a cross-attention-mechanism in an Encoder-Decoder model to learn an alignment between the input- and target-sequence (see Section 2.5). For this approach, the alignment is generated by choosing the most relevant input-element for a given output-element as identified by the attention-mechanism. 2.) An End-to-End approach where the data that is used to train the model is changed to include the alignment and the pre- and post-processing are adapted accordingly. 3.) An approach that combines the previous two, using the attention-mechanism to generate alignment in the same fashion as 1.) but is trained as in 2.) (the alignment in the output-sequence is ignored in this case), facilitating a better learning of the alignment within the attention-mechanism. A schematic overview of the three approaches is shown in Figure 3.1, while the rest of this chapter is dedicated to describing them in more detail. To avoid confusion, we explicitly distinguish between pure DRS and DRS with alignment, where a pure DRS-clause would for example be *b2 REF x1* and a DRS clause with alignment would be *b2 REF x1 % He [0...2]* (and the former can be said to be the pure DRS-part or just DRS-part of the latter).

3.1 Alignment through Attention

Attention-mechanisms in an Encoder-Decoder architecture (Bahdanau et al., 2014; Luong et al., 2015) have been observed to be capable of learning an alignment between input- and output-sequence and have been used extensively (Garg et al., 2019; Xu et al., 2015;

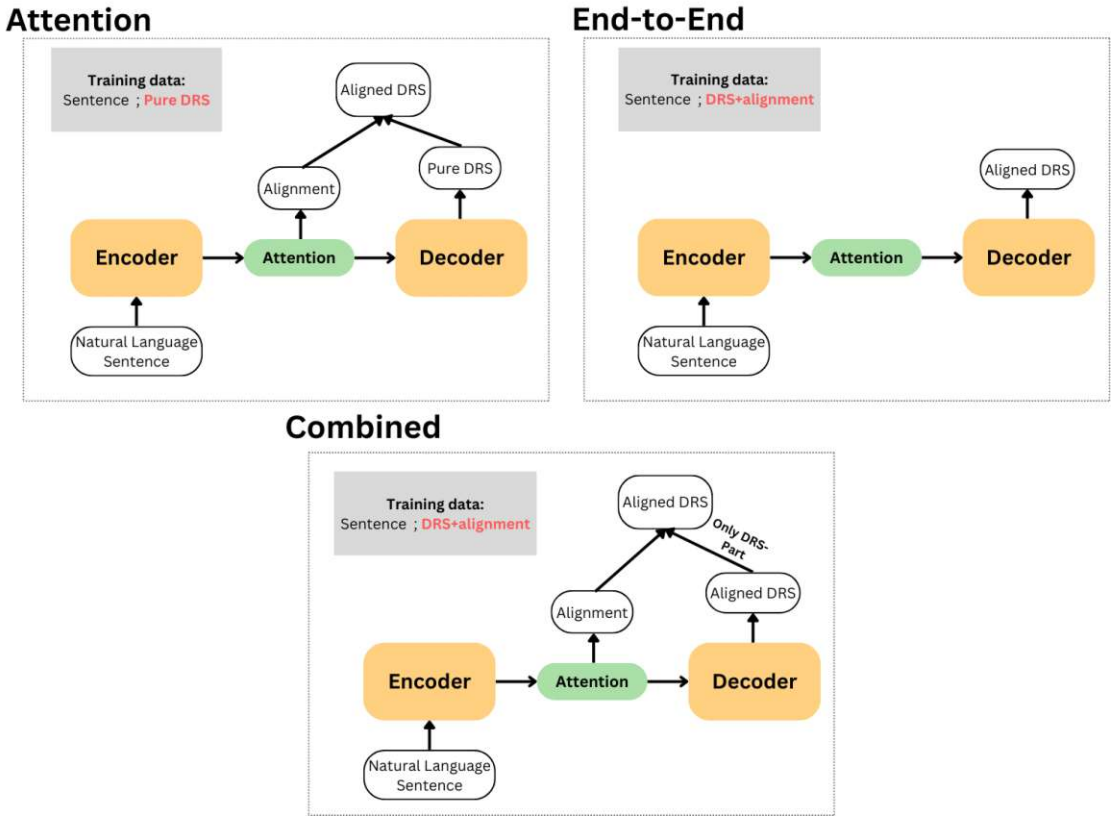


Figure 3.1: Schematic overview of three strategies that extend a DRS parser to also produce alignment. The Attention-approach extracts alignment from an attention mechanism, the End-to-End approach is trained to produce DRS alongside alignment in an End-to-End fashion. The Combination approach is trained like in the End-to-End approach but only the DRS-part of the decoder output is used, while the alignment comes from the attention-mechanism like it is done for the Attention approach.

Fu et al., 2016). These mechanisms use a scoring-function that expresses to what extent a given element in the output sequence matches all the elements in the input sequence. These scores are then used to calculate a weighted sum of each encoder hidden-state (i.e. of the input sequence) which serves as a context-vector for the decoder, allowing it to focus on parts of the input sequence that are most relevant for the element that is currently being generated by the decoder (see Section 2.5). While this can increase performance of Encoder-Decoder models, the attention-scores can also constitute an alignment between the two sequences. For the task of neural machine translation (i.e. translating between sequences of different languages), it has been shown that the alignment is linguistically plausible. As such, it may also be the case that for the task of DRS parsing, attention-scores constitute an alignment between the generated DRS-clauses and the input sequence. However, the following issues have to be addressed:

1. The seq2seq model this work is based on produces output not on a clause but on an individual element level. This means the parser does not output e.g. *b1 REF x1* in one decoder timestep but rather *b1* followed by *REF* followed by *x1* (or, more precisely, its representation after being rewritten using relative variable names as described in Section 2.6).

2. Sentences in the PMB have correct end-of-sentence-punctuation (i.e '.', '!', etc.), however, these elements in the input sequence are never used in the alignment, meaning no DRS-clauses are aligned to such tokens.

The basic idea for this method of alignment generation is to take the input-sequence token with the largest alignment-score (as calculated in Eq. 2.7) to the currently produced element in the decoder and use it as the alignment for that element. Formally, this means given an input sequence $x = \{x_1, ..., x_T\}$ and corresponding (encoder-) timesteps $\tau = \{1, ..., T\}$, selecting the timestep t such that h_t, the hidden-state of the encoder corresponding to x_t, has the maximum alignment-score $\alpha_{t't}$ at decoder timestep t' out of all timesteps in τ:

$$\underset{t \epsilon \tau}{\text{argmax}} \quad \alpha_{t't} = \underset{t \epsilon \tau}{\text{argmax}} \quad \frac{exp(e_{t't})}{\sum_{k=1}^{T} exp(e_{t'k})}. \tag{3.1}$$

In order to address 1., attention-scores of individual elements generated by the decoder that belong the same DRS-clause (as indicated by a special separator-element '***' that separates DRS-clauses) are averaged and the selection of the input-position with the largest alignment score is performed on this average (i.e. on DRS-clause-level) instead of on the individual alignment-scores. Let $y = \{y_1, ..., y_{T'}\}$ be the sequence of individual elements generated by the decoder and let $C = \{C_1, ..., C_N\}$ be the disjoint set of DRS-clauses where each clause $C_n = \{y_{t'}, ..., y_{t'+l}\}$ is a continuous subset of y and $y = C_1 \cup C_2 \cup ... \cup C_N$. For a given DRS-clause C_n, let τ_n be the set of timesteps corresponding to individual elements $y_{t'}$ that make up C_n. The clause-level alignment score is then calculated as:

$$\alpha_{nt} = \frac{\sum_{t' \epsilon \tau_n} \frac{exp(e_{t't})}{\sum_{k=1}^{T} exp(e_{t'k})}}{|\alpha_{nt}|}.$$

and the position of the encoder corresponding alignment is calculated as

$$t_{alignment} = \underset{t \epsilon \tau}{\text{argmax}} \quad \alpha_{nt}.$$

For dealing with 2, τ_n is restricted such that it only contains timesteps t where the corresponding element in the input sequence is not end-of-sentence punctuation (and not any special start- or end-of-sentence symbols that used by some NLP-framework), i.e.

$$\tau_n = y_t \notin \text{EOSP} | t \in \tau_n$$

where *EOSP* is the set of end-of-sentence punctuation characters. Figure 3.2 shows the heatmap of attention-scores resulting from applying the method described above for the

sentence *He played the piano and she sang.*
This method of alignment-generation can be seen as a search through the input-sequence for the most fitting element for each produced DRS-clause. This means that this approach will always produce a valid token from the input as alignment. Also, apart from the token, the desired alignment also contains the start- and end-index of that token within the input sequence (see Section 2.2). When the correct alignment-token is produced, attention through alignment will also always output the correct index-range, since looking up the start- and end-indices of a token within a sequence is trivial. Since the alignment is produced by calculating it from the attention-mechanism already present in the model, this method can generate alignment without making any changes the underlying pure DRS parser.
However, there is a problem for clauses that do not have any alignment. While it is very rare, the PMB does contain DRS-clauses that are aligned to no token of the input sequence, while the method described above will always result in some input-token being chosen as alignment.

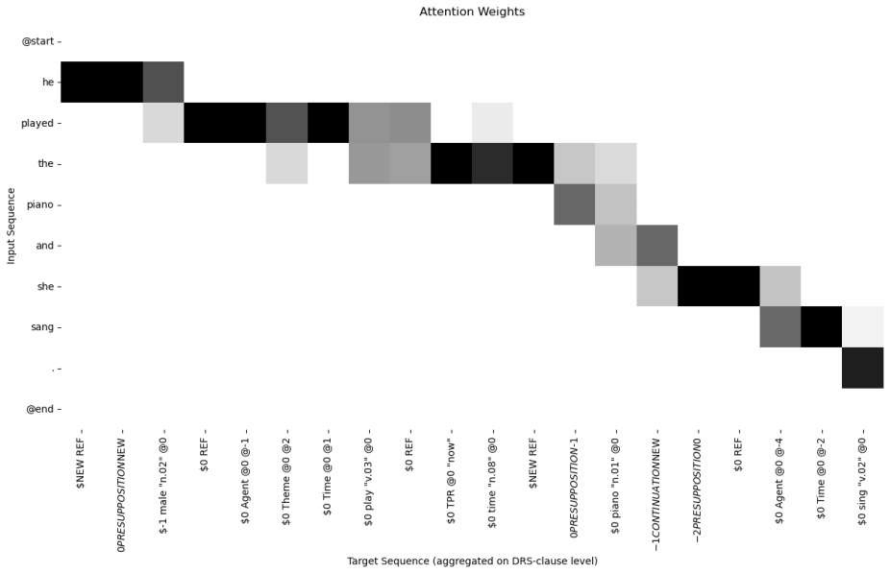


Figure 3.2: Heatmap of attention-scores for the sentence *He played the piano and she sang.* on an aggregated DRS-clause level (PMB release 3.0.0), using bilinear / general attention. DRS-clauses are presented on the x-axis in the relative naming scheme for variables. The input sequence on the y-axis includes technical tokens "@start" and "@end", which are being added by AllenNLP, the NLP-framework used for implementation. These tokens are never chosen as alignment.

3.2 Alignment in an End-to-End fashion

While DRS are complex, recursive structures with strict requirements for what constitutes a syntactically and semantically valid DRS, the End-to-End model employed by Noord et

al. (see Section 2.7) performs this complex task very well. As such, it seems reasonable to assume that this model would also be able to learn the task of alignment-generation on top of DRS parsing. Therefore, the same model as with the previous approach is trained, but the training-data now also explicitly contains the alignment. This means that a clause is given as e.g. *b2 REF x1 % He [0...2]* instead of just *b2 REF x1*. Unlike the Alignment through Attention-approach, this method affects the underlying DRS parser, as it uses different training data.
For this approach, alignment-generation is treated as a string-generation task. The model needs to generate the token and indices of the alignment in the same fashion that it generates DRS. It has no mechanism that allows it to directly copy any part of the input sequence. This means that, unlike the Alignment through Attention-approach, such a system is not guaranteed to output an element which can also be found in the input sequence. Furthermore, there is also no guarantee that the produced start- and end-indices of the alignment will also be correct if the predicted alignment-token is correct, or that the index-range will correspond to the size of the generated token. While this may seem like very serious disadvantages compared to the approach described in Section 3.1, it is also not possible to ensure that a seq2seq model outputs valid DRS, but they perform very well on such a complex task regardless.
Since for this approach, the task that the model is trained on, string-generation, is different from finding an alignment between input- and output-sequence, post-processing to find the most similar token in the input sequence for the generated alignment is applied. Specifically, for each produced alignment-token that does not exactly match one of the input-tokens, the Levenshtein distance is calculated between that alignment-token and each input-token. The input token with the lowest Levenshtein distance is chosen as the final alignment. This means that the chosen token from the input sequence replaces the alignment-token that was initially produced by the model in the output.
Formally, let $x = \{x_1, ..., x_T\}$ be the input sequence (in natural language) and $y = \{y_1, ..., y_{T'}\}$ be the output sequence (individual elements of DRS-clauses with alignment) and let $C = \{C_1, ..., C_N\}$ be the disjoint set of DRS-clauses as described in Section 3.1 but also including alignment for each clause C_n. Let $y_n^{AlignTok}$ be the element in C_n that is the generated alignment-token. The final alignment-token after post-processing is found using:

$$\max_{y_t \epsilon y} \quad L(y_n^{AlignTok}, y_t),$$

where L is the Levenshtein distance (Levenshtein, 1966; Hyyrö, 2001).
This method guarantees that the produced alignment-token will be one of the tokens from the input sequence.

3.3 Combining Attention and End-to-End alignment

The previously described approaches generate alignment in different ways. One simply reads from an attention-mechanism that was already present in the original DRS-parser, while the other makes changes to the training data and pre- and post-processing and

tries to produce alignment in the same fashion it generates DRS (End-to-End). While the underlying DRS-parser uses an attention-mechanism in both cases, the alignment is only extracted from this attention-mechanism for the first method. However, it is of course also possible to read the alignment from the attention-mechanism for the second method, and using it as the final alignment instead of the alignment that was produced as part of the output, thus combining the two approaches. Such a combined system could solve the following issues:

- Attention-mechanism can learn to align very well between input- and output sequence, even if producing such an alignment is not part of the task, as was shown for e.g. machine translation (Bahdanau et al., 2014). However, the heatmap of attention-scores for Alignment through Attention-method in Figure 3.2 shows some DRS-clauses where corresponding alignment does not seem entirely clear. For example, we can see that for the very first clause on the x-axis (*$ NEW REF*, using relative naming scheme as in Section 2.6) all the mass of the attention-scores is centered on a single input token, *he*. However, for certain other clauses, most noticeably sixth from the left *$0 piano "n.01" @0*, where attention-scores are fairly evenly spread across three tokens of the input sequence, this is not the case. This suggests that in some cases, the model fails to align to exactly one token from the input sentence. A possible explanation could be that for the Alignment through Attention-method, the alignment was not part of the output-sequence the parser needs to produce, thus the system has no need to explicitly learn that alignment beyond what would be useful for the task of pure DRS-parsing. The alignment that can be read from the attention-mechanism may therefore be improved if the DRS parser is trained to also produce the alignment as output.
- While an alignment generated from the attention-mechanism has the benefit of always being a token that was selected from the input sequence, it cannot produce an empty/missing-alignment, even though a small fraction of the PMB-data has missing alignments. The alignment generated in End-to-End fashion on the other hand can be empty (or some token representing an empty alignment that is not found in the input sequence). However, a combined system could simply pick and choose, taking the alignment produced in the End-to-End fashion if it is empty and taking alignment in attention-alignment fashion otherwise.

Indeed, it can be seen in Figure 3.3 that for the Combination approach, attention-scores for each clause in the sentence *He played the piano and she sang*, are much more concentrated on a single element of the input sequence compared to the attention-score-heatmap in Figure 3.2. The y-axis again shows the tokens of the input sequence, while the x-axis shows DRS-clauses resulting from concatenating the individual outputs that the decoder generated, only this time also including an alignment.

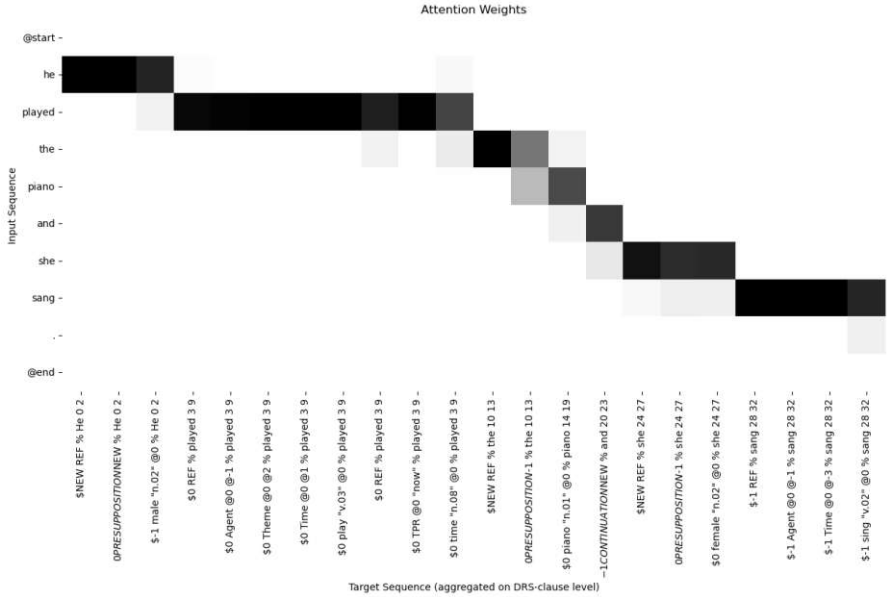


Figure 3.3: Heatmap of attention-scores for the sentence *He played the piano and she sang.* on an aggregated DRS-clause level (PMB release 3.0.0) for the Combination alignment-generation method.

CHAPTER 4

Experiments

This chapter describes the experiment setup and technology used in this work. Processing of the data is explained, an overview of the model-architecture is given and hyperparameters are provided. Lastly, experiments with the alignment-generation methods proposed in Chapter 3 are described.

Data & Setup
All experiments are conducted using the English data of the 3.0.0 release of the PMB (see Section 2.2). In particular, the silver-quality data and the train-set of the gold-quality data are used for initially training the model, followed by fine-tuning on only the train-set of the gold-quality data. Initial training on the gold + silver data is performed once using a seed 2222. The fine-tuning is done for five runs over predetermined seeds 2222, 3333, 4444, 5555 and 6666. Experiments are therefore evaluated over fiver runs. In order to save resources, the maximum number of epochs (for both initial training and fine-tuning) was capped at 4. The models are implemented in the open-source NLP framework AllenNLP (Gardner et al., 2018), specifically, a fork of AllenNLP version 0.9 (as this is used by the underlying DRS-only parser). For sentences in the PMB, proper names are indicated with a '~', i.e *New York* would be *New~York*. However, the data used in the experiments (and also for DRS-only parser of Noord et al.) are the raw input sentences, i.e. before being tokenized as in the PMB. The DRS and the alignment on the other hand (and the alignment) are used exactly in the format found as found in the PMB.

Pre-Processing
The (raw) natural language input sentences are tokenized using the NLTK-Toknizer (Bird and Loper, 2004). DRS are rewritten to a relative variable naming scheme, as described in Section 2.5, and the alignment for each DRS-clause is either kept (End-to-End and Combination approach) or discarded (Alignment through Attention-approach). If the alignment is kept, the '%'-character is used as a separator between the DRS-part of a

clause and the alignment-part. In order to separate entire clauses, '***' is used as a separator.

Post-processing
The produced DRS including alignment is post-processed: For the DRS-part of the resulting clauses, rewriting back to absolute variable names as well DRS-validation as described in Section 2.7 is performed. Any DRS found to not be semantically interpretable are entirely removed and replaced by a dummy-DRS that will have no matching clauses when compared to a goldlabel-DRS (see Chapter 5). Lastly, it is ensured that each clause includes an alignment, meaning that if, for a given clause, the produced alignment is malformed or an empty alignment was produced, a dummy-alignment *UNK [0...0]* is used instead.

Model architecture & Hyperparameters
As this work uses a pre-existing DRS-only parser and expands it to also generate alignment, the model architecture used during experimentation is almost identical to that employed by Noord et al. (van Noord et al., 2020) described in Section 2.7, the code for which is available on GitHub[1]. For the purposes of this work, only their single-encoder BERT-model is used, though the methods explored in this work should also work with any of their other setups. The model uses an Encoder-Decoder architecture with attention that is implemented on a fork of AllenNLP version 0.9. The architecture shown in Figure 2.6 uses a frozen BERT-base encoding layer for the input of the encoder-block. The encoder employs a bidirectional LSTM to encode the embedded input-sequence. The decoder uses a unidirectional LSTM, with each decoder-state being followed by a linear layer. During training time, the target-sequence that is being passed to the decoder is embedded using learnable GloVe-embeddings. An attention-mechanism is employed to allow the decoder to dynamically focus on parts of the input-sequence and compute a context-vector based on what is being focused on. For the scoring-function of the attention-mechanism, the dot-product function as described in Eq. 2.8 is being used. Lastly, the mean of all encoder hidden-states is fed to a linear layer that is used to initialize the hidden-state of the decoder. Hyperparameters are shown in Table 4.1. The methods for alignment-generation that this work proposes do not require changes to the underlying model, therefore, all experiments use the described architecture and hyperparameters with the only exception being the attention-mechanism, where we experiment with different types of attention (i.e. different scoring functions) and the number of epochs, which we limit to a maximum of 4, whereas Noord et al. used no such limit and employed early stopping. While the model-architecture is not changed, the End-to-End approach and Combination approaches use differently pre-processed training data (alignment not discarded), so they would likely benefit from an extensive parameter search to account for this change in setting. However, in order to save resources and since the focus of this work is alignment generation, this was not done.

Attention scoring-functions
A cross-attention-mechanism computes a context-vector for a given decoder timestep, by

[1] https://github.com/RikVN/Neural_DRS

Parameter	Value	Parameter	Value
Input Embedding		**Decoder**	
Type	bert-base-uncased	Type	LSTM
Input Embedding Size	768	Encoder Hidden Size	300
Max. # source tokens	125	Encoder LSTM Layers	1
trainable	false	max_norm	3
Target Embedding		scale_grad_by_freq	false
Type	pretrained GloVe	label_smoothing	0.0
Target Embedding Size	300	beam_size	10
Max. # target tokens	1160	max decoding steps	1000
trainable	true	schedule sampling	0.2
Encoder		**Trainer**	
Type	biLSTM	batch size	12
Encoder Hidden Size	300	optimizer	adam
Encoder LSTM Layers	1	learning rate	0.001
Attention		grad_norm	0.9
Type	dot-product / bilinear	max_epochs	4
normalize	true		
matrix_dim	- / 600		
vector_dim	- / 600		

Table 4.1: Hyperparameters used during the experiments. Except for the parameters in red text-color, all hyperparameters are equal to that of the underlying pure DRS parser of Noord et al.

calculating a weighted mean of all encoder hidden-states, where the weights are based on how relevant each encoder hidden-state is for the element that is being produced at that timestep. These weights are calculated using a scoring-function, as described in Section 2.5. The pure DRS parser of Noord et al. uses the relatively simple dot-product attention (Eq. 2.8 implemented in AllenNLP[2]). This may not be ideal for the purposes of learning an alignment, since it is based on a static calculation that does not include any learnable weights. For that reason, we also experiment with the scoring-function called general attention (see Eq. 2.9), also called bilinear attention. Its implementation is also available in AllenNLP[3]. The hyperparameters for the two attentions are shown in Table 4.1.

4.1 Alignment-generation

Alignment through Attention

For this approach, the DRS parser is trained on pure DRS for the target sequence, meaning the alignment found in the PMB is not used. For the type of attention, both

[2] https://docs.allennlp.org/main/api/modules/attention/dot_product_attention/

[3] https://docs.allennlp.org/main/api/modules/attention/bilinear_attention/

dot-product- and general-attention are considered. Otherwise, the model should be identical to the underlying pure DRS parser, even being trained on the same data, the only difference being the type of attention that is employed. Attention scores are aggregated and used to extract an alignment to the input sequence on DRS-clause level, as described in Section 3.1. Alignment and pure DRS is then combined to form the final output.

Alignment in an End-to-End fashion
For this approach, the alignment in the PMB data is kept and the model is trained to predict the alignment alongside each DRS-clause. The model is again the same as the underlying pure DRS parser, only the training data is different. Pre- and post-processing are adapted such that the alignment is ignored for any DRS-specific processing. Since for this approach, the alignment can be any token from the vocabulary as produced by the encoder, the produced alignment is mapped to the most likely candidate token from the input sequence, as described in Section 3.2. The Levenshtein distance (Hyyrö, 2001)[4] is used to determine the most likely candidate.

Combination approach
The previous two approaches are combined. Training data includes the alignment and general-attention is used instead of dot-product attention. Otherwise, model and hyperparameters are again identical to the pure DRS parser. An alignment is extracted from the attention-mechanism as with the Alignment through Attention-approach and is then used to replace the alignment generated as part of the output sequence. Only in cases where the End-to-End alignment indicates an empty alignment (a fraction of clauses in the PMB-data is missing an alignment), the End-to-End alignment is used as the final alignment, in all other cases it is overwritten by the attention-alignment.

[4] https://pypi.org/project/editdistance/

CHAPTER 5

Evaluation

This chapter explains the evaluation of DRS and its alignment to the input sequence. The metrics of F1-Score for the task of pure DRS parsing as it is implemented in Counter (van Noord et al., 2018a) and alignment accuracy are explained. A performance-comparison of the methods for generating alignment alongside DRS is given. Furthermore, a number of special cases in the target alignment that this work does not address are mentioned and the issue of having a task that comprises two sub-tasks, DRS parsing and alignment generation, is shortly discussed. Lastly, the implications of the results are explored.

5.1 Evaluating pure DRS

Comparing the DRS produced by a DRS parser to a goldlabel-DRS is not a trivial task. Since DRS are grounded in formal logic, it would be possible to translate them into first-order logic formula (Blackburn and Bos, 2005) and use a theorem prover to find whether the two DRS are semantically equal or not. However, this would only deliver a binary result that gives no information on how similar two DRS are other than exactly equal or not exactly equal. For that reason, given two DRS in flat clause format, they are compared by finding matching clauses and calculating an F-score based on that. Also, since the variable names in DRS are arbitrary, a hill-climbing approach is used to find a variable-mapping from the variables in one DRS to the other that maximizes F-Score. All of these steps are performed by the DRS-evaluation tool Counter (van Noord et al., 2018a), which is used for the evaluation in this work.
Figure 5.1 from van Noord et al. (2018a) shows an example where a DRS corresponding to the sentence *She fled Australia.* is considered as the gold-label DRS and a second DRS corresponding to the sentence *He smiled.* is compared to it. Green text-color indicates a clause for which a matching partner could be found within the gold DRS, red text-color indicates that no match could be found. This results in a number of true positives, false positives and false negatives, which are then used to calculate Precision, Recall and lastly

F1-Score, calculated as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}},$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}},$$

and

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

As mentioned above, matching is not based on simply comparing two given clauses, but also takes a variable mapping into account, i.e. variable *k0* in the first DRS may be called *b0* in the second DRS, as variable naming is arbitrary and independent. Such variable mappings are found by starting with a random initial variable mapping and then adding additional mappings if that would increase the F1-Score. As such an approach can easily get stuck in local optima, a fixed number of restarts are performed. For the example in Figure 5.1, the ideal variable mapping would be *k0->b0,e1->v1*, resulting in six true positives (predicted clauses that have a match in the gold-DRS), three false positives (predicted clauses that do not have a match in the gold-DRS) and seven false negatives (clauses from the gold-DRS that were not predicted). Lastly, since DRS contain arbitrary variable names and each variable needs to be introduced by REF-clause (e.g. *b1 REF x1* would introduce a variable *x1*), REF-clauses are ignored (not counted as true positives), as they can be considered trivial and would inflate the F1-Score. In Figure 5.1 they are therefore crossed out to indicate that they are not being counted. The example would result in an F1-Score of 54.5% with REF-clauses being counted and 40% without them.

5.2 Alignment Accuracy

The approaches for alignment generation proposed in this work expand a pure DRS parser to also output alignment alongside each produced DRS-clause. As such, each clause produced by such a system consists of two parts: the DRS-clause and the alignment (see 2.2). The alignment consists of a word or token from the input sequence, followed by its start- and end-index within the input sequence (e.g. *He [0...2]* for the sentence *He played the piano and she sang*). While this may suggest two aspects of the alignment that need to be evaluated, the token and the indices, this is not the case. Alignment based on attention will always produce a token from the input sequence, and once this token is chosen, generating the correct indices is trivial. While this is not initially true for the End-to-End approach, performing fuzzy-search to select a token from the input sequence based on the predicted alignment in post-processing leads to the same situation. As such, only the token of the alignment need to match for all proposed methods. For measuring alignment quality, only correctly predicted DRS-clauses (including REF clauses) are considered, and accuracy is defined as the ratio of such clauses that have been aligned to the correct input word. This means that if two DRS-clauses are found to be matching as

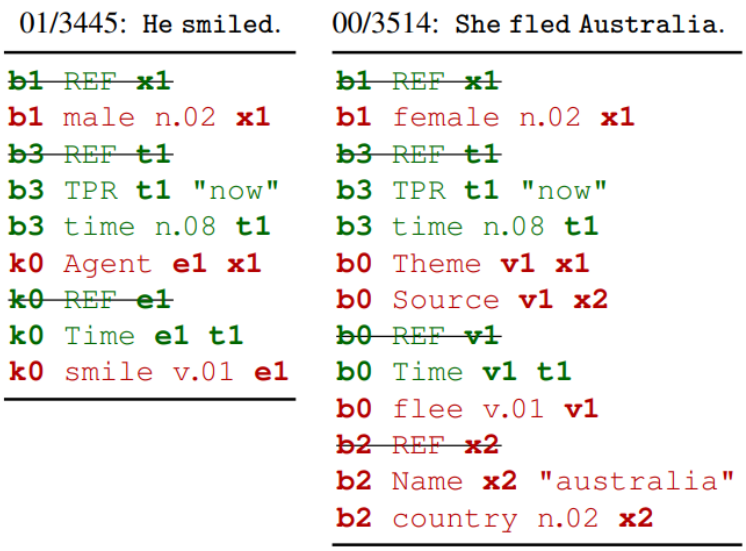


Figure 5.1: A DRS corresponding to the sentence *He smiled.* that is being compared to a gold-label DRS for *She fled Australia.* using Counter ,taken from van Noord et al. (2018a). Assuming a variable mapping *k0->b0,e1->v1*, green text indicates clauses where a match could be found while red text indicates that no match could be found.

in Section 5.1, it will be checked if their alignment also matches:

$$\text{Alignment Accuracy} = \frac{\text{Number of matching clauses with matching alignment}}{\text{Total Number of matching clauses}}$$

If, on the other hand, the DRS-part of a clause does not have a matching DRS-clause in the goldlabel-DRS, the predicted alignment is not evaluated, neither counting as correct nor incorrect. This is because if a predicted clause has no matching gold clause, there is nothing that the predicted alignment could be compared to. We therefore always calculate alignment accuracy only over correctly predicted DRS-clauses.
Lastly, while REF-clauses are ignored in pure DRS evaluation as they are considered trivial for the DRS parser to produce, producing a correct alignment for them is not trivial. Therefore, when it comes to computing Alignment Accuracy, REF-clauses are always included.

5.3 Special cases in the target alignment

The DRS in the PMB are aligned to a tokenized version of the raw input sentences where certain multi-token expressions are treated as a single token an joined together with a '~' (e.g *10~a.m*). However, the models are trained using raw input sentences, i.e. before any phrases such as proper names are identified as a single token. Furthermore, a small fraction of clauses has more than one alignment (see Section 2.2). However, the models are trained to predict exactly one (or in some cases potentially no) alignment for each

clause. Lastly, the alignments are cased, while the underlying pure DRS parser this work builds upon uses an uncased BERT-model for embedding the input sequence.
To address these three issues, the matching of a produced alignment to the target alignment is done slightly differently compared to the matching for exact equality that is usually applied:

- If the gold alignment is a single token that is made up of multiple tokens, i.e a token that contains a '∼', it is only verified whether the produced alignment is contained within the gold-alignment, as opposed to needing to be an exact match.
- If the gold-alignment consists of multiple alignments (i.e. the given DRS-clause is aligned to multiple elements from the input sequence), it is only verified whether the produced alignment corresponds to one of the gold-alignments.
- Casing is ignored for the purpose of evaluation alignments (be it when checking for an exact match or for inclusion, as in the above points).

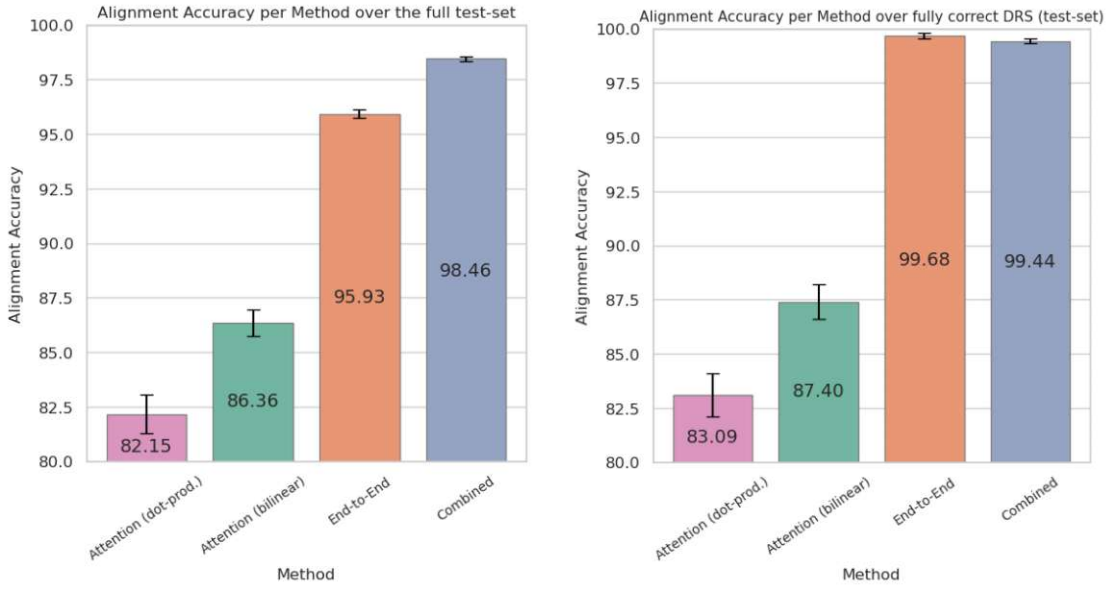
5.4 Evaluating Alignment on top of DRS

When evaluating the performance of a given DRS and alignment parser, it is difficult to separate the task of DRS parsing from alignment generation. While it is possible to calculate F1-Score on only the DRS-part of a clause, thus fully removing alignment-generation from the task of pure DRS parsing, the opposite is not as easy. While it makes sense to evaluate alignment only on correctly predicted DRS-clauses, this still leaves the issue of partially incorrectly parsed DRSs, i.e. DRSs with some matching clauses for which alignment is evaluated and some non-matching clauses for which alignment is ignored. For example, a parser may generate a DRS that looks nothing like the goldlabel-DRS, but that still contains a number of matching clauses by chance. In this scenario, alignment accuracy would be calculated for these matching clauses. However, since the generated DRS is very different from the goldlabel-DRS, it begs the question to what extent the system can be expected to successfully align the (faulty) DRS-clauses to the input sequence.
To address this issue to some extent, the evaluation of alignment-quality in this thesis is performed once on the full output of the parsers and once on the subset of that output where the pure DRS-part has been fully correctly parsed.

5.5 Results

All results of our experiments on the dev- and test-sets of the PMB 3.0.0 release are provided in Table 5.1. Figure 5.2a shows Alignment Accuracy of the proposed systems on the full data of the PMB 3.0.0 test-set, meaning that erroneous DRS (DRS with at least one incorrect clause) are also included. For the Alignment through Attention-approach, values for both dot-product attention and bilinear attention (see Chapter 4 and Section

2.5) are given. It can be seen that the model employing bilinear attention achieves significantly higher alignment accuracy with a mean of 86.36 than the model employing dot-product alignment with a mean of 82.15. This suggests that bilinear attention is better able to capture the desired alignment, which can also be seen visualized in Figure 5.3, which shows heatmaps of attention scores for (a) dot-product attention and (b) bilinear attention. These plots show that for bilinear attention, for each DRS-clause, most of the mass of attention weights is focused on one element of the input sequence, whereas it is much less focused for dot-product attention.
The relatively simple approach of not removing the alignment from the training data and thus generating alignment in an End-to-End fashion (and then mapping the produced alignment to the most likely candidate in the input sequence, which is done in post-processing), achieves a mean accuracy 95.93, vastly outperforming the attention-approaches. The combined system uses bilinear attention, as it has achieved better results for in the Alignment through Attention-approach. It performs best out of all compared models with a mean accuracy of 98.46.

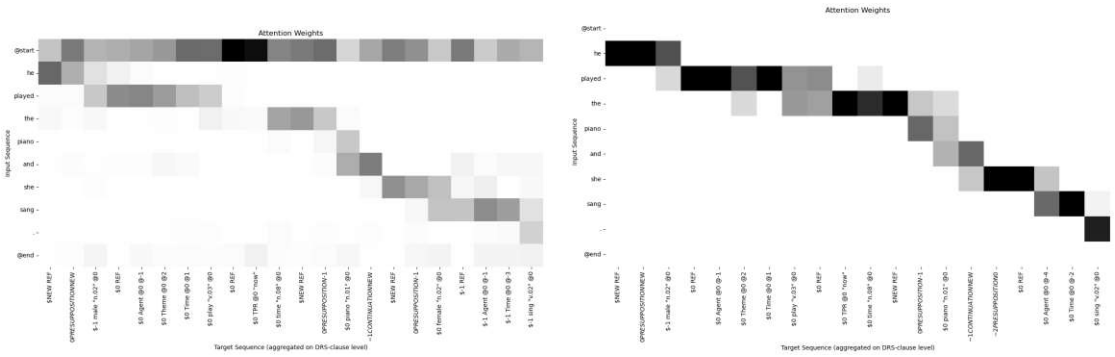


(a) Alignment accuracy test-set (b) Alignment accuracy correct DRS of test-set

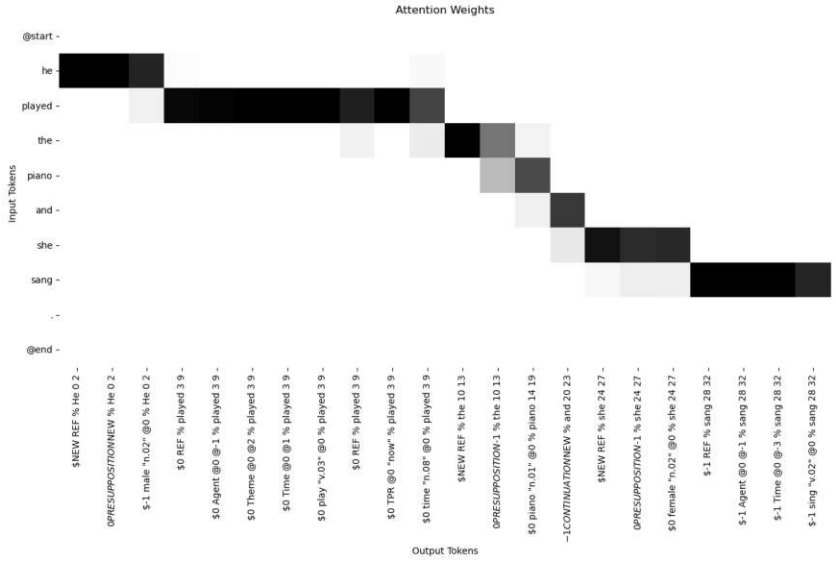
Figure 5.2: Comparison of alignment accuracy for the proposed methods on the English test-set of the PMB 3.0.0 release, once for the full test-set (a), and once for a subset of the data that contains only DRS that were fully correctly parsed (b). Results are shown over five runs and including REF-clauses.

Figure 5.2b shows performance of the employed methods for a subset consisting of the data where the DRS-part is fully correct. For this data, all systems can be seen to achieve a higher alignment accuracy compared to the evaluation on the full test-set. It is still the case that the attention alignment system based on bilinear attention outperforms

the one based on dot-product attention, and that both systems are outperformed by the End-to-End and Combination approaches, with a difference of more than 10%. However, for this scenario, the End-to-End approach is the best performing system with an accuracy of 99.68, marginally outperforming the Combination approach with 99.44.



(a) Attention scores for dot-product attention (b) Attention scores for bilinear attention



(c) Attention scores for the Combination approach using bilinear attention

Figure 5.3: Attention scores for the Alignment through Attention-approach, comparing dot-product attention and bilinear/general attention, and for the Combination approach using bilinear attention. Scores are aggregated on DRS-clause level as described in Section 3.1.

While this work's main focus is the parsing of alignment, the performance of any model on the underlying task of pure DRS parsing is also relevant. Since alignment can be clearly separated from the DRS in each clause, performance on the task of pure DRS parsing can also be evaluated. Figure 5.4 shows F1-Scores for the task of pure DRS parsing on the test-set. The attention-approaches, which are trained on the task of pure

DRS parsing, perform very similar to each other (87.10 and 87.17) and significantly better than the End-to-End and Combination approach (85.74 in both cases), which were both trained on the task of DRS plus alignment parsing. None of the models trained during the experiments achieve the same F1-Score of 88.53 as the underlying model of Noord et al. (van Noord et al., 2020) even though the Alignment through Attention-model (with dot-product attention) uses the exact same architecture. This is likely because the experiments in this work were limited to 4 epochs in order to save resources. Noord et al. do not explicitly mention for how many epochs the specific model that this work is using as base was trained for, however, their earlier work on an almost identical model (van Noord et al., 2018b) mentions between 13-15 epochs. All results discussed up to this point, and results on the dev-set, are also given in Table 5.1.

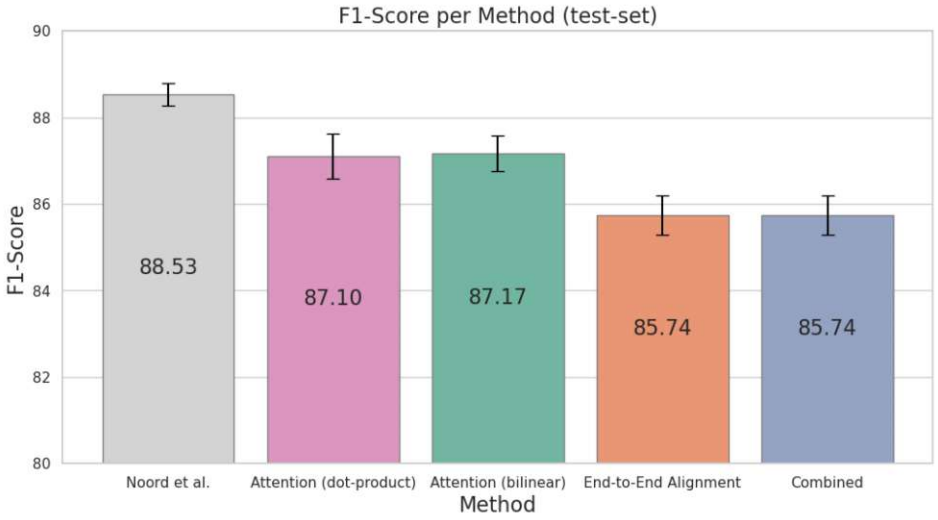


Figure 5.4: Comparison of F1-Score for the proposed methods on the English test-set of the PMB 3.0.0 release, using only the pure DRS-part of each clause (without alignment).

5.6 Discussion

The Alignment through Attention-approach, i.e .the approach that trains a model as a pure DRS parser and then extracts the alignment from the cross-attention-mechanism employed by that parser, shows drastically lower performance than the other two approaches, which are trained on predicting the alignment as output. However, they still manage to predict the correct alignment in more than 82% of clauses. This suggests that alignment between DRS and the input sequence is learned even if the alignment is not explicitly part of the target sequence that needs to be generated. However, requiring alignment as part of the output seems to improve how well this is learned, as indicated by the better performance of the Combination approach and the more clear assignment of attention weights to a single input token in Figure 5.3c compared to Figure 5.3b.
For the End-to-End and Combination approaches, it is not completely straightforward to

Set	Approach	Accuracy	Accuracy (corr. DRS)	F1-Score
Dev	Noord et al.	-	-	87.58 ± 0.19
	Attention (dot-prod.)	82.15 ± 0.91	83.33 ± 0.91	86.69 ± 0.25
	Attention (bilinear)	86.34 ± 0.59	88.08 ± 0.54	86.40 ± 0.48
	End-to-End	95.84 ± 0.19	99.56 ± 0.09	84.89 ± 0.30
	Combination	98.49 ± 0.13	99.33 ± 0.08	84.89 ± 0.30
Test	Noord et al.	-	-	88.53 ± 0.26
	Attention (dot-prod.)	82.15 ± 0.88	83.09 ± 1.00	87.10 ± 0.52
	Attention (bilinear)	86.36 ± 0.60	87.40 ± 0.81	87.17 ± 0.41
	End-to-End Alignment	95.93 ± 0.19	99.68 ± 0.12	85.74 ± 0.46
	Combination	98.46 ± 0.11	99.44 ± 0.11	85.74 ± 0.46

Table 5.1: Performance of the proposed methods with standard deviations. Accuracy refers to the alignment accuracy on the full English dev- and test-set of the PMB 3.0.0 release, Accuracy (correct DRS) means accuracy on a subset that contains only fully correctly parsed DRS, and F1-Score is calculated using Counter and relates only to the DRS-part of any given clause. The shown results are mean values with standard-deviations for five runs. REF-clauses are included for alignment accuracy but excluded for F1-Score.

say which one is ultimately superior. When comparing the two, the End-to-End approach performs best on fully correct DRSs, but noticeably worse on the full data. However, since it cannot be expected that the pure DRS part will always be correctly parsed, the Combination approach seems the best choice overall. The performance differences between the full data and the subset of fully correct DRSs suggests that a correctly parsed DRS is important to producing a correct alignment. The performance of each approach on the task of pure DRS parsing is also relevant. The Alignment through Attention-approaches, which are trained for pure DRS parsing, outperform the other approaches in this regard. Also, while using bilinear attention instead of dot-product attention has a significant impact on alignment accuracy, this does not appear to be the case for the F1-Score on DRS parsing. This suggests that an attention scoring-function that does not use any learnable weights such as the dot-product attention may be worse at learning alignment between input and output, but might be just as helpful for overall model-performance as an attention scoring-function that does use learnable weights. Since the End-to-End- and Combination-approaches also had to generate alignment during training, it can be argued that they were trained on a more complex task than pure DRS parsing. However, we did not attempt an optimization of hyperparameters (instead using the hyperparameters of the underlying pure DRS parser of Noord et al.) for this new task. As such, the observed drop in F1-Score is unsurprising, and it might be possible to close the gap to the Alignment through Attention-approaches by finding a more fitting set of hyperparameters.

CHAPTER 6

Qualitative analysis

This chapter covers a manual inspection of the alignment generated by the End-to-End and Combination approaches described in Section 3.2 and Section 3.3 on a fraction of the PMB 3.0.0 dev-set data. The goal of this analysis is to gain insights in the kinds of errors that these systems produce and the issue they might have that lead to incorrect predictions. To that end, the output produced by the DRS alignment parsers that are found to have at least one incorrect alignment are extracted for inspection. This is done separately for the full data and the subset of the data consisting of DRSs where the pure DRS-part is fully correct, as described in Section 5.4.

6.1 Strategy for Manual Inspection

A DRS parser employing the End-to-End alignment method as well as one using the combined alignment method were used, and the produced DRS with at least one incorrect alignment were gathered. The full generated DRS and goldlabel-DRS (both with alignment) followed by a list of the incorrect (non-matching) alignments were collected. Each such sample was then inspected and a number of categories that describe some aspect of the error were assigned. No restrictions on the nature of an error-category were set, in order to be able to express a wide spectrum of situations. Figure 6.1 shows an example for the qualitative analysis of the sentence *Tom is addicted to heroin .* and its DRS as produced by the End-to-End alignment system before applying the post-processing.

It shows first the sentence, followed by Precision, Recall and F1-Score of the pure DRS. If all three values are 1.0, it means that the pure DRS was fully correctly parsed. This is followed by a listing of the alignment errors, meaning the predicted- and gold-alignments that were found to be not matching for matching DRS-clauses (see Section 5.4). In Figure 6.1, the model incorrectly predicted *sobs* instead of *heroin* as the alignment for the last two clauses, which is why the same alignment error is listed twice. This is then followed by the entire produced DRS with alignment, and lastly the entire gold-DRS

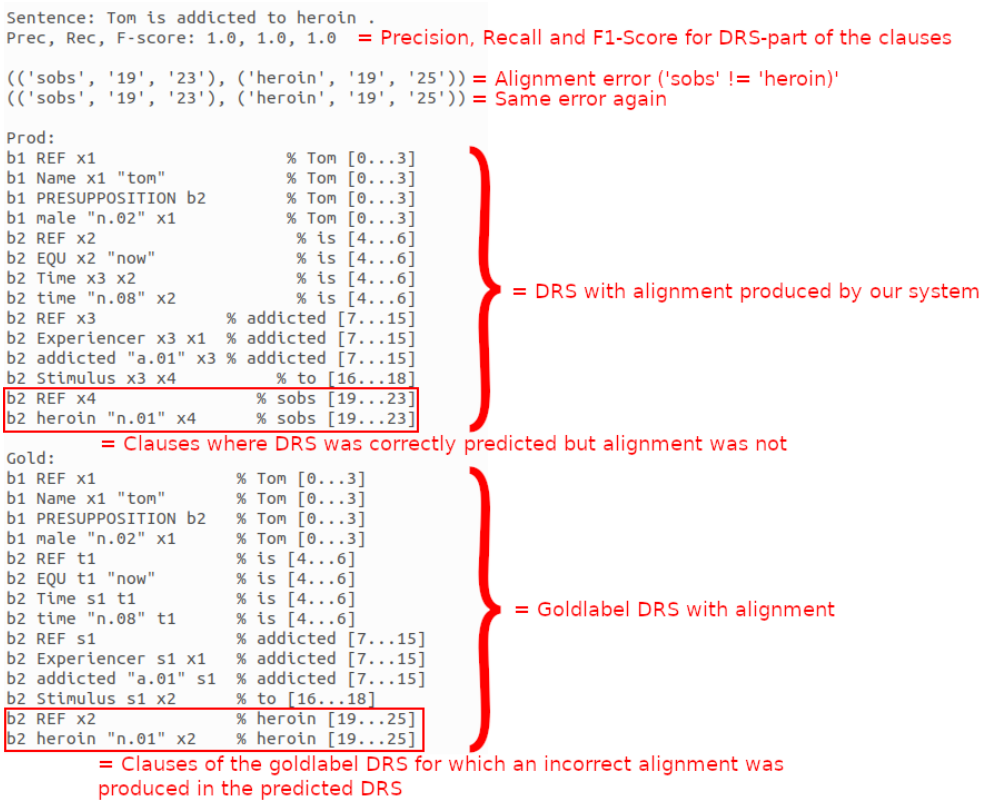


Figure 6.1: An example for how DRS were inspected during the qualitative analysis. The top shows the sentence *Tom is addicted to heroin .*, followed by Precision, Recall and F-Score which show that the pure DRS-part of this sample was fully correctly parsed. This is followed by the two found alignment errors (predicted alignment to the left, gold alignment to the right) followed by the predicted- and gold-DRSs with alignment. The example was produced by the End-to-End alignment system.

with alignment is shown. This example contains a lot of information on and around the prediction-error that the manual inspection tries to capture using the mentioned categories, for example:

- For one of the two errors, the corresponding DRS-clause represents the WordNet sense of *heroin*, so in this case the model correctly output the word *heroin* in the DRS-part but not in the alignment.
- The predicted token *sobs* has no obvious connection to the target alignment *heroin*, it is neither a similar looking word nor is it semantically similar.
- One of the errors happened on the very last clause of the DRS, which is a kind of error that was found to happen especially often for the Combination-approach, using bilinear / general attention.

- The model repeated the same mistake twice

Error-categories are not disjoint, and each sample can be assigned any number of them. A detailed description of those categories that were assigned frequently is provided below. This manual inspection was performed for End-to-End and Combination approaches, for a fraction of both the full data and the subset that contains only fully correct DRS, totaling 4 setups. For each run, between 40 and 45 samples were inspected, totaling 176 inspected samples over the four described scenarios. In particular, for the subset of fully correct DRS, 44 samples were inspected for the End-to-End approach and 45 for the Combination, and for all DRS those numbers were 45 and 42 respectively. While the two alignment-generation approaches use the same model, meaning that the pure DRS is identical for the End-to-End and Combination approaches, their alignment is different. Because of this, they make different alignment errors, which leads to different examples being collected for the manual inspection, though the inspected samples of the 4 setups can intersect.

End-to-End alignment The approach that generates alignment in an End-to-End fashion is able to freely generate any token of the vocabulary as the alignment. Because of this, post-processing is applied to choose a valid token from the input sequence as alignment (see Section 3.2). However, for the purposes of manual inspection, seeing the produced End-to-End alignment before applying this post-processing may provide additional insights. For this reason, the post-processed version of the output is only used when deciding when an alignment error has occurred, but the examination of that error is performed on the non-post-processed output.

6.2 Categories

This section details the most relevant alignment-error categories that were found and assigned during the manual inspection. However, this is not an extensive list, as some categories were found to be too infrequent and therefore discarded.

Semantically Similar - describes an error where the produced alignment is semantically close to the target alignment, yet no match could be determined. This category is only relevant for the End-to-End approach (before post-processing), as it can freely generate any token of the vocabulary as alignment and is not restricted to choosing a token from the input sequence. An example would be a target alignment *much*, but a predicted alignment *often* as can be seen in Figure 6.2a, or *toxic* and *poisonous* as can be seen in Figure 6.2b.

Hallucination - describes an error that can only be produced by End-to-End approach where the produced alignment is not any of the tokens from the input sequence and where there is also no apparent connection between the produced alignment and the target alignment. For example, Figure 6.1 shows a predicted alignment *sobs*, which has

```
Sentence: Tom dreamed of her often .
Prec, Rec, F-score: 1.0, 1.0, 1.0

(('much', '19', '23'), ('often', '19', '24'))

Prod:
b1 REF x1                % Tom [0...3]
b1 Name x1 "tom"         % Tom [0...3]
b1 PRESUPPOSITION b2     % Tom [0...3]
b1 male "n.02" x1        % Tom [0...3]
b2 REF x2             % dreamed [4...11]
b2 Experiencer x2 x1  % dreamed [4...11]
b2 TPR x3 "now"       % dreamed [4...11]
b2 dream "v.01" x2    % dreamed [4...11]
b2 REF x3             % dreamed [4...11]
b2 Time x2 x3         % dreamed [4...11]
b2 time "n.08" x3     % dreamed [4...11]
b2 Stimulus x2 x4        % of [12...14]
b3 REF x4               % her [15...18]
b3 PRESUPPOSITION b2    % her [15...18]
b3 female "n.02" x4     % her [15...18]
b2 Quantity x3 "+"     % much [19...23]

Gold:
b1 REF x1             % Tom [0...3]
b1 Name x1 "tom"      % Tom [0...3]
b1 PRESUPPOSITION b2  % Tom [0...3]
b1 male "n.02" x1     % Tom [0...3]
b2 REF e1             % dreamed [4...11]
b2 Experiencer e1 x1  % dreamed [4...11]
b2 TPR t1 "now"       % dreamed [4...11]
b2 dream "v.01" e1    % dreamed [4...11]
b2 REF t1             % dreamed [4...11] often [19...24]
b2 Time e1 t1         % dreamed [4...11] often [19...24]
b2 time "n.08" t1     % dreamed [4...11] often [19...24]
b2 Stimulus e1 x2     % of [12...14]
b3 REF x2             % her [15...18]
b3 PRESUPPOSITION b2  % her [15...18]
b3 female "n.02" x2   % her [15...18]
b2 Quantity t1 "+"    % often [19...24]
```

(a) Example where *much* was predicted instead of *often*, for the last clause shown.

```
Sentence: Is hexane toxic ?
Prec, Rec, F-score: 0.6, 0.6667, 0.6316

(('poisonous', '12', '21'), ('toxic', '10', '15'))
(('poisonous', '12', '21'), ('toxic', '10', '15'))

Prod:
b1 REF x1                      % Is [0...2]
b1 EQU x1 "now"                % Is [0...2]
b1 Time x3 x1                  % Is [0...2]
b1 time "n.08" x1              % Is [0...2]
b2 REF x2                  % Hebron [3...11]
b2 PRESUPPOSITION b1  % John~Dalton [3...11]
b2 company "n.01" x2  % John~Dalton [3...11]
b1 REF x3               % poisonous [12...21]
b1 Attribute x2 x3      % poisonous [12...21]
b1 poisonous "a.01" x3 % poisonous [12...21]

Gold:
b1 REF t1            % Is [0...2]
b1 EQU t1 "now"      % Is [0...2]
b1 Time s1 t1        % Is [0...2]
b1 time "n.08" t1    % Is [0...2]
b1 REF x1            % hexane [3...9]
b1 hexane "n.01" x1  % hexane [3...9]
b1 REF s1            % toxic [10...15]
b1 Attribute x1 s1   % toxic [10...15]
b1 toxic "a.01" s1   % toxic [10...15]
```

(b) Example where *poisonous* was predicted instead of *toxic* twice. There was no attempt to match the alignment for the second before last clause, as the pure DRS-part was incorrectly predicted (*x2* in the predicted DRS was incorrectly introduced in a different box *b2*.

[htbp]

```
Sentence: Ken was fined 7,000 yen for speeding .
Prec, Rec, F-score: 1.0, 1.0, 1.0

(('7', '14', '17'), ('7,000', '14', '19'))

Prod:
b1 REF x1                    % Ken [0...3]
b1 Name x1 "ken"             % Ken [0...3]
b1 PRESUPPOSITION b2         % Ken [0...3]
b1 male "n.02" x1            % Ken [0...3]
b2 REF x2                    % was [4...7]
b2 TPR x2 "now"              % was [4...7]
b2 Time x3 x2                % was [4...7]
b2 time "n.08" x2            % was [4...7]
b2 REF x3                 % fined [8...13]
b2 Asset x3 x4            % fined [8...13]
b2 Recipient x3 x1        % fined [8...13]
b2 fine "v.01" x3         % fined [8...13]
b2 REF x4                   % 7 [14...17]
b2 Quantity x4 "7,000"   % 7,000 [14...17]
b2 Unit x4 "yen"           % yen [18...21]
b2 measure "n.02" x4       % yen [18...21]
b2 Theme x3 x5             % for [22...25]
b2 REF x5              % speeding [26...34]
b2 speeding "n.01" x5 % speeding [26...34]

Gold:
b1 REF x1               % Ken [0...3]
b1 Name x1 "ken"        % Ken [0...3]
b1 PRESUPPOSITION b2    % Ken [0...3]
b1 male "n.02" x1       % Ken [0...3]
b2 REF t1               % was [4...7]
b2 TPR t1 "now"         % was [4...7]
b2 Time e1 t1           % was [4...7]
b2 time "n.08" t1       % was [4...7]
b2 REF e1               % fined [8...13]
b2 Asset e1 x2          % fined [8...13]
b2 Recipient e1 x1      % fined [8...13]
b2 fine "v.01" e1       % fined [8...13]
b2 REF x2               % 7,000 [14...19]
b2 Quantity x2 "7,000"  % 7,000 [14...19]
b2 Unit x2 "yen"        % yen [20...23]
b2 measure "n.02" x2    % yen [20...23]
b2 Theme e1 x3          % for [24...27]
b2 REF x3               % speeding [28...36]
b2 speeding "n.01" x3   % speeding [28...36]
```

(c) Example where 7 is predicted instead of 7,000. Interestingly, it is correctly aligned for the following clause.

```
Sentence: A hare raced with a tortoise .
Prec, Rec, F-score: 0.8333, 0.8333, 0.8333

(('chatted', '7', '14'), ('raced', '7', '12'))
(('chatted', '7', '14'), ('raced', '7', '12'))
(('chatted', '7', '14'), ('raced', '7', '12'))
(('chatted', '7', '14'), ('raced', '7', '12'))
(('chatted', '7', '14'), ('raced', '7', '12'))
(('chatted', '7', '14'), ('raced', '7', '12'))

Prod:
b1 REF x1                       % A [0...1]
b1 hare "n.01" x1            % hare [2...6]
b1 REF x2                 % chatted [7...14]
b1 REF x3                 % chatted [7...14]
b1 TPR x3 "now"           % chatted [7...14]
b1 Theme x2 x1            % chatted [7...14]
b1 Time x2 x3             % chatted [7...14]
b1 race "v.01" x2         % chatted [7...14]
b1 time "n.08" x3         % chatted [7...14]
b1 Instrument x2 x4         % with [15...19]
b1 REF x4                     % a [20...21]
b1 tortoise "n.01" x4 % tortoise [22...30]

Gold:
b1 REF x1               % A [0...1]
b1 hare "n.01" x1       % hare [2...6]
b1 REF e1               % raced [7...12]
b1 REF t1               % raced [7...12]
b1 TPR t1 "now"         % raced [7...12]
b1 Theme e1 x1          % raced [7...12]
b1 Time e1 t1           % raced [7...12]
b1 race "v.02" e1       % raced [7...12]
b1 time "n.08" t1       % raced [7...12]
b1 Co-Theme e1 x2       % with [13...17]
b1 REF x2               % a [18...19]
b1 tortoise "n.01" x2   % tortoise [20...28]
```

(d) Example where *chatted* was predicted instead of *raced*, for six clauses in the DRS.

Figure 6.2: Examples inspected during the qualitative analysis of alignment errors produced by End-to-End approach.

```
Sentence: What a genius he is !
Prec, Rec, F-score: 1.0, 1.0, 1.0

(('what', '0', '4'), ('a', '5', '6'))

Prod:
b1 REF x1                % what [0...4]
b1 REF x2              % genius [7...13]
b1 Role x1 x2          % genius [7...13]
b1 genius "n.01" x2    % genius [7...13]
b1 person "n.01" x1    % genius [7...13]
b2 REF x3                % he [14...16]
b2 PRESUPPOSITION b1     % he [14...16]
b2 male "n.02" x3        % he [14...16]
b1 REF x4                % is [17...19]
b1 REF x5                % is [17...19]
b1 Co-Theme x4 x1        % is [17...19]
b1 EQU x5 "now"          % is [17...19]
b1 Theme x4 x3           % is [17...19]
b1 Time x4 x5            % is [17...19]
b1 be "v.01" x4          % is [17...19]
b1 time "n.08" x5        % is [17...19]

Gold:
b2 REF x2              % a [5...6]
b2 REF x3              % genius [7...13]
b2 Role x2 x3          % genius [7...13]
b2 genius "n.01" x3    % genius [7...13]
b2 person "n.01" x2    % genius [7...13]
b1 REF x1              % he [14...16]
b1 PRESUPPOSITION b2   % he [14...16]
b1 male "n.02" x1      % he [14...16]
b2 REF e1              % is [17...19]
b2 REF t1              % is [17...19]
b2 Co-Theme e1 x2      % is [17...19]
b2 EQU t1 "now"        % is [17...19]
b2 Theme e1 x1         % is [17...19]
b2 Time e1 t1          % is [17...19]
b2 be "v.01" e1        % is [17...19]
b2 time "n.08" t1      % is [17...19]
```

(a) *what* was predicted instead of *a*, for the first last clause shown in the example.

```
Sentence: Mr. Ford is all right now .
Prec, Rec, F-score: 1.0, 1.0, 1.0

(('now', '22', '25'), ('all~right', '12', '21'))

Prod:
b1 REF x1                 % mr. [0...3]
b1 REF x2                 % mr. [0...3]
b1 PRESUPPOSITION b2      % mr. [0...3]
b1 Title x1 x2            % mr. [0...3]
b1 mr "n.01" x2           % mr. [0...3]
b1 Name x1 "ford"        % ford [4...8]
b1 male "n.02" x1        % ford [4...8]
b2 REF x3                  % is [9...11]
b2 EQU x3 "now"            % is [9...11]
b2 Time x4 x3              % is [9...11]
b2 time "n.08" x3          % is [9...11]
b2 REF x4                % all [12...15]
b2 Attribute x1 x4       % all [12...15]
b2 all_right "a.01" x4 % now [22...25]

Gold:
b1 REF x1                 % Mr. [0...3]
b1 REF x2                 % Mr. [0...3]
b1 PRESUPPOSITION b2      % Mr. [0...3]
b1 Title x1 x2            % Mr. [0...3]
b1 mr "n.01" x2           % Mr. [0...3]
b1 Name x1 "ford"         % Ford [4...8]
b1 male "n.02" x1         % Ford [4...8]
b2 REF t1                 % is [9...11] now [22...25]
b2 EQU t1 "now"           % is [9...11] now [22...25]
b2 Time s1 t1             % is [9...11] now [22...25]
b2 time "n.08" t1         % is [9...11] now [22...25]
b2 REF s1                 % all~right [12...21]
b2 Attribute x1 s1        % all~right [12...21]
b2 all_right "a.01" s1    % all~right [12...21]
```

(b) *now* was predicted instead of *all*, *right*, or *all~right*, since for multitoken alignments we require our systems to only produce either of the tokens connected by '~'.

```
Sentence: I chopped a tree down .
Prec, Rec, F-score: 1.0, 1.0, 1.0

(('down', '17', '21'), ('tree', '12', '16'))

Prod:
b1 REF x1                % chopped [2...9]
b1 REF x2                % chopped [2...9]
b1 Agent x1 "speaker"    % chopped [2...9]
b1 Patient x1 x3         % chopped [2...9]
b1 TPR x2 "now"          % chopped [2...9]
b1 Time x1 x2            % chopped [2...9]
b1 chop_down "v.01" x1 % chopped [2...9]
b1 time "n.08" x2        % chopped [2...9]
b1 REF x3                   % a [10...11]
b1 tree "n.01" x3         % down [17...21]

Gold:
b1 REF e1                % chopped [2...9]
b1 REF t1                % chopped [2...9]
b1 Agent e1 "speaker"    % chopped [2...9]
b1 Patient e1 x1         % chopped [2...9]
b1 TPR t1 "now"          % chopped [2...9]
b1 Time e1 t1            % chopped [2...9]
b1 chop_down "v.01" e1   % chopped [2...9]
b1 time "n.08" t1        % chopped [2...9]
b1 REF x1                % a [10...11]
b1 tree "n.01" x1        % tree [12...16]
```

(c) Example where *down* was predicted instead of *tree*, for the laust clause.

```
Sentence: It 's October the third .
Prec, Rec, F-score: 0.3077, 0.8, 0.4445

(("'s", '3', '5'), ('It', '0', '2'))
(('third', '18', '23'), ('October', '5', '12'))

Prod:
b1 REF x1                      % it [0...2]
b1 PRESUPPOSITION b2           % it [0...2]
b1 entity "n.01" x1            % it [0...2]
b2 REF x2                      % 's [3...5]
b2 Co-Theme x2 x3              % 's [3...5]
b2 EQU x3 "now"                % 's [3...5]
b2 Theme x2 x1                 % 's [3...5]
b2 Time x2 x3                  % 's [3...5]
b2 be "v.02" x2                % 's [3...5]
b2 REF x3                      % 's [3...5]
b2 MonthOfYear x3 "10" % october [6...13]
b2 DayOfMonth x3 "20"    % third [18...23]
b2 time "n.08" x3        % third [18...23]

Gold:
b1 REF t1                % It [0...2]
b1 EQU t1 "now"          % 's [2...4]
b1 MonthOfYear t1 "10"   % October [5...12]
b1 time "n.08" t1        % October [5...12]
b1 DayOfMonth t1 "03"    % third [17...22]
```

(d) Example of a "Bad DRS" where the predicted DRS looks very different from the goldlabel.

Figure 6.3: Examples inspected during the qualitative analysis of alignment errors produced by End-to-End approach.

little in common with the target alignment *heroin*. Another example is shown in Figure 6.2d.

Correct Wordsense - describes an alignment-error for a Wordnet sense DRS-clause where the alignment was incorrectly predicted but a token equal or very similar to the target alignment was predicted in the DRS-clause as the word sense. An example is shown in Figure 6.2d, where for the DRS-clause *b1 race "v.01" x1* in the predicted DRS, the word *race* is actually produced as part of the WordNet sense, but the system still failed to produce *raced* as the alignment. Another example is shown in Figure 6.3c.

Numeral - describes an error where the alignment consists of a number that was incorrectly predicted. An example can be seen in Figure 6.2c.

Repeated Mistake - describes a situation where the exact same alignment error is made multiple times within the same DRS. Examples can be seen in Figures 6.2d and 6.2b. For this category, the number of errors caused by repetition was also tracked.

Out of Vocabulary - describes an error that can only occur when using the End-to-End approach, where the out-of-vocabulary token is predicted as alignment.

Bad DRS - describes a situation where the DRS produced by the parser is significantly different from the gold-DRS, to the point where it is not possible to be sure for which DRS-clause an alignment error has occurred. Because Counter tries to find the maximum number of matching clauses under a variable mapping (see Section 5.1), two very differently looking DRS can still have a few clauses that will be counted as matching, and as such their alignment is compared when evaluating alignment. In such a scenario, the "Bad DRS" error category is assigned, and no further error-categories are allowed. An example is seen in Figure 6.3d.

Multitoken Alignment - describes an error where the target alignment consists of multiple tokens that are considered to be belonging together as per the tokenization performed for the data in the PMB, indicated by a '~' character, e.g *New~York* or *10~am*. When evaluating the produced alignment to such a gold-alignment, we check for inclusion instead of equality (see Section 5.3). Despite this, errors on such alignments were found relatively frequently when considering that multitoken alignment make up only a small fraction of all the alignments in the PMB. An example can be seen in Figure 6.3b.

One Off (Left) & One Off (Right) - describes an error where the predicted token is the one preceding or succeeding the actual alignment within the input sequence. An example of a One Off (Left) error is shown in Figure 6.3a and two One Off (Right) errors are shown in Figures 6.3c and 6.3b. Due to the way that the Combination approach generates alignment from an attention-mechanism, this error is very relevant for that approach.

First Clause - describes an alignment error on the very first clause of the DRS. While a DRS is, in theory, an unordered set of clauses, this category is relevant for approaches that generate alignment from the attention-mechanism. AllenNLP (Gardner et al., 2018), the framework our models are implemented in, adds a special start- and end-token to

each sequence. The Combination approach ensures that such a token is never chosen as alignment, but it may be the case that for the first clause, some attention-weight is given to this technical start-token. An example is shown in Figure 6.3a.

Last Clause - describes an alignment error on the very last clause of the DRS, similar to the First Clause-error. In addition to the technical end-of-sentence-token used by the NLP-framework, there is also the end-of-sentence-punctuation which may be problematic for alignment-systems based on attention, see Figures 6.2a, 6.3c and 6.3b.

6.3 Results & Discussion

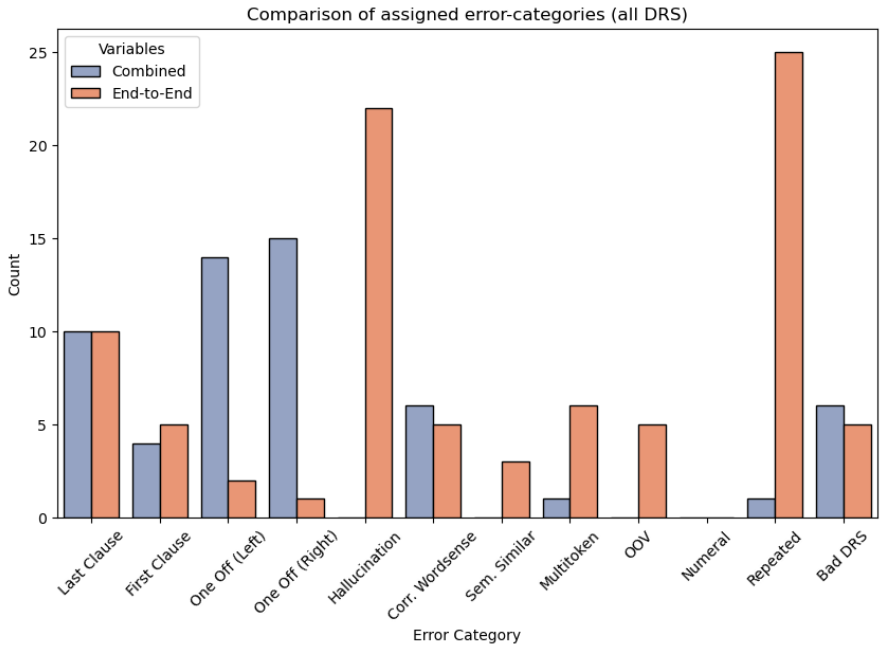


Figure 6.4: Count of assigned error-categories for the Combination and End-to-End approaches on a fraction of the PMB 3.0.0 dev-set. In total, 44 samples were inspected for the End-to-End approach and 45 for the Combination approach.

The strategy with which the two compared approaches generate the alignment is fundamentally different. The End-to-End approach generates both the DRS-clause and its predicted alignment as part of the output sequence. An element from the input sequence is then chosen based on similarity to that predicted alignment in post-processing. The Combination approach directly selects one element from the input sequence and chooses it as the alignment for a produced DRS-clause, replacing the alignment generated in End-to-End fashion. These differences in method are reflected in the errors found during the manual inspection. Figure 6.4 shows counts of the assigned error-categories for both correct and partially incorrect DRS, while Figure 6.5 shows them only for fully correct DRS.

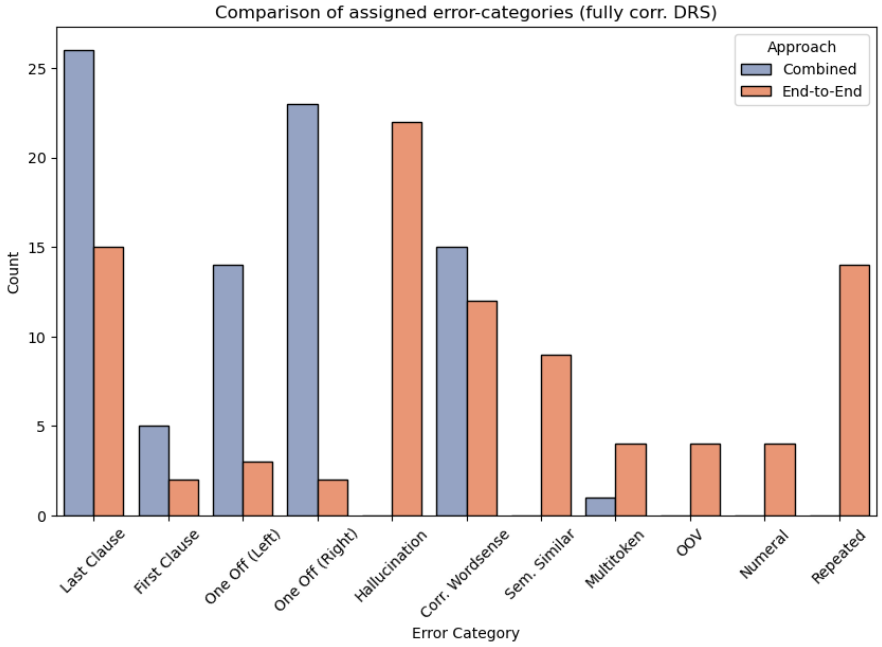


Figure 6.5: Count of assigned error-categories for the Combination and End-to-End approaches on a fraction of the PMB 3.0.0 dev-set on only fully correct DRS. In total, 45 samples were inspected for the End-to-End approach and 42 for the Combination approach.

Repeated Mistake - For the End-to-End system, the "Repeated Mistake"-Category was assigned very frequently in both scenarios, with it being assigned to more than half of the inspected DRS which are allowed to contain parsing errors. This means that whatever the actual alignment error was, the system then repeated it multiple times within that same DRS. The average number of repetition-errors for all DRS that have at least two repetition-errors was found to be 3.88 for all DRS and 3.14 for only fully correct DRS. The combined system on the other hand only made a single repetition-error on a partially correct DRS, and none at all on the fully correct DRS. This may be a big contributor to the observed performance difference between the two approaches on all DRS, as discussed in Chapter 5.

First Clause, Last Clause, Correct Wordsense When it comes to the very first and last clauses in a DRS, it can be seen that both systems produce errors on the first clause rather rarely, while producing errors in the last clause frequently. For the subset of fully correct DRS, this is clearly happening more frequently for the combined system, with 26 "Last Clause"-errors for the combined system and 15 for the End-to-End system, while the counts are very similar between the two when considering all DRS. This suggests that

- the combined system struggles with the very last clause of a DRS. The last DRS-clause is often associated with the next-to-last element in the input sequence, since

the very last element in the input sequence is usually end-of-sentence-punctuation and does therefore not have any corresponding clauses. Furthermore, the NLP-framework our models are implemented in appends a technical end-of-sentence token, which is never predicted but that can receive an attention-score greater than zero from the cross-attention-mechanism. While further investigation would be required to confirm this, the Combination approach's proneness to produce significantly more "Last Clause"-errors on the fully correct DRSs is a potential indicator for this issue.

- While DRS are formally unordered sets, our model treats DRS parsing as a sequence generation task, which is why the ordering of the DRS-clauses in the PMB can matter. During the manual inspection, it was observed that a large number of DRS have the WordNet sense of some verb or noun as their very last clause. Examples of this can be seen in Figures 6.1, 6.2a, 6.2c, 6.2d and 6.3c. These kinds of clauses may be particularly difficult to align, leading to the high number of "Last Clause"-errors observed for both systems.

A category heavily related to such DRS, and assigned similarly often to the output of both approaches, is "Correct Wordsense", i.e. an alignment error on a DRS clause that correctly represents a WordNet sense of the alignment. This is interesting, as it would seem that the alignment of a DRS-clause should be trivial if the DRS-clause more or less already contains exactly that alignment. However, both systems failed to do so, resulting in about 5 observed errors on all DRS and between 12 and 15 for the fully correct DRS. Indeed, the two error categories "Last Clause" and "Correct Wordsense" seem to be connected. For these two categories, the inspected examples for fully correct DRS and all DRS were concatenated and it was counted how often the two categories occur together (on the same DRS) for the two approaches. On the combined system, it was found that out of 36 "Last Clause"-errors, 21 were also "Correct Wordsense"-errors. For the End-to-End approach, these numbers were 25 and 16 respectively. This suggests that both models may benefit from some mechanism that allows them to make use of the WordNet sense predicted in the DRS-clause in cases where the WordNet sense seems plausible while the alignment does not.

Some of the proposed error-categories are naturally only really relevant for one of the two alignment-generation methods, and either infrequent or impossible to observe with the other. However, while they do not allow for comparison between the two approaches, they still provide insight in the two systems individually.
One Off (Left), One Off (Right) - The "One Off"-error is the most frequent error-type found for the combined system (when counting left and right together), making up 37 out of 45 errors for the fully correct DRSs and 29 out of 42 for all DRS. Therefore, when the Combination approach makes an alignment error, it will in many cases be just one of the neighbors of the actual alignment. While "One Off (Right)"-errors occurred noticeably more frequently for the fully correct DRS, the frequency is about equal with that of "One Off (Left)"-errors for all DRS. For the End-to-End system, this error is very infrequent,

as it does not produce any token from the input sequence in most cases, thus making a "One Off"-error very rare.

Hallucination - For the End-to-End approach, the largest kind of error observed besides repeated mistakes was of the "Hallucination"-category, meaning the alignment produced by the system (prior to finding corresponding an element from the input sequence in post-processing) is very different from the target-alignment (and not any element from the input sequence), and it was not possible to come up with an explanation of why the system might have produced such an output. Using the Combination approach, such an error is impossible to produce, as it will always choose some token from the input sequence as alignment.

Others Lastly, error-categories "Multitoken" and "Numeral" have been observed mainly for the End-to-End approach. While there are only a few of those errors among the inspected examples, these mistakes can only appear on a fraction of the dataset, e.g. only about 3% of samples in the PMB contain a multitoken alignment (see Section 2.2). Taking this into account, the two error-categories can actually be considered to occur relatively frequent.

CHAPTER 7

Conclusion

This thesis expands upon an existing neural seq2seq DRS parser by producing an alignment between input sequence (natural language) and output sequence (DRS) that is found in the training data but ignored by the original DRS parser. This is done by both explicitly making the alignment a part of the output sequence that the parser needs to produce, as well using a cross-attention-mechanism and extracting the alignment from it based on the attention-scores between elements in the input- and output-sequence.

The contributions of this work are formulated as responses to the Research Questions asked in Chapter 1:

1. **How can DRS that include an alignment to the input sequence be generated and what is the impact of the choice of alignment generation method on the alignment accuracy? How does it affect performance on the pure DRS generation task?**
   We propose three methods for generating the desired alignment:

   - Alignment through Attention, where a pure DRS parser is trained and the alignment extracted from a cross-attention-mechanism.
   - End-to-End, where the model is trained to produce the alignment alongside DRS is an End-to-End fashion.
   - The Combination approach that is trained as in the End-to-End approach, but the alignment in the output sequence is then overwritten by an alignment extracted from the cross-attention-mechanism.

   The Alignment through Attention approach achieves the lowest alignment accuracy, however, since it is trained as a DRS-only parser, it achieves the highest F-Score for the task of pure DRS parsing among the three approaches. The Combination

performs best for the task of alignment generation, achieving an almost perfect alignment accuracy of more than 98%, however, since it is trained on generating DRS and alignment (and no parameter-tuning was performed to account for this change in scenario), its performance on the task of pure DRS parsing is slightly decreased compared to the Alignment through Attention approach. However, this is a small decrease of in F1-Score (from 88.53 to 85.74) for DRS parsing, while an alignment accuracy increases from 82.15 to 98.46.
For the approaches that use attention to generate alignment, two types of attention, dot-product attention and general/bilinear attention were used, and general/bilinear attention was found to be better able to learn the desired alignment.

2. **What are the characteristics of alignment errors? Can any shortcomings in the proposed systems be identified based on the errors they produce?**
A manual inspection of alignment errors has shown that the End-to-End approach can produce alignments that are very different from any of the tokens of the input sequence (including the target alignment). It also makes repeated mistakes, meaning that the same incorrect alignment is output many times over in the same DRS.
The Combination approach produces almost no repeated errors, and many of its incorrectly predicted alignments are "One Off"-errors, meaning that a neighboring word of the correct alignment is predicted.
Furthermore, it was found that many alignment errors of both approaches are made in the very last clause in a DRS, as this clause often constitutes the WordNet sense of some noun or verb that may be particularly difficult to align. Interestingly, in such cases, the DRS-clause has been observed to frequently contain the alignment (or a phrase very similar to the alignment) as part of the WordNet sense, meaning that the alignment-generation approaches may benefit from a mechanism that allows them to make use of this WordNet sense in cases where the initial alignment seems implausible.

List of Figures

List of Tables

Bibliography

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Lasha Abzianidze, Rik van Noord, Hessel Haagsma, and Johan Bos. 2019. The first shared task on discourse representation structure parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden. Association for Computational Linguistics.

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. A platform for collaborative semantic annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 92–96, Avignon, France. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646, Dublin, Ireland. Association for Computational Linguistics.

Patrick Blackburn and Johan Bos. 2005. Representation and inference for natural language - a first course in computational semantics. In *CSLI Studies in Computational Linguistics*.

Claire Bonial, William Corvey, Martha Palmer, Volha V Petukhova, and Harry Bunt. 2011. A hierarchical unification of lirics and verbnet semantic roles. In *2011 IEEE Fifth International Conference on Semantic Computing*, pages 483–489. IEEE.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Johan Bos. 2015. Open-domain semantic parsing with boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304, Vilnius, Lithuania. Linköping University Electronic Press, Sweden.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J. Venhuizen, and Johannes Bjerva. 2017. *The Groningen Meaning Bank*, pages 463–496. Springer Netherlands, Dordrecht.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kilian Evang. 2019. Transition-based DRS parsing using stack-LSTMs. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden. Association for Computational Linguistics.

Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2769–2778, Hong Kong, China. Association for Computational Linguistics.

Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. 2016. Aligning where to see and what to tell: Image captioning with region-based attention and scene-specific contexts. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2321–2334.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A

deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4453–4462, Hong Kong, China. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Heikki Hyyrö. 2001. Explaining and extending the bit-parallel approximate string matching algorithm of myers. Technical report, Citeseer.

Mark Johnson and Ewan Klein. 1986. Discourse, anaphora and parsing. In *Coling 1986 Volume 1: The 11th International Conference on Computational Linguistics*.

Hans Kamp, Josef van Genabith, and Uwe Reyle. 2011. Discourse representation theory. In *Handbook of philosophical logic*, pages 125–394. Springer.

Phong Le and Willem Zuidema. 2012a. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012*, pages 1535–1552, Mumbai, India. The COLING 2012 Organizing Committee.

Phong Le and Willem Zuidema. 2012b. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012*, pages 1535–1552, Mumbai, India. The COLING 2012 Organizing Committee.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics – doklady*, 10(8):707–710.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2018. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2019a. Discourse representation parsing for sentences and documents. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6248–6262, Florence, Italy. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, and Mirella Lapata. 2019b. Discourse representation structure parsing with recurrent neural networks and the transformer model. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Reinhard Muskens. 1996. Combining montague semantics and discourse representation. *Linguistics and philosophy*, pages 143–186.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Fernando Pereira and Stuart M. Shieber. 1987. Prolog and natural-language analysis.

Wessel Poelman, Rik van Noord, and Johan Bos. 2022. Transparent semantic parsing with Universal Dependencies using graph transformations. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4186–4192, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71:599–607.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Mark Steedman. 2001. *The syntactic process*. MIT press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Rob A Van der Sandt. 1992. Presupposition projection as anaphora resolution. *Journal of semantics*, 9(4):333–377.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Rik van Noord, Antonio Toral, and Johan Bos. 2019. Linguistic information in neural semantic parsing with multiple encoders. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 24–31, Gothenburg, Sweden. Association for Computational Linguistics.

Rik van Noord, Antonio Toral, and Johan Bos. 2020. Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4587–4603, Online. Association for Computational Linguistics.

Noortje J Venhuizen, Johan Bos, Petra Hendriks, and Harm Brouwer. 2018. Discourse Semantics with Information Structure. *Journal of Semantics*, 35(1):127–169.

Hajime Wada and Nicholas Asher. 1986. BUILDRS: An implementation of DR theory and LFG. In *Coling 1986 Volume 1: The 11th International Conference on Computational Linguistics*.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.