University of Groningen

Character-based Neural Semantic Parsing

van Noord, Rik

DOI:
[10.33612/diss.169308968](https://doi.org/10.33612/diss.169308968)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2021

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):
van Noord, R. (2021). *Character-based Neural Semantic Parsing*. [Thesis fully internal (DIV), University of Groningen]. University of Groningen. <https://doi.org/10.33612/diss.169308968>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Character-based Neural Semantic Parsing

Rik van Noord

university of
 groningen

The work in this thesis has been carried out under the auspices of the Center for Language and Cognition Groningen (CLCG) of the Faculty of Arts of the University of Groningen.

Groningen Dissertations in Linguistics 193

© 2021, Rik van Noord

Document prepared with $\text{\LaTeX} 2_{\epsilon}$ and typeset by pdf $\text{\TeX}$
(Droid Serif and Lato fonts)

Cover design: Anna de Koster

rijksuniversiteit
 groningen

Character-based Neural Semantic Parsing

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus prof. dr. C. Wijmenga
en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

donderdag 31 mei 2021 om 16:15 uur

door

Rik Iwan Kay van Noord

geboren op 28 oktober 1991
te Den Bosch, Nederland

Promotor

Prof. dr. ing. Johan Bos

Copromotor

Dr. Antonio Toral Ruiz

Beoordelingscommissie

Prof. dr. Michael Biehl

Prof. dr. Anette Frank

Prof. dr. Mirella Lapata

Acknowledgements

I quite enjoyed my time as a PhD-student. This is largely due to all the people I spent time with over the past 4 years. I feel like I owe all of you a sincere thank you, so here goes.

First, and most importantly, I'd like to thank Johan for hiring me and guiding me through the project. You gave me a lot of freedom, which I appreciated, but also gently pushed me back on track if needed. Let the record show that some of the more creative ideas in this thesis were yours. I really enjoyed working together, but also have great memories of the soccer matches and parties you hosted (despite your kids always beating me at chess).

Antonio, asking you to be my co-supervisor was probably one of the best decisions I made during the PhD. Your knowledge about neural networks, eye for detail and pressing for experimental rigour made this a much better thesis. Above all, you were simply very pleasant to work with. Thanks a lot!

Lasha, though not as an official supervisor, you helped me a lot during these 4 years (even co-authoring two of the papers) and I appreciate that. There is no one else out there who can be so clever, funny and confusing at the same time. I will surely always remember the "compensation cake".

Of course, thanks to the reading committee, Michael Biehl, Anette Frank and Mirella Lapata, for reading and accepting my thesis. You provided me with a lot of useful feedback, which definitely improved the thesis.

As a PhD-student, you spend a lot of time with other PhD-students. I was lucky enough to get to know quite a bunch of friendly, nice, smart and funny ones:¹ Ahmet, Anna Pot, Chunliu, Dieke, Gosse, Hessel, Huiyuan, Johannes, Kilian, Lukas, Martijn, Masha, Prajit, Rob, Stephan and Steven. I also should

¹Not all adjectives apply to all people.

not forget the new PhD-students whom I've only met while playing online pictionary: Raoul, Silvia, Teja and Wietse. So far, I like you as well.

Generally, the CL department in Groningen was (and is) a great place to work, so I'd like to thank all my other colleagues as well: Andreas, Arianna, Barbara, Daniël, Gertjan, Gosse, Gregory, Leonie, Malvina, Martijn, and Tommaso. Sorry for often (always?) shifting the lunch conversation towards football and for all the complaints about the soup prices.²

Some colleagues even turned into (dare I say) friends by playing bi-weekly pubquizzes: Anna Pot, Dieke, Hessel, Pauline and Steven. It was an honour to captain the team and I hope we will continue playing.

In no particular order I would also like to thank my "normal" friends: Haye & Melissa, Edwin & Marisa, Maikel & MC, Joost & Amelie, Peter & Dineke & Willemijn, Marc & Mirjam, Jorrit & Floor and everyone from Vijfje 10 for the great memories. No doubt I will still see all of you in the future.

En natuurlijk een bedankje voor de neven, nichten, ooms, tantes en oma van de Wijgergangen & Van Noorden en de schoonfamilie van de De Kusters. Hopelijk kunnen we er later nog een leuk feestje van maken. Uiteraard speciale aandacht voor ons gezin: papa, mama, Ida & Mart en Kris & Rémi. Alle spelletjes, etentjes en weekendjes weg waren altijd erg gezellig.

Als laatste een enorm bedankje voor mijn verloofde Anna. Als je 6 maanden lang in een kamer van 8m² allebei een PhD kan afmaken, terwijl je ook nog je huis niet uit mag vanwege een pandemie, dan kan je alles aan. Wat er ook gebeurt, wij redden het wel.

²Seriously though, a 50% increase in price, who does that?

Contents

Contents	vii
1 Introduction	1
1.1 Semantic Parsing	2
1.2 Chapter Guide	3
1.3 Publications	6
I Background	9
2 Computational Semantics	11
2.1 Semantic Formalisms	11
2.1.1 Abstract Meaning Representations	11
2.1.2 Discourse Representation Structures	15
2.1.3 Comparing AMR and DRS	20
2.2 Semantic Parsing	22
2.2.1 Rule-based Approaches	22
2.2.2 Closed Domain Semantic Parsing	23
2.2.3 Open Domain Semantic Parsing	24
3 Sequence-to-sequence Architecture	29
3.1 Neural Networks	29
3.2 Recurrent Neural Networks	32
3.3 Sequence-to-sequence Models	35
3.4 Hyper-parameters	41
3.5 Characters versus Words	42
II Abstract Meaning Representation	47
4 Neural AMR parsing	49
4.1 Introduction	50
4.2 Method and Data	51
4.2.1 Abstract Meaning Representations	51
4.2.2 The Basic Translation Model	53
4.2.3 Postprocessing and Restoring Information	55
4.2.4 Baseline Results	57
4.3 Improving the Basic Translation Model	58

4.3.1	AMR Re-ordering	58
4.3.2	Introducing Super Characters	59
4.3.3	Adding Part-of-Speech Information	60
4.3.4	Adding Silver Standard Data	61
4.3.5	Optimizing Training	63
4.4	Results and Discussion	64
4.5	Conclusions and Future Work	67
5	Dealing with Coreference in Neural AMR Parsing	69
5.1	Introduction	70
5.2	Abstract Meaning Representations	71
5.3	Method	72
5.3.1	Variable-free AMRs	72
5.3.2	Neural Model	77
5.3.3	Experiments	77
5.4	Results and Discussion	78
5.4.1	Main Results	78
5.4.2	Detailed Analysis	79
5.5	Conclusions	80
III	Discourse Representation Structures	81
6	An Evaluation Method for DRS Parsing	83
6.1	Introduction	84
6.2	Scoped Meaning Representations	85
6.2.1	Discourse Representation Structures	86
6.2.2	Comparing DRSs to AMRs	87
6.3	The Parallel Meaning Bank	89
6.4	Matching Scoped Representations	91
6.4.1	Evaluation by Matching	91
6.4.2	Evaluating the Matching Procedure	93
6.5	Counter in Action	97
6.5.1	Semantic Parsing	97
6.5.2	Comparing Translations	99
6.6	Conclusions and Future Work	101

7	Neural DRS Parsing	103
7.1	Introduction	104
7.2	Discourse Representation Structures	105
7.3	Method	108
7.3.1	Annotated Data	108
7.3.2	Clausal Form Checker	110
7.3.3	Evaluation	111
7.3.4	Neural Architecture	112
7.4	Experiments with Data Representations	112
7.4.1	Between Characters and Words	113
7.4.2	Tokenization	115
7.4.3	Representing Variables	116
7.4.4	Casing	118
7.5	Experiments with Additional Data	118
7.6	Discussion	122
7.6.1	Comparison	122
7.6.2	Analysis	123
7.6.3	Manual Inspection	125
7.7	Conclusions and Future Work	126
8	Linguistic Information in Neural DRS Parsing	129
8.1	Introduction	130
8.2	Data and Methodology	131
8.2.1	Discourse Representation Structures	131
8.2.2	Representing Input and Output	131
8.2.3	Neural Architecture	133
8.2.4	Evaluation Procedure	136
8.3	Results	136
8.4	Shared Task Participation	139
8.4.1	Learning Curves	139
8.4.2	Improvement Methods	140
8.4.3	Competition Results	141
8.4.4	Error Analysis	143
8.5	Conclusions	143

IV Putting Characters into Context	145
9 Combining Characters with Contextual Embeddings	147
9.1 Introduction	148
9.2 Background	149
9.2.1 Transformers & BERT	149
9.2.2 Character-level Models	151
9.2.3 Discourse Representation Structures	151
9.2.4 Abstract Meaning Representations	153
9.3 Method	155
9.3.1 Neural Architecture	155
9.3.2 Representations	160
9.3.3 Data and Evaluation	162
9.4 Results	165
9.4.1 DRS Results	165
9.4.2 AMR Results	171
9.5 Analysis	175
9.5.1 Semantic Tag Analysis	175
9.5.2 Sentence Length Analysis	177
9.5.3 Discussion	180
9.6 Conclusions	181
V Conclusions	183
10 Conclusions	185
Appendices	189
A Postprocessing Improvement Methods	191
B Detailed Scores	193
C Sentence Analysis	205
Bibliography	209
Samenvatting	235

CHAPTER 1

Introduction

Humans and computers do not speak the same language. A lot of day-to-day tasks would be vastly more efficient if we could communicate with computers using natural language instead of relying on an interface. It is necessary, then, that the computer does not see a sentence as a collection of individual words, but instead can understand the deeper, compositional *meaning* of the sentence. A way to tackle this problem is to automatically assign a formal, structured *meaning representation* to each sentence, which are easy for computers to interpret. Say, for instance, that we want to produce such a meaning representation for the following example:

(1) Jack killed Tom.

We could say that there was a *killing* event, with Jack as the killer, and Tom as the person that was killed. Another formulation is that there are two people in this sentence (p1 and p2), one of which is Jack and one of which is Tom, and that Jack killed Tom. These two representations are shown in Figure 1.1.

Figure 1.1: Two example representations for the sentence *Jack killed Tom*.

1.1 Semantic Parsing

The method of going from text to meaning representation is better known as *semantic parsing*, while the flavor of meaning representation is called the *semantic formalism*. We can devise a set of rules to automatically create these representations by looking at the syntax: the subject is always the killer, and the direct object is the person that gets killed. We could easily extend this set of rules to handle other verbs, such as *hit*, *tickled*, *kissed*, *liked* and so on. Moreover, we could add rules that can handle transitive constructions (*Jack was killed by Tom*) and perhaps even automatically recognize negation (*Jack did not kill Tom*). However, now consider the following examples:

(2) Jack killed Tom with his gun.

(3) Jack killed Tom with his joke.

We could extend our set of syntactic rules to account for Example (2), but Example (3) already shows that this will not be sufficient. Clearly, no actual killing event took place here, as *killed* was used in a non-literal sense, even though the syntax is the same. We have to somehow account for the *meaning* of both the individual words and how they are combined in a sentence.

There is a large body of work that tried to add such a meaning component to existing sets of syntactic rules, usually called *grammars*. However, these grammar extensions had to be manually created, which is an expensive and time-consuming task. Moreover, they usually did not work well outside the domain they were created for. This is not surprising, considering the immense depth of natural language. We would rather have a method that automatically *learns* to produce the meaning representations, instead of being reliant on manually constructed rule-based components.

There are indeed a number of studies that attempt this (and we will review them in Section 2.2), which have gotten impressive results. However, they are usually still reliant on linguistic resources, such as syntactic analyses or predefined lexicons. We would prefer a method that can construct a meaning representation based on only the input text, without being dependent on any external resources. In this thesis, we will be working with *neural networks*, specifically *sequence-to-sequence* models (Sutskever et al.,

2014). These models can learn to transform a sequence of inputs (letters or words) to a sequence of outputs (parts of the meaning representations) and have achieved impressive performance on a number of Natural Language Processing (NLP) tasks (Vinyals et al., 2015; Rush et al., 2015; Xiao et al., 2016).

We will apply this model on two semantic formalisms for which open domain annotated corpora are available: the graph-based Abstract Meaning Representation (Banarescu et al., 2013) and the more expressive scoped meaning representations of Discourse Representation Structures (Kamp and Reyle, 1993). In particular, we are interested in finding out what the best representation of the input texts is for this model: individual letters (characters) or words? Word-level models are the more intuitive choice, but character-level models have a number of advantages, as they are, for instance, better equipped to deal with unknown words and spelling errors.

We have to solve a number of problems along the way. First, we propose novel methods to deal with the *variables* in the meaning representations. Variables, such as p1 and p2 in Figure 1.1, are commonly used to accurately model semantic phenomena, including coreference, control constructions and scope. As the variable names themselves are arbitrary and meaningless, they present a challenge to our sequence-to-sequence model. Second, while semantic parsing data sets are generally small, these models are known to be data-hungry. To combat this, we use automatically generated (*silver standard*) data to increase performance. Third, we investigate how much we can benefit from injecting linguistic knowledge (such as syntactic analyses) into the system. In the last content chapter, we discuss the rise of *contextual language models* (Devlin et al., 2019) that happened throughout the course of this thesis, and try to determine if character-level representations are still useful in this new era of NLP.

1.2 Chapter Guide

This thesis is divided in five parts and contains ten chapters. The research in this thesis centers around developing open domain neural semantic parsers that can produce accurate (formal) meaning representations. We aim to answer the following seven research questions:

- RQ1** Can neural sequence-to-sequence models be used to produce accurate meaning representations?
- RQ2** How can we best represent the input texts for these models: characters or words?
- RQ3** To what extent can we improve performance by employing silver standard data?
- RQ4** How can we best deal with the variables in the meaning representations?
- RQ5** AMR has a well-established evaluation method. Can we construct a similar method to evaluate Discourse Representation Structures?
- RQ6** Can we improve neural semantic parsers by injecting linguistic knowledge?
- RQ7** Can we combine character-level representations with representations from a contextual language model to improve neural semantic parsing? If so, what is the best method of combining them?

Part I - Background

Part I sets the stage for the rest of the thesis. It contains an extensive description of the two semantic formalisms we will be using throughout the thesis: AMR and DRS. We also review the previous work on semantic parsing. Most importantly, we describe our neural sequence-to-sequence model in detail, starting from a general definition of neural networks. Finally, we describe the advantages and disadvantages of character-level models, as well as a number of studies that successfully utilized them.

Part II - Abstract Meaning Representation

Part II contains our first attempts at neural semantic parsing. In Chapter 4 we apply our sequence-to-sequence model to AMR parsing and describe a number of methods needed to achieve good performance (**RQ1**). We show that a character-level model greatly outperforms a word-level model (**RQ2**) and substantially improve performance by supplying the model with silver standard training data (**RQ3**). In Chapter 5 we focus on the aspect that was most challenging for our model: dealing with variable re-entrancies (**RQ4**).

Part III - Discourse Representation Structures

Part III details our experiments on producing meaning representations based on formal semantics. To be able to evaluate our semantic parser, we develop an automatic evaluation method for DRS in Chapter 6 (**RQ5**). We are interested if the lessons we learned in Part I are still applicable when training a neural semantic parser on this more challenging domain. In Chapter 7, we find this is indeed the case: our model outperforms baseline DRS parsers (**RQ1**), with characters as the preferred representation (**RQ2**), while benefiting a lot from using silver standard data (**RQ3**). Best performance is reached by rewriting the variable names to a more general representation, based on the order they were introduced (**RQ5**). Finally, in Chapter 8 we show that exploiting linguistic information has modest, but significant benefits for the character-level model (**RQ6**).

Part IV - Characters & Contextual embeddings

Part IV is concerned with a relatively new phenomenon in NLP: pretrained contextual language models. In Chapter 9 we outline the effect they had on the field and try to determine whether character-level representations can still be useful. We devise two different methods of combining character-level and contextual language model representations, which we test on both AMR and multi-lingual DRS. We find that they can indeed still be useful, with modest, but consistent, improvements across different formalisms, data sets, language models and languages (**RQ7**).

Part V - Conclusions

Part V concludes with Chapter 10, which provides an overview of our main conclusions based on the answers on our research questions listed above. Finally, we reflect on our work in this thesis and outline possible directions for future work in (neural) semantic parsing.

1.3 Publications

This thesis is based on the following peer-reviewed publications:

1. van Noord, R. and Bos, J. (2017c). Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108 (**Chapter 4**)
2. van Noord, R. and Bos, J. (2017b). The Meaning Factory at SemEval-2017 task 9: Producing AMRs with neural semantic parsing. In *SemEval*, pages 929–933, Vancouver, Canada (**Chapter 4**)
3. van Noord, R. and Bos, J. (2017a). Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 41–49, Montpellier, France (**Chapter 5**)
4. van Noord, R., Abzianidze, L., Haagsma, H., and Bos, J. (2018a). Evaluating scoped meaning representations. In *LREC*, pages 1685–1693, Paris, France. ELRA (**Chapter 6**)
5. van Noord, R., Abzianidze, L., Toral, A., and Bos, J. (2018b). Exploring neural methods for parsing discourse representation structures. *TACL*, 6:619–633 (**Chapter 7**)
6. van Noord, R., Toral, A., and Bos, J. (2019). Linguistic information in neural semantic parsing with multiple encoders. In *IWCS - Short Papers*, pages 24–31, Gothenburg, Sweden (**Chapter 8**)
7. van Noord, R. (2019). Neural Boxer at the IWCS shared task on DRS parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden (**Chapter 8**)
8. van Noord, R., Toral, A., and Bos, J. (2020). Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *EMNLP*, pages 4587–4603, Online. ACL (**Chapter 9**)

During the course of this thesis, the author was also involved in these publications:

9. Abzianidze, L., Bjerva, J., Evang, K., Haagsma, H., van Noord, R., Ludmann, P., Nguyen, D.-D., and Bos, J. (2017). The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *EACL: Volume 2, Short Papers*, pages 242–247, Valencia, Spain
10. van der Goot, R., van Noord, R., and van Noord, G. (2018). A taxonomy for in-depth evaluation of normalization for user generated content. In *LREC*, Miyazaki, Japan. ELRA
11. Kuijper, M., van Lenthe, M., and van Noord, R. (2018). UG18 at SemEval-2018 task 1: Generating additional training data for predicting emotion intensity in Spanish. In *SemEval*, pages 279–285, New Orleans, Louisiana
12. Veenhoven, R., Snijders, S., van der Hall, D., and van Noord, R. (2018). Using translated data to improve deep learning author profiling models. In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*, volume 2125
13. Abzianidze, L., van Noord, R., Haagsma, H., and Bos, J. (2019). The first shared task on discourse representation structure parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden
14. Nissim, M., van Noord, R., and van der Goot, R. (2020). Fair is better than sensational: Man is to doctor as woman is to doctor. *Computational Linguistics*, 46(2):487–497

PART I

Background

CHAPTER 2

Computational Semantics

2.1 Semantic Formalisms

This thesis will focus on two semantic formalisms: Abstract Meaning Representation and Discourse Representation Structures. In this section, we provide a detailed overview of both formalisms.

2.1.1 Abstract Meaning Representations

The first semantic formalism we will be focusing on in this thesis is Abstract Meaning Representation (AMR). The basics of this formalism were first described by Langkilde and Knight (1998), though only after the release of the first AMR corpus (Banarescu et al., 2013) it gained popularity in the field of NLP. AMR aims to model the meaning of individual sentences by assigning them a rooted, labeled and directed graph, derived from the PENMAN notation (Kasper, 1989; Bateman, 1990). It abstracts away from syntax: sentences that have the same basic meaning should have the exact same semantic graph. AMRs are not created compositionally, i.e., there is no necessary alignment between the words in a sentence and the semantic structures.

An example AMR is shown in Figure 2.1. There are three common ways to represent AMRs: in string format (top left), graph format (top right) and triple format (bottom). In this section we will use the prettier graph format, but in other chapters we will be using the string format, as it better matches how our AMR parser handles the data. The triple format is commonly only used during evaluation.

Figure 2.1: Three equivalent AMR representations for the sentence *The clown wants to perform*: string tree format (top left), graph format (top right) and triple format (bottom).

AMRs consist of *concepts* (graph nodes) and *relations* (graph edges). AMR concepts are closely related to the words in the sentence. They can either be English words (`clown`), PropBank (Palmer et al., 2005) framesets (`want-01`, `perform-01`) or special keywords (e.g., `have-org-role-91`). The PropBank frames are used to distinguish different word senses, as well as selecting the correct relations for that frame. For example, `fold-02` has an `:ARG0` that should indicate a person, while `:ARG1` should represent *hands*, used when someone folds their hands. However, `fold-04` only has an `:ARG0`, which is a card player, used when, for example, a poker player folds a set of cards in a game. AMR does not differentiate between nouns and verbs: PropBank frames are often used to represent nouns as well, for example using `opine-01` to describe someone having an opinion.

The PropBank arguments are not the only possible arguments: AMR contains relations that cover general semantic relations (`:poss`, `:polarity`), quantities (`:quant`, `:unit`), dates (`:day`, `:year`) and lists (`:op1`, `:op2`). AMRs can also model coreference and control structures by using graph re-entrancies, i.e., adding an extra edge to an existing node, as is shown in Figure 2.1. In this case, the clown is both the *wanter* and the *performer*. AMR

handles this by assigning a variable to each instance, which can then later be used to indicate re-entrancy. In Chapter 5, we look into this phenomenon in more detail.

AMR handles negation by using the `:polarity` relation with a negative constant. It represents the scope of the negation by where this node attaches in the graph, see, for example, the difference between the left and right AMR in Figure 2.2. Named entities in the AMR are represented by the `:name` relation, with standardized forms such as `person`, `country` and `organization`, while they are grounded using wikification (Cucerzan, 2007), as is shown in Figure 2.3 (left). Here, the named entity *Dick Advocaat* is linked to the Wikipedia page *Dick_Advocaat*.

Figure 2.2: AMRs for *The clown does not have to laugh* (left) and *The clown must not laugh* (right). The attachment of the polarity determines the scope of the negation.

There are also a number of simplifications and drawbacks of AMR. First, AMR is heavily biased towards English, and not meant to be an interlingua.¹ Second, it does not consider grammatical number and definiteness, with the consequence that *The clowns want to perform* and *A clown wants to perform* have identical graphs. Third, it does not model tense and aspect, meaning

¹There are efforts towards multi-lingual AMR, with annotation efforts for Czech (Xue et al., 2014), Chinese (Li et al., 2016), Brazilian Portuguese (Anchiêta and Pardo, 2018; Sobrevilla Cabezudo and Pardo, 2019) and Spanish (Migueles-Abraira et al., 2018). Moreover, there are case studies for Korean (Choe et al., 2019) and Vietnamese (Linh and Nguyen, 2019).

Figure 2.3: AMR representations for the sentence *Dick Advocaat wants to perform* (left) and *No one can resist* (right).

that *The clown wanted to perform* has the same graph as well. Fourth, it does not handle quantifier scope in a principled way. For example, see how AMR models *No one can resist* in the right AMR of Figure 2.3.² Bos (2016) showed that there is a way to formally model universal quantification in AMR, but this method can at most handle a single universal quantifier. Moreover, Pustejovsky et al. (2019) showed that there also can be ambiguities for a single universal quantifier: *Everyone in the room listened to a talk* has the same graph structure for everyone listening to the same talk, or each person listening to a different talk.

However, these simplifications also make AMR an easy to understand formalism and allowed a large number of people to annotate AMRs, which in turn led to the release of large-scale corpora. Before this, meaning representations data sets were generally small and not always fully manually annotated. The release that we will be working with in this thesis (LDC2017T10) already contains 39,260 gold standard AMRs, while the most recent release (LDC2020T02) has 59,255. This allowed for the development of a large number of AMR parsers, which will be discussed in Section 2.2, Chapter 4 and Chapter 9.

²We based the AMR on the gold standard example *No one can enter otherwise*.

2.1.2 Discourse Representation Structures

Discourse Representation Structures (DRSs) are formal meaning representations based on Discourse Representation Theory (DRT, Kamp 1984; Kamp and Reyle 1993). DRT is a formalism that explores meaning representations based on formal semantics. The main reason for its development was its capability in handling *donkey sentences* (Geach, 1962) such as *Every farmer who owns a donkey beats it*. The semantic formalisms at the time could not account for the fact that the indefinite description of *a donkey* has to play the role of a universal quantifier when referred to by the pronoun *it*. DRT resolved this by introducing *discourse referents*, which are referents to each non-anaphoric noun phrase in the discourse, that can then potentially be used to bind anaphora that occur later in the discourse. In general, DRT can serve as a unifying formalism to represent a large number of semantic phenomena in a single meaning representation (DRS). Intuitively, a DRS can be seen as a mental representation of the discourse by the hearer. Among others, DRT has been used to study presuppositions (Van der Sandt, 1992), rhetorical structure (Asher and Lascarides, 2003) and conventional implicatures (Venhuizen, 2015). Another nice feature of DRSs is that they can be automatically translated to first-order logic (Muskens, 1996), which in turn can aid programs that perform inference on natural language (Blackburn et al., 1998; Blackburn and Bos, 2005). Next, we will discuss the specific dialect of DRS that we will be using throughout this thesis. This is the format that is used in the Parallel Meaning Bank project (Abzianidze et al., 2017).

A full DRS is commonly seen as a collection of *boxes*. A box consists of *discourse referents* and *conditions*. The discourse referents are indicators of discourse elements, e.g., persons or events. The conditions assert information over these discourse elements. For example, Figure 2.4a shows a simplified DRS for the sentence *Tom owns a credit card*. There are two discourse referents, x_1 and x_2 . The conditions then assert that x_1 is named Tom, x_2 is a credit card, and that x_1 is the owner of x_2 . The conditions can also be interpreted as *truthconditions*, which are satisfied if there indeed exists someone named Tom that owns a credit card. In our DRS interpretation, we represent Tom (x_1) a bit more formal: x_1 is a male that is named Tom (see Figure 2.4b).

(a)		(b)		(c)		
x1	x2	x1	x2	x1	x2	e1
Tom (x1)	credit card (x2)	male (x1)	name (x1, Tom)	male.n.02 (x1)	Name (x1, "tom")	
owns (x1, x2)		credit card (x2)	owns (x1, x2)	credit_card.n.01 (x2)		
				own.v.01 (e1)	Pivot (e1, x1)	
					Theme (e1, x2)	

Figure 2.4: Example DRSs for the sentence *Tom owns a credit card*. A simplified DRS is shown in (a) and extended in (b) by using a more principled representation of named entities. In (c), the concepts are grounded in WordNet, while the roles are grounded in VerbNet.

DRS conditions can either be basic or complex. A basic DRS condition can be one of three types: a concept, a role or a comparison operator. The concepts are grounded using WordNet (Fellbaum, 1998), indicating the lemma, part-of-speech and sense number. This can span multiple tokens in the input: *credit card* is represented as `credit_card.n.01`. We use neo-Davidsonian event semantics to represent *events* (Parsons, 1990). An event, usually introduced by a verb, has its own discourse referent (e_1). The verb that invoked this event is represented using a WordNet synset (`own.v.01`), while the roles the participants play in this event are grounded in VerbNet (Bonial et al., 2011). For our example sentence, the verb *own* introduces the role `Pivot` for the thing that owns something, while `Theme` is used for the thing that is owned. These conditions are two-place predicates, as they have to relate the event to the event participants. This DRS is shown in Figure 2.4c. Lastly, the comparison operators can be used to relate and compare discourse referents to each other, such as $x_1 < x_2$ (x_1 is smaller than x_2) or $x_1 \text{ SZN } x_2$ (x_1 is spatially under x_2). The arguments of the DRS conditions are called *terms*. Terms are usually the discourse referents (which are often called variables), but can also be *constants*. Constants are used to represent, among others, discourse direction ("speaker", "hearer"), questions ("?"), names ("tom"), quantities ("40") and tense ("now").

Complex conditions, on the other hand, are used to indicate logical relations between the sets of conditions and can represent their scope. They are defined as follows:

- If B is a DRS, then $\neg B$, $\Diamond B$ and $\Box B$ are complex conditions;
- If x is a variable, and B is a DRS, then $x:B$ is a complex condition;
- If B and B' are DRSs, then $B \Rightarrow B'$ and $B \vee B'$ are complex conditions.

We can use the complex conditions to model *Tom doesn't own a credit card* as shown in Figure 2.5a. Moreover, we now have a principled way of modelling *No one can resist* (Figure 2.6a), as opposed to the right AMR in Figure 2.3. It can be interpreted as “It is not the case that there exists a person for whom it is possible to be an agent in a resisting event”. We also want to model tense, so we can model the difference between *Tom doesn't own a credit card* and *Tom didn't own a credit card*. In English, the tense is usually introduced by the main verb in the sentence. It is modeled as a time period ($t1$), represented by the concept `time.n.08` and role `Time`. The tense is then represented by conditions such as $\prec$ (temporally precedes) and $=$ (equal to). The latter is used for the present tense, in which the time period equals the constant “now”, as in $t = \text{“now”}$ in Figure 2.5b.

Figure 2.5: Example DRSs for the sentence *Tom doesn't own a credit card*, without tense (a) and with tense (b).

The conditions for *Tom* in Figure 2.5a are modeled outside the negation, because the sentence implies that a male named Tom exists, whether he owns a credit card or not. This is known as *projected content* or a *presupposition*. In our dialect of DRS, we model these presuppositions in separate boxes outside of the main box of the DRS, as is shown in Figure 2.6b, based on Van der Sandt (1992) and projective DRT (Venhuizen et al., 2018).

Figure 2.6: Example DRSs for the sentence *No one can resist* (a) and *Tom doesn't own a credit card* (b). In (b), we model the presupposition in a separate box.

Even though our current format in principle can handle multi-sentence documents, understanding the relation between different discourse segments is known to be a necessary component of discourse understanding (Grosz and Sidner, 1986). We model this by using *rhetorical relations* from Segmented DRT (Asher, 1993; Asher and Lascarides, 2003). In essence, this allows us to describe the discourse relations between different (possibly nested) boxes of a DRS. Each box is given its own identifier (**b1**, **b2**, etc) so we can indicate the relations between the boxes. Common relations include **CONTINUATION** (*I can't squeeze this orange. It's dry*), **CONSEQUENCE** (*He who has the money, also has the power*) and **CONTRAST** (*The telephone rang, but no one answered*, see Figure 2.8 on page 21). A definition of DRS in Backus-Naur form is given in Figure 2.7, mostly based on Abzianidze et al. (2019).

```

<DRS>                ::= {<DRS>} <box>
<box>                 ::= <simple box> | <segmented box>
<simple box>           ::= {<referent>} {<condition>}
<condition>           ::= <basic condition> | <complex condition>
<term>                ::= <referent> | <constant>
<basic condition>     ::= <concept><pos_sense_number>(<term>)
                        | <semantic role> (<term>, <term>)
                        | <term> <comparison operator> <term>
<complex condition>   ::= ¬ <box> | ◇ <box> | □ <box>
                        | <box> ⇒ <box>
                        | <referent> : <box>
<segmented box>       ::= {<box>} {<condition>}
<condition>           ::= <relation> (<label>, <label>)

```

Figure 2.7: A definition of DRS given in Backus-Naur form.

The DRS dialect we will be using in this thesis is based on the Parallel Meaning Bank (Abzianidze et al., 2017), which in turn was heavily based on the Groningen Meaning Bank (Basile et al., 2012a,b; Bos et al., 2017). We will describe these two corpora below.

Groningen Meaning Bank The first large corpus annotated with DRSs was the Groningen Meaning Bank (GMB). It was designed to unify a large range of semantic phenomena in a single formalism. The GMB focuses mostly on multi-sentence documents of news wire texts. The latest release (2.2.0) contains 10,000 documents and over a million tokens. The DRSs are not annotated from scratch, rather they are corrected versions of automatically generated output. However, the annotations are not on the level of the final meaning representation, but focus on correcting intermediate *layers*. Each layer focuses on a different aspect of syntax and semantics, which can be corrected at the token-level. There are ten different layers in the GMB: tokenization, POS-tagging, lemmatization, CCG supertagging, named entity recognition, animacy tagging, word sense disambiguation, thematic role labelling, scope annotation and coreference resolution. The output of these layers is then fed to the rule-based semantic parser Boxer (Bos, 2008b), which produces the final DRS. There is a trained tagger available for a number of these layers, which could be retrained when more annotations became available.

An issue with the GMB is that there is no set of gold standard documents available for evaluation, as the released documents are only partially corrected by human annotators. For semantic parsing this is problematic, as it is unclear to what extent we are modelling the output of Boxer, instead of learning how to produce accurate DRSs.

Parallel Meaning Bank The follow-up project of the GMB is the Parallel Meaning Bank (PMB). This is a corpus of *parallel* texts annotated with DRSs. English is used as the pivot language, i.e., each document contains English, while translations can either be in German, Italian or Dutch. It follows the same process of layer-wise annotation by correcting automatically produced token-level tags. The layers, though based on the same principles, are changed to allow them to be language-neutral. Also, the POS-tagging, named entity recognition and animacy tagging layers are resolved in a single semantic tagging layer (Bjerva et al., 2016b; Abzianidze and Bos, 2017). Moreover, the concept and role symbols in the final DRS are grounded in WordNet (Fellbaum, 1998) and VerbNet (Bonial et al., 2011). The aim of the PMB is that the final DRS is language-neutral, i.e., the final DRS of *Tom doesn't own a credit card* and its Dutch translation *Tom heeft geen creditcard* should be equivalent if the translation is meaning preserving. The PMB contains a large number of shorter, sentence-level documents, which allows for a set of gold standard DRSs that can be used during training, and most importantly, evaluation. The work in this thesis will therefore focus on the PMB; a more detailed introduction of the corpus is given in Chapter 6. Specifically, we will be working with PMB release 1.0.0 (Chapter 6), 2.1.0 (Chapter 7), 2.2.0 (Chapter 8) and 3.0.0 (Chapter 9). Detailed descriptions of these data sets will be provided in the respective chapters.

2.1.3 Comparing AMR and DRS

There are some notable differences between AMR and our variant of DRS. In short, DRS is a more expressive formalism than AMR, as it models more semantic phenomena. The main difference is that DRS models scope explicitly, allowing a more principled representation of negation and quantification. Another major difference is that DRS can handle multi-sentence documents

Figure 2.8: Example AMR (top) and segmented DRS (bottom) for the sentence *The telephone rang, but no one answered.*

by using explicit discourse relations, while AMR is designed to only give sentence-level representations (though it can handle some multi-sentence cases). DRS also explicitly models presuppositions, which is not done in AMR.

There are also a few smaller differences. DRS concepts and roles are grounded in WordNet (Fellbaum, 1998) and VerbNet (Bonial et al., 2011), respectively, while AMR only uses PropBank (Palmer et al., 2005) for verbs, leaving nouns ungrounded. On the other hand, AMR grounds named entities using wikification (Cucerzan, 2007), while DRS does not have such a component. Another difference is that DRS models tense, meaning it has different meaning representations for things that already happened and things that will happen in the future. In Figure 2.8, we highlight the difference in expres-

siveness of AMR and DRS. DRS explicitly models past tense (*rang, answered*) and presupposition (*telephone*), while also having a principled treatment of the negation (*no-one*).

We believe these formalisms make for an interesting testing domain for semantic parsers, as they aim to model meaning from quite different perspectives. In this sense, we believe that if a system is able to successfully model both AMR and DRS, it is likely that it can also learn other semantic formalisms. We compare AMR to DRS in more detail in Chapter 6.

2.2 Semantic Parsing

Semantic parsing is the task of automatically mapping a natural language text into a formal, interpretable meaning representation. Informally speaking, a meaning representation describes *who did what to whom, when, and where, and to what extent this is the case or not*. In this section, we will describe previous semantic parsing approaches, ranging from the traditional rule-based systems to the recent neural network models. We will discuss open domain and closed domain approaches in separate sections, with particular interest for AMR and DRS parsing systems.

2.2.1 Rule-based Approaches

Already since the 1950s, extracting the meaning of a sentence was thought to be a major component of possible automatic machine translation (Weaver, 1955; Masterman, 1961). The first studies on what now would be considered *semantic parsing* emerged in the 1970s. SHDRU (Winograd, 1972) and LUNAR (Woods et al., 1972) were systems that constructed a semantic representation based on its syntactic analysis by applying a set of rules. SHDRU was a system that could manipulate a block world based on user input, while LUNAR could answer questions based on a database of Apollo 11 research. Schank (1975), in his overview work, took it a step further and proposed that any natural language processing problem has three different parts: (i) mapping sentences to a meaning representation, (ii) processing this meaning representation and (iii) translating the produced meaning representation back to natural language. The developed system, MARGIE, could make para-

phrases and inferences from natural language and was based on Conceptual Dependency Theory (Schank, 1972). Wilks (1972) introduced an English-French machine translation systems that worked similarly. It first mapped the English input sentence to a meaning representation using Preference Semantics (Wilks, 1975), which was then used to generate the translated French sentence.

These parsers were heavily rule-based, though, since they relied on syntactic and semantic grammars. This remained the dominant approach until the 1990s, with approaches including Hendrix et al. (1978), Damerau (1981), Templeton and Burger (1983), Johnson and Klein (1986), Pereira and Shieber (1987) and many others. The approaches usually followed the same recipe: a rule-based transformation is applied on the syntactic analysis of the sentence, with the aim to create a formal (meaning) representation. These rules were hand-crafted, limited to the domain they were designed for and required considerable domain expertise during design. This severely limited the general applicability of the created systems.

2.2.2 Closed Domain Semantic Parsing

The common definition of semantic parsing also includes tasks which map natural language sentences to computer-interpretable interpretations, such as text-to-SQL (ATIS, Hemphill et al. 1990; Dahl et al. 1994) and text-to-prolog (GeoQuery, Zelle and Mooney 1996). These tasks had corresponding annotated data sets on which performance could be evaluated, which drove the creation of data-driven parsers. These parsers could at least (partially) *learn* how to produce the final representations by using statistical methods (Pieraccini et al., 1992; Miller et al., 1994; Zelle and Mooney, 1996). Models on these types of data sets remained dominant through the 2000s, with, among others, approaches based on lambda-calculus (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007), parse trees (Ge and Mooney, 2005, 2009), support vector machines (Kate and Mooney, 2006), tree transducers (Jones et al., 2012) and statistical machine translation techniques (Wong and Mooney, 2006; Andreas et al., 2013). The need for annotated training data could be avoided by using weakly supervised or even unsupervised learning methods (Clarke

et al., 2010; Goldwasser et al., 2011; Poon, 2013). More recently, successful approaches included neural sequence-to-sequence (Xiao et al., 2016; Jia and Liang, 2016) and sequence-to-tree (Dong and Lapata, 2016) models.

2.2.3 Open Domain Semantic Parsing

Most of the previous parsers were only developed to work on a single domain. In the context of this thesis, however, we are interested in parsers that produce general purpose *deep* meaning representations on *open domain* natural language sentences. For example, the ATIS and GeoQuery data sets contain only sentences about either flight information or geography in the United States, which is of course a huge simplification of natural language. We also consider semantic role labelling (Gildea and Jurafsky, 2000) to be *shallow*, though it is an integral part of both the AMR and DRS representations. Other semantically annotated corpora do not combine their semantic annotations in a single representation, such as FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), the Penn Discourse Treebank (Prasad et al., 2008) and OntoNotes (Hovy et al., 2006). In this section, we will discuss approaches that aimed to produce a single meaning representation that contains a variety of semantic phenomena on open domain texts.

Initial approaches Initial approaches that pursued *open domain semantic parsing* were based on the Combinatory Categorical Grammar (CCG) formalism (Bos et al., 2004; Bos, 2005), which led to the development of the semantic parser Boxer (Bos, 2008b). Boxer is a combination of statistical (tokenization, POS-tagging, named entity recognition and CCG parsing) and rule-based methods that produces Discourse Representation Structures (DRSs, Kamp and Reyle 1993). A similar approach was taken by Allen et al. (2008), in that it used statistical methods to provide features to a hand-built grammar with semantic restrictions, using logical form language (Allen et al., 2007) as the semantic formalism. Another approach within this tradition was Minimal Recursion Semantics (Copestake et al., 2005), for which grammar based parsers were proposed (Copestake, 2007).

Evaluation An issue with these approaches was that it was not immediately clear how to evaluate them (Bos, 2008a). An attractive option is to com-

pare the produced semantic structures to a gold standard (glass box evaluation). However, creating such a gold standard is not straightforward, as semantic annotation is a hard and time-consuming task. Moreover, while this gives us an accuracy of parsers for a certain formalism, we cannot compare the adequacy of the formalisms themselves. Another option is black-box evaluation, in which the performance of models is judged on how well they do on a downstream task, such as Recognizing Textual Entailment (Dagan et al., 2005). Currently, glass-box evaluation is the common method of evaluating semantic parsers. AMR has a well-defined glass-box evaluation method (Cai and Knight, 2013), while we develop a glass-box evaluation method for DRSs in Chapter 6.

AMR parsing In 2013, the Abstract Meaning Representation (AMR) corpus was released (Banarescu et al., 2013). In Section 2.1.1, we gave a more detailed overview of AMR, but in short, AMR aims to give a structured representation of the meaning of a sentence in a single rooted, directed graph, consisting of relations (edges) and concepts (nodes). The first AMR parsing approaches were heavily based on syntactic parsing techniques. Flanigan et al. (2014) and Flanigan et al. (2016) use a two-step model that identifies concepts and relations separately, with relation prediction based on the maximum spanning tree algorithm used in dependency parsing (McDonald et al., 2005). A transition-based method was proposed by Wang et al. (2015a,b), in which a dependency parse is transformed to an AMR graph. Extensions to this method were proposed by Goodman et al. (2016) who used imitation learning and Damonte et al. (2017) who processed the sentence left-to-right based on the Arc-Eager dependency parser (Nivre, 2004). Peng et al. (2015) used hyperedge replacement grammar to produce the AMR graphs, while other approaches were based on CCG (Artzi et al., 2015; Misra and Artzi, 2016) or statistical machine translation (Pust et al., 2015).

Neural AMR parsing The previous approaches have in common that they are dependent on syntactic parsers, grammars or specific alignment between the words and graph fragments. We would prefer a method that does not have such dependencies, as they (i) introduce an extra step of complexity and possible errors, (ii) are often not available for non-English

languages and (iii) can be hard to transfer to other domains. Barzdins and Gosko (2016) proposed such a method: a character-level neural sequence-to-sequence model for AMR parsing, which does not depend on any linguistic information. However, it was quite far removed from state-of-the-art performance. Our AMR parser in Chapter 4 is based on this approach, with an overview of the model given in the next chapter. Contemporary to our work in Chapter 4, Peng et al. (2017b) and Konstas et al. (2017) also apply sequence-to-sequence models to AMR parsing. However, both models are word-based instead of character-based and depend on extensive anonymization of the input to reach good performance, which was still not close to the state-of-the-art. Foland and Martin (2017) did obtain state-of-the-art by using five bi-LSTM networks to produce the AMRs, though they are still dependent on specific alignments. Also, a number of AMR parsers were developed through the course of this thesis. We describe and compare to those approaches in Chapter 9.

DRS parsing Early DRS parsing approaches were either fully rule-based (Johnson and Klein, 1986; Wada and Asher, 1986) or relied on rules in combination with statistical methods (Bos, 2008b). The introduction of the Groningen Meaning Bank (GMB, Basile et al. 2012a,b; Bos et al. 2017)³ allowed for the emergence of parsers based on supervised learning. The first parser on GMB data was proposed by Le and Zuidema (2012) who converted the DRSSs to graphs and trained a parser with dependency parsing techniques to combine partial graphs to a full graph. They also use a probabilistic model to learn the lexicon with partial graphs, as opposed to having a fixed lexicon when using lambda calculus. The GMB was followed-up by the Parallel Meaning Bank (PMB, Abzianidze et al. 2017), which annotates parallel texts in English, German, Italian and Dutch with DRSSs. An initial release of this data was used by the cross-lingual CCG-based parser of Evang and Bos (2016). The PMB is described in more detail in Chapter 6, while we describe our own DRS parsers trained on PMB data in Chapters 7, 8 and 9, as well as other (contemporary) developed DRS parsers (Liu et al., 2018a, 2019a; Fancellu et al., 2019; Fu et al., 2020).

³See Section 2.1.2 for a more detailed description of the GMB.

Other formalisms There are a number of semantically annotated corpora that consist of semantic dependencies, which are somewhere in between semantic role labelling and full, deep meaning representations (Hajič et al., 2012; Ivanova et al., 2012). These were the target meaning representations in two consecutive shared tasks (Oepen et al., 2014, 2015), in which the best systems used an extension of an existing dependency parser (Martins and Almeida, 2014) and an SVM-based sequence labelling approach (Kanerva et al., 2015). Later, neural models improved performance by applying multi-task learning (Peng et al., 2017a, 2018; Stanovsky and Dagan, 2018). The composition-based parser of Lindemann et al. (2019) achieved state-of-the-art on a number of semantic graph banks and is described in more detail in Chapter 9. Another formalism is Universal Cognitive Conceptual Annotation (UCCA, Abend and Rappoport 2013), a graph-based semantic formalism that is not based on a syntactic foundation, similar as AMR. It consists of multiple layers, in which each layer represents a semantic distinction. The first UCCA parser was a neural transition-based approach (Hershcovich et al., 2017), on which they later improved by using multi-task learning (Hershcovich et al., 2018).

CHAPTER 3

Sequence-to-sequence Architecture

In this chapter we give a detailed description of the artificial neural network (NN) model that we will be using throughout the thesis. To put that in a wider context, we will also give a brief overview of the history of NNs, though the reader is assumed to have some background knowledge in the workings of supervised machine learning and basic feed-forward neural networks. There are many excellent resources available that explain machine learning (Hastie et al., 2009) and NNs (Goodfellow et al., 2016; Goldberg and Hirst, 2017) in detail.

3.1 Neural Networks

In a very basic sense, neural networks are systems that, given a set of *functions* and *weights*, automatically convert a certain input to a certain output. The first work that is considered to be *neural* is the McCulloch-Pitts neuron (McCulloch and Pitts, 1943), which was an early model of brain function. For the model to work, a human operator had to manually set the weights. The first models that could actually *learn* the respective weights were the perceptron (Rosenblatt, 1958) and ADALINE (Widrow and Hoff, 1960). The latter modified the weights by using stochastic gradient descent (SGD), which is still a commonly used method of training deep learning models. These linear models had severe limitations, though, as they famously could not model the XOR-function (Minsky and Papert, 1969). The first real breakthrough took place in the 1980s, with the introduction of non-linear activation functions combined with backwards propagation of errors (backpropagation, Rumel-

hart et al. 1986). This paper gave us the definition of the well-known feed-forward neural network (FFNN) with 1 hidden layer:¹

$$\text{FFNN}(\mathbf{x}) = g(\mathbf{W}^1 \mathbf{x}) \mathbf{W}^2 \quad (3.1)$$

Here $\mathbf{W}^1$ and $\mathbf{W}^2$ are weight matrices for the linear transformations, $\mathbf{x}$ the input vector, and g a non-linear activation function that is applied element wise. Common activation functions are the (logistic) sigmoid (σ) and hyperbolic tangent ($\tanh$), which are defined as follows:

$$\begin{aligned} \sigma(x) &= \frac{e^x}{e^x + 1} \\ \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \end{aligned} \quad (3.2)$$

This non-linearity is crucial, since it allowed the modelling of more complex phenomena, such as the XOR-function and most real-world problems. Without the non-linearity it does not matter how many matrices (layers) there are, the resulting model will always be linear.

Backpropagation works by calculating the gradient of the *loss function* and is still the backbone of all current neural network systems. The loss function measures how well our network is able to model the training set. During training, it calculates the difference between what the network would have predicted and the ground truth output labels. More intuitively, it sees the total loss as a geometric area, as is shown in Figure 3.1. By calculating the gradient of where we currently are, we can update the weights in such a fashion that the loss moves towards a local or global minimum.

Figure 3.1: Geometric representation of a (non-convex) loss function.

¹Throughout this thesis, biases are omitted for brevity.

In other words, training is performed by feeding input vectors to the model for which we know the correct outcome, calculating what it would have predicted in its current state, and measuring how far off we are (i.e., the loss function). The gradient of the loss function is then used in combination with an optimizer to change the weight matrices. This is how learning takes place: the network automatically changes its weight matrices to make more accurate predictions. Optimizers are functions that tell us *how* to update the weight matrices. They are discussed in more detail in Section 3.4.

In practice, we calculate the loss and update the weights over a sample of k input examples $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k)$ at each time step, since this is both more efficient and more stable. This is referred to as the *batch size*. Moreover, we can specify how much we want to update the weights by setting a predefined *learning rate*. In essence, the gradient gives us the direction of the step we want to take, while the learning rate determines how big of a step this is. Setting this value is an important step of training a model. If the learning rate is too small, we might get stuck in an undesirable local minimum, or the model will take too long to converge. If the learning rate is too large, we might miss desirable (local) minima.

Note that we are not simply interested in learning to model the training input; we want our trained model to be able to generalize to unseen examples. Therefore, we need to stop training at a certain point, as otherwise our system might only learn to perfectly model the training set (usually referred to as overfitting). The most common method is to have a held out validation set that is not used during training. After each i training examples or iterations over the training data, we calculate the loss on this held out set. If the loss stopped decreasing, we stop training and have our final model. This is often checked per *epoch*, which is a full iteration over the training set.

3.2 Recurrent Neural Networks

We have left the output of the model unspecified so far, but it can only predict a single value or single vector given a fixed size input vector. This is problematic for our purposes, since we are working with *sequences*, in both natural language and semantic structures. Therefore, we would like to have a model that can handle sequences of data. Elman (1990) proposed exactly this, an extension to FFNNs that can handle sequences of arbitrary lengths, called Recurrent Neural Networks (RNNs).² An RNN is basically a sequence of copies of the same FFNN, with connections between the steps in the sequence, which are referred to as *time steps*. An example of this network is shown in Figure 3.2.

Figure 3.2: Schematic overview of the simple RNN (Elman, 1990).

For each time step t over a sequence of input vectors $(x_1, x_2, \dots, x_k)$, the network takes the current input vector x_t into account, as well as the previous state of the RNN s_{t-1} , and calculates the new state s_t by applying a $\tanh$ non-linearity:

$$y_t = s_t = \text{RNN}(s_{t-1}, x_t) = \tanh([s_{t-1}; x_t] \mathbf{W}) \quad (3.3)$$

The initial vector s_0 is usually the zero vector. Importantly, the weights of this network are shared across all time steps. The loss is backpropagated through the network using backpropagation through time (Werbos, 1990).

²We describe the Elman (1990) RNN since this is most similar to the model we will be using, but note that there is earlier work describing (variants of) RNNs (Hopfield, 1982; Jordan, 1986).

A nice feature of an RNN is that we can easily stack *layers* on top of each other. The output of each time step of RNN^{n-1} is simply fed as input to the corresponding time step of RNN^n , with the loss backpropagating through the n layers.

However, there are two problems with this simple RNN architecture. For one, it suffers from *vanishing gradients*. Since, for long sequences, the output layer is quite far away from the first elements of the RNN, the gradient of those initial layers depends on a multiplication of a lot of numbers smaller than 1, resulting in such a small gradient that no learning takes place.³ Second, it has a hard time learning long term dependencies (Bengio et al., 1994), i.e., at the end of the sequence, the model did not retain enough information from the beginning of the sequence, which is clearly a problem when processing language. To get around these issues, Hochreiter and Schmidhuber (1997) introduced a method that became very popular in the field of natural language processing (NLP): Long Short Term Memory (LSTM).

The LSTM has two state vectors that are passed through the sequence: the cell state c and the hidden state h . Intuitively, the cell state functions as a long term *memory cell*, while the hidden state can be seen as the current working memory. Access to these cells is controlled by *gates*, which are train-

Figure 3.3: Schematic overview of the LSTM architecture.

³If we initialize the weights to be very large, we get the opposite problem, *exploding gradients*.

able sigmoidal layers that determine how much information is passed on to the next step. The *forget gate* f controls how much of the previous memory $\mathbf{c}_{j-1}$ we keep after seeing a new instance $\mathbf{x}_j$. The *input gate* i determines to what extent we add the new information to $\mathbf{c}_{j-1}$, while the *output gate* o controls what we will actually output as our new hidden state $\mathbf{h}_j$. An overview of a single LSTM block is shown in Figure 3.3. Mathematically, an LSTM is defined as follows:

$$\begin{aligned}
 \mathbf{s}_j &= \text{LSTM}(\mathbf{s}_{j-1}, \mathbf{x}_j) = [\mathbf{h}_j; \mathbf{c}_j] \\
 \mathbf{f}_j &= \sigma(\mathbf{W}_f \mathbf{x}_j + \mathbf{V}_f \mathbf{h}_{j-1}) \\
 \mathbf{i}_j &= \sigma(\mathbf{W}_i \mathbf{x}_j + \mathbf{V}_i \mathbf{h}_{j-1}) \\
 \mathbf{o}_j &= \sigma(\mathbf{W}_o \mathbf{x}_j + \mathbf{V}_o \mathbf{h}_{j-1}) \\
 \mathbf{z}_j &= \tanh(\mathbf{W}_z \mathbf{x}_j + \mathbf{V}_z \mathbf{h}_{j-1}) \\
 \mathbf{c}_j &= \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \mathbf{z}_j \\
 \mathbf{h}_j &= \mathbf{o}_j \odot \tanh(\mathbf{c}_j)
 \end{aligned} \tag{3.4}$$

Here, $\mathbf{W}$ and $\mathbf{V}$ are learnable weight matrices, σ is the sigmoid function and $\odot$ denotes element wise multiplication. The described architecture is the one that is commonly used currently and is the default implementation in the popular deep learning libraries Keras (Chollet et al., 2015) and Pytorch (Paszke et al., 2019). However, there exist many variants of this architecture. For example, the forget gate was not part of the original LSTM, but was introduced in subsequent work (Gers et al., 1999). Peephole connections (Gers and Schmidhuber, 2000) were added to improve on precise timing predictions, but were later found to not significantly improve the scores across a range of tasks (Greff et al., 2016). Cho et al. (2014b) introduced a simpler variant of the LSTM: Gated Recurrent Unit (GRU). This variant does not use the output activation function, and merges the input and forget gate into an update gate. GRU was found to generally not outperform LSTM (Chung et al., 2014; Jozefowicz et al., 2015), though can be an attractive choice in practice as it is faster and more memory efficient.

3.3 Sequence-to-sequence Models

There is still one final hurdle that we need to overcome before we can use this network for our semantic parsing tasks. This LSTM-based RNN can at most output one vector for each input vector it processes. This works perfectly well for sequence tagging tasks such as part-of-speech (POS) tagging, for which we need an output tag for each input word. However, in our case, the input sequence does not (necessarily) have the same length as the output sequence. Sutskever et al. (2014) proposed a *sequence-to-sequence* model to deal with this problem, often also referred to as the *encoder-decoder* architecture. They tested their approach on machine translation, but it quickly turned out to be useful for other tasks as well, such as syntactic parsing (Vinyals et al., 2015), text summarization (Rush et al., 2015) and (closed domain) semantic parsing (Xiao et al., 2016; Dong and Lapata, 2016).

Figure 3.4: Schematic overview of the basic sequence-to-sequence architecture using LSTMs.

In this model, an LSTM is run over a sequence of input vectors (encoding), after which its final vector s_k (often referred to as the context vector) is fed to a different LSTM, which produces the output (decoding). A schematic overview of the basic architecture is shown in Figure 3.4. During training, the decoder at time step j is fed with the previous decoder state d_{j-1} and the vector p_{j-1} of the previous target symbol t_{j-1} . This is known as teacher-forcing (Williams and Zipser, 1989). We can do this during training, but during prediction we obviously do not have access to these ground truth target

symbols. In that case, we use the output vector y_{j-1} of the previous step.⁴

Though we can now train a model for our semantic parsing tasks, there is an important extra mechanism that we will use to boost performance: *attention* (Bahdanau et al., 2015). The previously described encoder-decoder models have a clear bottleneck: all the information of the input sequence needs to be encoded in a single vector. This turned out to be a problem, especially for processing longer sequences (Cho et al., 2014a). Attention addresses this problem by allowing the decoder to have access to all encoder states $s_1 \dots s_k$ at each time step. The model is asked to *align* the input and target sequences (an analogy that works well in machine translation), but essentially learns to only pay attention to parts of the input sequence that are relevant for the current prediction. It calculates an attention vector $\mathbf{a}'_j$ by comparing the current hidden decoder state $\mathbf{d}_j$ to all encoder states as follows:

$$\begin{aligned} \mathbf{a}'_j &= (b_1 \dots b_k) \\ b_i &= f(\mathbf{d}_j, \mathbf{s}_i) \\ f(\mathbf{d}_j, \mathbf{s}_i) &= \frac{\exp(\text{score}(\mathbf{d}_j, \mathbf{s}_i))}{\sum_{l=1}^k \exp(\text{score}(\mathbf{d}_j, \mathbf{s}_l))} \end{aligned} \quad (3.5)$$

For the score function, we will be using either *general* or *dot-product* as defined by Luong et al. (2015):

$$\begin{aligned} \text{score}_{\text{gen}}(\mathbf{d}_j, \mathbf{s}_l) &= \mathbf{d}_j^\top \mathbf{s}_l \\ \text{score}_{\text{dot}}(\mathbf{d}_j, \mathbf{s}_l) &= \mathbf{d}_j^\top \mathbf{W}_a \mathbf{s}_l \end{aligned} \quad (3.6)$$

The resulting vector $\mathbf{a}'_j$ is of length k and is then used to calculate a weighted average over the encoder states to calculate the final attention vector $\mathbf{a}_j$:

$$\mathbf{a}_j = \sum_{i=1}^k \mathbf{a}'_{ji} \mathbf{s}_i \quad (3.7)$$

⁴We can do this in a clever way using *beam search*, which will be explained on page 39.

This final attention vector $\mathbf{a}_j$ is then concatenated to the current hidden state $\mathbf{d}_j$ and put through a linear layer with $\tanh$ non-linearity to produce the output vector $\mathbf{y}_j$:

$$\mathbf{y}_j = \tanh(\mathbf{W}_c[\mathbf{d}_j; \mathbf{a}_j]) \quad (3.8)$$

An overview of our current architecture is shown in Figure 3.5, with input vectors $\mathbf{x}_1 \dots \mathbf{x}_k$ and, during training, target vectors $\mathbf{t}_1 \dots \mathbf{t}_m$.

Figure 3.5: Schematic overview of the sequence-to-sequence model with attention.

So far, we have only worked with input and output *vectors*. However, we will working with natural language, so we have to find a way to (i) convert our input words⁵ to vectors and (ii) output words instead of vectors. For (i), we will be using an *embedding layer* that transforms each word to a vector. This idea was initially proposed by Rumelhart et al. (1986), though got its first breakthrough in Bengio et al. (2003). First, each word in our source vocabulary is converted to a randomly initialized vector. These vectors together form the weight matrix $\mathbf{W}_w$ during training, which we can update just as the other weight matrices, by applying backpropagation. This weight matrix

⁵Note that we are not limited to using words, we can also use characters as our preferred representation. We will discuss this in Section 3.5.

also functions as a lookup table during training, e.g., if we need a vector for the third word in the sequence, which is the 57th word in the vocabulary, we have $\mathbf{e}_3 = \mathbf{W}_{w_{57}}$. The loss is backpropagated all the way back to this layer, so the representation of each word is thus pushed to be as useful as possible, relevant to the task at hand.

There exist better methods than simply randomly initializing the initial word vectors. This is known as the concept of *pretrained word embeddings* (Mikolov et al., 2013; Pennington et al., 2014). Here, we train our weight matrix of words $\mathbf{W}_w$ on a different task for which lots of data is available (usually language modelling) and use those weights as a starting point when training our own model. The idea is that these learned weights represent knowledge about language in general, for example, words that are similar in meaning should have similar vectors. An issue with this type of embedding is that it completely ignores the context in which the words occur, i.e., it cannot differentiate between different senses of a word. This was addressed by *contextual embeddings* (Peters et al., 2018; Devlin et al., 2019), which do take the context into account, and had a major influence on the field. We will describe them in more detail in Chapter 9.

To be able to produce words as output (ii), we add a *softmax layer* as our final layer after decoding. First, we create a linear layer with a weight matrix $\mathbf{W}_t$ which contains a vector for each word in the target vocabulary. The output of this layer is a vector with the size of the number of words in the target vocabulary. This vector is then put through the softmax function, which forces the vector to sum to 1 and in essence creates a probability distribution over the full target vocabulary. At each decoding step, we can then simply take the word with the highest probability as our output, which is known as greedy decoding. Mathematically, it works as follows:

$$\begin{aligned}
 o_j &= \arg \max (\text{softmax}(\mathbf{v}_j)) \\
 \text{softmax}(\mathbf{v}_j)_i &= \frac{\exp(\mathbf{v}_{ji})}{\sum_{l=1}^k \exp(\mathbf{v}_{jl})} \\
 \mathbf{v}_j &= \mathbf{W}_t \mathbf{y}_j
 \end{aligned} \tag{3.9}$$

Here, o_j is the final output word and y_j is the output of the LSTM decoder (with attention) at timestep j . In practice, greedy decoding has been superseded by the use of *beam search*.⁶ Instead of only keeping a single prediction, it always keeps b predictions with the highest probability in memory. For example, for a beam size b of 3, we select the 3 best output words at timestep 1. We continue decoding at timestep 2 and calculate 3 new predictions for each of the initial predictions, giving us 9 current predictions. We then only select the 3 predictions with highest probability of those 9, and continue to timestep 3. We apply the same process there, and continue doing so, until we stop decoding. At this point, we simply take the sequence with the highest probability as our final prediction.

This process does not have a stopping criterion yet. We follow the common practical solution by appending a special end of sequence token ($\langle \text{EOS} \rangle$) to the target sequences of the training set. The model then learns to produce this token, and we then stop the process if we have produced this token during decoding. Similarly, we add a special beginning of sequence token ($\langle \text{BOS} \rangle$), so that we can feed a target embedding vector to the decoder when processing our first actual target symbol.

We have two final tricks to improve the performance of our sequence-to-sequence model. First, instead of only running the encoder LSTM from left-to-right, we run an LSTM in *both* directions, and concatenate the resulting vector at each timestep: $s_j = [\vec{s}_j; \overleftarrow{s}_j]$. This is what's called a *bidirectional* model, and has been proposed by Schuster and Paliwal (1997) for a standard RNN, while Graves (2008) extended it to the bidirectional LSTM (bi-LSTM). Second, we apply *dropout* (Srivastava et al., 2014), a regularization technique that works on the output of a certain layer, and sets individual output cells to 0 with a certain probability p .⁷ The intuition is that the model cannot learn to rely on only a few individual weights (as they can be dropped out), which in turn prevents overfitting. In RNNs, dropout is usually applied on the output of recurrent layers (naive dropout, Zaremba et al. 2014), but can also be applied within the recurrent layers (variational dropout, Gal and Ghahramani

⁶First uses in combination with RNNs include Graves (2012), Boulanger-Lewandowski et al. (2013) and Sutskever et al. (2014).

⁷This is not applied during prediction.

2016). We will describe the type of dropout we use for each experiment in the respective chapters.

This is the general model that we will be using for the semantic parsing experiments described in Chapters 4, 5 and 7. A schematic overview of the final model is shown in Figure 3.6. In Chapter 8 and 9 we extend this model by employing an extra encoder. Moreover, in Chapter 9 we will also experiment with the Transformer architecture (Vaswani et al., 2017) and pretrained contextual word embeddings (Peters et al., 2018; Devlin et al., 2019). Details of those architectures can be found in the respective chapters.

Figure 3.6: Schematic overview of the full sequence-to-sequence model with attention, for encoding the sentence *He bragged about it*. The first LSTM is the encoder, the second is the decoder. They can contain multiple layers, though only one is depicted here.

3.4 Hyper-parameters

The weights in the weight matrices are often described as the model *parameters* (usually referred to as θ) and they contain the values that the model *learns*. However, there are certain model settings that need to be set in advance, and are not learned by the model. These settings are usually referred to as *hyper-parameters*. It is crucial to set them (close) to their optimal value, but it is impossible to know these values in advance. That is why training a neural network always comes with a *tuning* stage, in which different combinations of hyper-parameter settings are tried out, until we are satisfied with the result. During tuning, we evaluate performance on a held out validation or development set (usually the same one that tells us when to stop training). These values can already tell us a lot about the performance of our model, but it is likely that we (at least partially) overfit on this validation set, especially if we run a large number of tuning experiments. That is why we measure the final performance on a second held out *test set*, which we never looked at during tuning.

There are also certain parts of the architecture that we can treat as hyper-parameters, i.e., we try multiple architectures and go with the architecture that has the best validation performance. This can include the type of RNN (LSTM or GRU), the type of attention (general or dot-product) or the addition of an extra linear between the encoder and decoder (see Chapter 8).

Perhaps the most important hyper-parameter is the *optimizer*, which is the function that determines how we update the weight matrices. The most common optimizer is SGD, which is also at the core of the other optimization functions. As explained above, SGD updates each parameter by looking at the gradient of the loss function, while using the predefined learning rate to determine how big each update must be. A problem with SGD is that each individual parameter is updated in the same way. However, we would prefer larger updates for the weights of infrequent tokens, while very frequent tokens do not need large updates, especially near the end of training. This principle was implemented by the AdaGrad algorithm (Duchi et al., 2011), which was shown to improve the robustness of SGD when training deep neural networks (Dean et al., 2012). However, a remaining issue with this algorithm

is that the learning rate keeps shrinking until it reaches practically zero, at which no learning is taking place anymore. Two approaches independently aimed to fix this problem (Zeiler, 2012; Tieleman and Hinton, 2012), and were later combined to form the currently most popular optimization function Adam (Kingma and Ba, 2015).⁸ In Chapter 4, 5 and 7 we will use the SGD optimizer, while in Chapter 8 and 9 we found Adam to be preferable.⁹ Please see Ruder (2016) for a more extensive and detailed overview of optimization functions. Specific settings for all hyper-parameters are reported in the respective chapters.

3.5 Characters versus Words

The input to a sequence-to-sequence model is a sequence of tokens. One of the main aims of this thesis is to determine what the best method of representing the input sentence is for neural semantic parsing: splitting the sentence into a sequence of characters, words, or somewhere in between. But, why would we even bother with individual characters, given that the more common word-level representations surely are a better general unit of meaning? There are a number of common arguments. For one (i), character-level models are potentially able to recognize that certain words are different morphological inflections of the same lexeme (e.g., *win*, *wins*, *won*, *winning*), while word-level models treat these as individual, non-related words. In other words, pure word-level models are completely blind to any prefixes, suffixes and stems. Second (ii), character-level models can handle words that the model did not see before. For example, even if it never saw *sprinting* before, it can form a representation based on the times it saw *sprint* or *sprinted* during training, while word-level models have to treat it as an unknown word. This is especially useful for languages in which compound nouns are combined in a single word (such as Dutch), as well as for noisy texts that contain a lot of spelling errors. A related third (iii) argument is that there are no unknown words in general for character-level models, while this can

⁸This is a quickly moving research field; there exist a number of newer variations on Adam, such as AMSGrad (Reddi et al., 2018), AdamW (Loshchilov and Hutter, 2019) and QHAdam (Ma and Yarats, 2019). We will not be experimenting with those.

⁹This was found during tuning of the respective systems.

be a problem for word-level models. This is more likely to be a problem for relatively small training sets, such as in semantic parsing. Fourth (iv), there is no need for explicit segmentation of the document in meaningful tokens, which simplifies preprocessing. Indeed, in practice, character-level models have had successes across a range of other NLP tasks. We will discuss those in the next section.

To be fair, arguments (i) and (ii) are already (partially) addressed by using pretrained word embeddings, be it static (Mikolov et al., 2013; Pennington et al., 2014) or contextual (Peters et al., 2018; Devlin et al., 2019). When trained on a very large corpus, the model will recognize that *win*, *wins*, *won*, *winning* are similar in meaning, as they always occur in similar contexts. Moreover, this will also reduce the number of unknown words, as we are not reliant on our training set to form our vocabulary, but can use all words that the model saw during pretraining. For contextual embeddings this often applies even more, as they use automatically learned *subword units* (Sennrich et al., 2016b) instead of full words. The idea of this method is that it, for example, splits *sprinting* to lexeme *sprint* and morphological inflection *ing*, reducing the data sparsity. This is not always what happens in practice, though, as the method is purely frequency-based. We will compare our character-level model to word-level models that use static embeddings in Chapter 4 and 7, and to models that use contextual embeddings in Chapter 9.

Of course, using character-level representations instead of word-level representations also has disadvantages. Most importantly, it greatly increases the average sequence length. This is not only more resource intensive, it also makes them harder to model, since capturing long term dependencies is a known problem of LSTMs (Cho et al., 2014a). Especially if we want to move to document-level analysis, only using characters might be infeasible. Moreover, the characters themselves do not carry meaning. The meaning of *car* cannot be compositionally inferred by looking at the meaning of *c*, *a* and *r* individually. This means the model has to be more aware of the meaning of combinations of sequences than it has to be for word-level models. Another disadvantage is the fact that we now have to explicitly model spaces to indicate word boundaries. Lastly, the fact that related words consist of the same characters is a potential advantage, but

there are of course also many words that are orthographically similar, but not related at all.

Note that the arguments in the previous paragraphs only hold for languages that use the Latin alphabet, as this thesis will only be concerned with such languages. We will mostly be using English data sets, though in Chapter 9 we experiment with DRS parsing for German, Italian and Dutch. For writing systems that use a different alphabet (e.g., Chinese, Japanese), a different story has to be told.

Previous Character-level Models

The power of character-level representations has long been known in the field. In earlier work, they were successfully used in a text-to-speech neural network (Sejnowski and Rosenberg, 1987) and a finite state transducer for grapheme to phoneme conversion (Kaplan and Kay, 1994). It must be noted, though, that using characters was the clear choice in these tasks, as their models had to produce phonemes. Church (1993), on the other hand, used character representations to align parallel texts, which worked well for noisy (and more realistic) texts. Dunning (1994) used character n-grams in his Bayesian model for automatically identifying languages. This type of character n-grams also worked well in information (document) retrieval (Cavnar, 1995; Damashek, 1995), named entity recognition (Cucerzan and Yarowsky, 1999; Klein et al., 2003) and authorship attribution (Peng et al., 2003). Vilar et al. (2007) trained an end-to-end statistical machine translation system with just characters as input (and not character n-grams), but did not get close to word-level performance. A similar system of Nakov and Tiedemann (2012), on the other hand, did outperform a word-level model, though they only evaluated on Macedonian-Bulgarian translation. Character-level models were presumed to be a good fit here, since the languages are closely related, and not many other resources are available.

More recently, character-level representations regained popularity as input representations for neural networks, starting with success in general language modelling with Recurrent Neural Networks (Sutskever et al., 2011). Santos and Zadorozny (2014) were the first to combine character and word-

level representation in a deep neural network, getting state-of-the-art results on POS-tagging. Ling et al. (2015) introduced an LSTM-based method that uses characters to obtain word-level representations, which in turn was used by Ballesteros et al. (2015) in their neural dependency parser. Plank et al. (2016) created a similar method for POS-tagging, evaluating on 22 different languages, noting that character-representations especially helped for Indoeuropean and Slavic languages.

Kim et al. (2016) introduced the char-CNN model, which uses characters to obtain word-level representations using a Convolutional Neural Network. Instead of seeing the word as just a sequence of characters (as in the previous LSTM-based models), this network can also look at n-grams of characters by applying different *filters*, making it a lot more powerful. Bojanowski et al. (2017) also applied this idea of exploiting character n-grams in their language modelling work, in which the final word representation is the sum of all n-gram representations of a word.

Chung et al. (2016b) showed that a character-level *decoder* can be beneficial in neural machine translation (NMT), without taking word-level information into account. Interestingly, Costa-jussà and Fonollosa (2016) changed the *encoder* of their NMT system to use the char-CNN, but did not change the decoder. Luong and Manning (2016) did use characters in both the encoder and decoder, but only used this information for unknown words, otherwise still relying on word representations.

In semantic parsing, if character-level representations are employed, they are commonly used in combination with non-contextual word-level representations, and not in an end-to-end fashion. For example, Ballesteros and Al-Onaizan (2017) and Groschwitz et al. (2018) concatenated a character-based representation to the word representation for each word. Barzdins and Gosko (2016) did use an end-to-end character-level model for AMR parsing, though their results were quite far from state-of-the-art performance. Our work in Chapter 4 is based on their approach.

Hybrid representation Sennrich et al. (2016b) applied the byte-pair encoding (BPE, Gage 1994) method to NMT, which can automatically convert words into sequences of subword units, without having to specify morphological information of a given language. It starts with the training set as a sequence of characters, and then iteratively replaces the most frequent pair of characters (e.g., 't', 'h') with the combined characters ('th') for a predefined number of iterations. This way, frequent words are still represented as full words, while less frequent words can be formed by multiple subword units. There are no unknown words anymore, at worst the model represents a word as a sequence of characters, thus seemingly eliminating the need for explicit character-level representations. The final representation can be seen as a hybrid between characters and words. Though this method quickly became the standard method in NMT, it was not until the release of BERT (Devlin et al., 2019) that it became the standard method for most NLP applications.¹⁰ We will compare our character-level model to a BPE-based model in Chapter 7, while in Chapter 9 we compare to a model based on BERT, as well as combination of BERT and characters.

¹⁰BERT used WordPiece during training, which is very similar to BPE. In Chapter 9 we describe BERT and its impact on the field in more detail.

PART II

Abstract Meaning Representation

CHAPTER 4

Neural AMR Parsing

We evaluate a character-level sequence-to-sequence model for neural semantic parsing on a large corpus of sentences annotated with Abstract Meaning Representations (AMRs). With some trivial preprocessing and postprocessing of AMRs, we obtain a baseline accuracy of 53.1. We examine five different approaches to improve this baseline result: (i) reordering AMR branches to match the word order of the input sentence increases performance to 58.3; (ii) adding part-of-speech tags (automatically produced) to the input shows improvement as well (57.2); (iii) So does the introduction of super characters (conflating frequent sequences of characters to a single character), reaching 57.4; (iv) optimizing the training process by using pre-training and averaging a set of models increases performance to 58.7; (v) adding silver-standard training data obtained by an off-the-shelf parser yields the biggest improvement, resulting in an F-score of 64.0. Combining all five techniques leads to an F-score of 71.0 on holdout data. This is remarkable because of the relative simplicity of the approach.

Chapter adapted from:

van Noord, R. and Bos, J. (2017b). The Meaning Factory at SemEval-2017 task 9: Producing AMRs with neural semantic parsing. In *SemEval*, pages 929–933, Vancouver, Canada

van Noord, R. and Bos, J. (2017c). Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108

4.1 Introduction

Various approaches to open-domain semantic parsing have been proposed in the last years. What we now could refer to as “traditional” approaches are semantic parsers that use supervised learning to create a syntactic analysis on which the meaning representations are constructed, usually in a compositional way. Research in this area comprises Bos et al. (2004), Copestake et al. (2005), Butler (2010), Le and Zuidema (2012), Lewis and Steedman (2013), Bos (2015), Artzi et al. (2015), and many others. Efforts to create data sets of sentences paired with meaning representations have stimulated research in semantic parsing (Banarescu et al., 2013; Bos et al., 2017), especially those using the formalism of Abstract Meaning Representation (AMR), for which also shared tasks have been organized (May, 2016). In this chapter, therefore, we concentrate on semantic parsing of AMRs, because large gold-standard data sets are available and various different approaches can be compared.

In contrast to the traditional approaches mentioned above, there have been interesting attempts recently to view semantic parsing as a translation task, mapping English expressions to logical forms under supervision of some deep learning method. Dong and Lapata (2016) used sequence-to-sequence (seq2seq) and sequence-to-tree (seq2tree) neural translation models to produce logical forms from sentences for four different data sets (but not AMRs). Barzdins and Gosko (2016) used a similar method to produce AMRs in the context of the previously mentioned shared task, but the performance of their neural parser was still far below the state-of-the-art. Despite this, their method inspired other researchers to adopt this seq2seq approach (Peng et al., 2017b; Konstas et al., 2017). But, even though they got substantial improvements over Barzdins and Gosko (2016), their systems still did not come close to state-of-the-art. The neural approach of Folland and Martin (2017) did reach state-of-the-art performance, but they used five bi-LSTM networks instead of a single seq2seq model.

What all these attempts have in common, and why they are fascinating, is that they completely avoid complex models of the syntactic and semantic parsing process and therefore do not rely on heavily engineered features. However, except for Barzdins and Gosko (2016), they also only

use word-level input. This is interesting, because Barzdins and Gosko (2016) obtained a substantial improvement for their character-level model over their word-level model. Character-embeddings, since they were introduced by Sutskever et al. (2011), have also shown improvements in a number of areas, such as POS-tagging (Santos and Zadrozny, 2014; Plank et al., 2016), text classification (Zhang et al., 2015), and, most importantly, Neural Machine Translation (Chung et al., 2016b).

The aim of this chapter is to find out how far we can push character-level neural semantic parsing: can we reach accuracy scores comparable with traditional approaches to semantic parsing? More specifically, our objectives are (1) reproducing the results of Barzdins and Gosko (2016); (2) improving on their results by employing several novel techniques; and (3) investigating whether adding silver standard data can improve neural semantic parsing.

We make three main contributions. First, we show that character-level models can obtain good performance on neural semantic parsing. Second, we introduce a number of novel techniques to improve performance. Third, we show that adding silver standard to the training data makes a considerable (positive) difference for neural AMR parsing. Our final model reaches an F-score of 71.0, which is the current state-of-the-art in AMR parsing.¹

4.2 Method and Data

We first give a bit of background on AMRs. Then we outline the basic ideas of the character-based translation model with English sentences as input and AMRs as output. We then establish a baseline system with the aim to improve it in the next section.

4.2.1 Abstract Meaning Representations

In our experiments utilizing neural semantic parsing we will focus on parsing Abstract Meaning Representations (AMRs). AMRs were introduced by Banarescu et al. (2013) and are acyclic, directed graphs that represent the

¹This was at the time of writing (May 2017). Current state-of-the-art scores and methods (as of August 2020) are discussed in Chapter 9.

meaning of a sentence. There are, in fact, three ways to display an AMR: as a graph, as a set of triples, or as a tree in string format. An example of an AMR is shown in Figure 4.1, here displayed as a tree, the format that is used in the annotated corpora.

```
(a / affect-01
  :ARG0 (w / wave-04
    :ARG1 (h2 / heat)
    :location (c / country
      :wiki "France"
      :name (n / name
        :op1 "France"))))
  :ARG1 (p / person
    :ARG0-of (s / strike-02
      :mod (h / hunger-01
        :ARG0 p))))
```

Figure 4.1: AMR representing the meaning of *Hunger strikers were affected by France’s heat wave*.

An AMR consists of concepts that are linked to variable names with a slash. In the example above we have that *a* is an instance of the concept *affect-01*, and *p* is an instance of the concept *person*. Note that the names of the variables are not important. Concepts can be related to each other by using two-place relations, which are indicated by a colon. So, the first *:ARG0* is an ordered relation between *a* and *w*. Inverse relations are denoted by the suffix *-of*. Note that, if one concept relates to more than one other concept (for instance, in the example above, the node *a* is related to *w* via *:ARG0*, and to *p* via *:ARG1*), the order of these relations within the AMR is not important. AMRs also allow for a re-occurrence of variables: the concept *person* with variable *p* stands in a relation with *affect-01* as well as with *hunger-01*. For a more extensive overview of AMR, see Section 2.1.1.

For evaluation purposes, AMRs are converted into triples. The accuracy of an AMR parser is computed by precision and recall on matching triples between gold standard AMRs and system-produced AMRs, using the *Smatch* system (Cai and Knight, 2013). For the evaluation of our experiments we

use the sentences annotated with AMRs from LDC release LDC2017T10², consisting of 36,521 training AMRs, 1,368 development AMRs and 1,371 test AMRs. This release also includes the PropBank frameset and comes with pre-aligned AMRs and sentences. In all results shown in this chapter, the models are trained on the training data. As development and test data we use the designated dev and test set from LDC2017T10, which are the exact same sets that are used in LDC2015E89. We remove HTML-tags from the input sentences, but URLs are kept in.

Setting	Value	Setting	Value
Layers	2	RNN type	brnn
Nodes	500	Dropout	0.3
Epochs	20–25	Vocabulary	100–200
Optimizer	sgd	Max length	750
Learning rate	0.1	Beam size	5
Decay	0.7	Replace unk	true

Table 4.1: Hyper-parameter settings of the seq2seq model.

4.2.2 The Basic Translation Model

To create our sequence-to-sequence translation model, we use OpenNMT (Klein et al., 2017). We train a model with bidirectional encoding and general attention (Luong et al., 2015). This model is described in detail in see Section 3.3. Since training a full model takes two to three days on a GPU, we perform a heuristic hyper-parameter search instead of an exhaustive one. We started out with a default model and changed only one hyper-parameter value in separate experiments. If we improved over the default, the setting was kept and combined with other hyper-parameter settings that improved performance. All models were only tested on the development set. Ultimately, we arrived at the settings shown in Table 4.1. All our described models in this chapter are trained with these settings. Training is stopped 3 epochs after there is no improvement in validation perplexity on the development set anymore. The best performing model on the development set is then used to decode the test set.

²<https://catalog.ldc.upenn.edu/LDC2017T10>

In contrast to Peng et al. (2017b) and Konstas et al. (2017), who use word-level input, we use character-level input. We did experiment with word-based models, but they never obtained F-scores higher than 30.0, even when using the GloVe pretrained word embeddings (Pennington et al., 2014). This is in line with Peng et al. (2017b) and Konstas et al. (2017), who only arrived at their final F-scores by applying extensive anonymization methods. These methods are heavily rule-based, so we prefer not to be dependent on them. We will compare our scores to their word-based models in Section 4.4.

<pre>(m / material :mod (r / raw) :domain (o / opium) :ARG1-of (u / use-01 :ARG2 (p / make-01 :ARG1 (h / heroin) :ARG2 o)))</pre>	<pre>(material :mod (raw) :domain (opium) :ARG1-of (use-01 :ARG2 (make-01 :ARG1 (heroin) :ARG2 (opium))))</pre>
---	---

Figure 4.2: Example of the original AMR (left) and the variable-free AMR (right) displaying the meaning of *Opium is the raw material used to make heroin*.

Following Barzdins and Gosko (2016), we do not want our model to learn the arbitrary characters that are used to represent variables. The characters itself do not carry any semantic information and are only necessary to indicate co-referring nodes. Therefore we remove all variables from the AMRs and simply duplicate co-referring nodes from the input. An example of such a preprocessed AMR is shown in Figure 4.2. Note that this means that we lose information, since the variables cannot be put back perfectly. We describe an approach to restore the co-referring nodes in the output in the next section. In Chapter 5, we will describe a more principled approach of handling coreference. All wikification relations present in AMRs in the training set are also removed and restored in a postprocessing step. Newlines present in an AMR are replaced by spaces, and multiple spaces are squeezed into single ones (so the input AMR is represented on a single line).³

³Pre- and postprocessing scripts available at <https://github.com/RikVN/AMR>

4.2.3 Postprocessing and Restoring Information

The output of the seq2seq model is, of course, an AMR without variables, without wiki-links, and without co-occurrent variables. Furthermore, because of the character-based seq2seq model, it could well be that there are brackets in the output that do not match, or that some nodes representing concepts are incomplete. This, obviously, needs to be fixed. First, the variables in the AMRs are restored by assigning a unique variable to each concept. We also try to fix invalidly produced AMRs by applying a few heuristics, such as inserting parentheses and quotes, or by removing unfinished nodes. This is done by using the restoring script from Barzdins and Gosko (2016).⁴ Then, we apply three methods to increase the quality of the AMRs. They are described below.

Model	# Nodes pruned	# AMRs changed	F-score
Baseline	0	0	54.8
Removing all re-occurrent nodes (RCNs)	1426	689	55.4
Removing all RCNs with same parent	135	95	55.0
Removing all RCNs with frequency >2	427	249	55.3
Removing all RCNs with same parent, but also nodes with frequency >2	496	302	55.5

Table 4.2: Statistics of the different pruning methods. Methods were applied on the output of our baseline model on the dev set.

Pruning

A problem with our model is the fact that the decoder does not explicitly keep track of what it has already produced. As a consequence, we sometimes end up with duplicated, redundant material in our generated AMRs. This hurts precision. We propose four different methods to remove this redundant material. This is done on node level, where nodes are defined as relation-concept pairs without children, e.g., `:mod (raw)` and `:domain (opium)`. The statistics of applying these four methods on our baseline model (dev set) are shown in Table 4.2. Note that all these processes are trade-offs: usually

⁴Taken from <https://github.com/didzis/tensorflowAMR/>

duplicates are correctly recognized as redundant and can be removed, but sometimes we erroneously remove actual re-occurent nodes.

The first method simply removes all re-occurent nodes and is already quite effective: the F-score increases by 0.6. The second method is more careful and only removes duplicate nodes if they have the same parent. This helps, but only by a small margin. The third method does not consider parent nodes, but removes nodes if they occur more than twice in the full AMR. This method also increases the F-score, but does not outperform the first method yet. The fourth method is a combination of the second and third method. All re-occurent nodes with the same parent are removed, but also nodes occurring more than twice are removed. This results in the best F-score, an increase of 0.7 over the baseline. Two example AMRs whose branches are pruned using the fourth method are shown in Figure 4.3.

<pre>(material :mod (raw) :mod (raw) :domain (opium) :ARG1-of (use-01 :ARG2 (make-01 :ARG1 (heroin) :ARG2 (opium))))</pre>	<pre>(material :mod (raw) :domain (opium :mod (raw)) :ARG1-of (use-01 :ARG2 (make-01 :ARG1 (heroin) :mod (raw) :ARG2 (opium))))</pre>
---	--

Figure 4.3: Example of pruned branches for the produced AMRs of *Opium is the raw material used to make heroin*. In the left AMR, the second occurrence of `:mod (raw)` is already removed, because both branches are children of `material`. However, in the right AMR, none of the `:mod (raw)` branches share the same parent, so only the third occurrence is removed.

Wikification

Since we removed wikification relations in preprocessing, our model will never output such a link. We restore wiki links in the output AMR by using an off-the-shelf system (Daiber et al., 2013), following the method presented by Bjerva et al. (2016a). They look at the `:name` relations in an AMR and try to find this name on Wikipedia. If it has a page, the corresponding link gets added; otherwise the AMR remains unaltered.

Restoring co-referring nodes

Our system also tries to restore co-referring nodes. If we output a duplicate node (a node already produced for this AMR), it replaces the node by the variable name of the node encountered first. This can only happen once per unique node, since the third instance of such a node is already removed in the pruning phase. An example of how the co-referring nodes are restored is shown in Figure 4.4.

<pre>(m / material :mod (r / raw) :domain (o / opium :mod (r2 / raw)) :ARG1-of (u / use-01 :ARG2 (m2 / make-01 :ARG1 (h / heroin) :ARG2 (o2 / opium))))</pre>	<pre>(m / material :mod (r / raw) :domain (o / opium :mod r) :ARG1-of (u / use-01 :ARG2 (m2 / make-01 :ARG1 (h / heroin) :ARG2 o))))</pre>
---	--

Figure 4.4: Example of how co-referring nodes are restored. On the left an example of a produced AMR, on the right the AMR with coreference restored.

4.2.4 Baseline Results

Our first objective was to reproduce the results obtained by Barzdins and Gosko (2016). We did so, arriving at an F-score of 53.1 (see Table 4.3). Compared to the F-score of 43.0 by Barzdins and Gosko (2016), our score is significantly higher. This is probably due to the higher amount of training data and the fact that they used Tensorflow instead of OpenNMT. We also reproduced their results by using the exact same data, software and hyper-parameter settings as they did, obtaining an F1-score of 42.3.⁵

As is shown in Table 4.3, concept pruning, restoring coreference variables, and wikification all increase the F-score by about a percentage point each. This small gain of performance is what one could expect as each single operation has only a small impact on the overall contents of an AMR.

⁵We did not possess their wikification and coreference restoring scripts, so differences might be attributed to that.

	Type	Dev	Diff	Test	Diff
Baseline	seq2seq	54.8		53.1	
Postprocessing	Pruning	55.5	+ 0.7	53.7	+ 0.6
	Restoring coref	55.7	+ 0.9	54.2	+ 1.1
	Wikification	55.8	+ 1.0	54.1	+ 1.0
All postprocessing		57.3	+2.5	55.5	+ 2.4

Table 4.3: Results of our baseline seq2seq AMR parsing model, with for each postprocessing method its respective improvement.

4.3 Improving the Basic Translation Model

In the previous section we outlined our basic method of producing AMRs using a seq2seq model based on characters. In this section, we look at five different techniques to move beyond the F-score that we obtain with our basic method, that we will consider in this section as a baseline.

4.3.1 AMR Re-ordering

Although AMRs are unordered by definition, in our textual representation of the AMRs there is an order of the branches. However, these branches do not necessarily follow the word order in the corresponding English sentence. It has been shown that for (statistical) machine translation reordering improves translation quality (Collins et al., 2005). We use the provided alignments to permute the AMR in such a way that it best matches the word order. We do this both on sub-tree level and on individual node level. The best matching AMR is defined as the AMR in which the order of the nodes (when traversing over the AMR depth-first) is the closest to the order of the words in the English sentence, following the alignments. An example of an AMR with a branch order best matching the input sentence is shown in Figure 4.5.

We are also able to use this approach to augment the training data, since each reordering of the AMR provides us with a new AMR-sentence pair. Due to the exponential increase, large AMRs often have thousands of possible orders. We performed a number of experiments to find out how we could best

<pre>(material :mod (raw) :domain (opium) :ARG1-of (use-01 :ARG2 (make-01 :ARG1 (heroin) :ARG2 (opium))))</pre>	<pre>(material :domain (opium) :mod (raw) :ARG1-of (use-01 :ARG2 (make-01 :ARG2 (opium) :ARG1 (heroin))))</pre>
---	---

Figure 4.5: Example of a variable-free AMR before (left) and after re-ordering (right) for the sentence *Opium is the raw material used to make heroin*.

exploit this surplus of data. Ultimately, we found that it is most beneficial to “double” the training data by adding the best matching AMR to the existing data set.

Negative results Instead of ordering the AMR nodes reflected by the word order of sentence, we also tried two different experiments based on consistency. The first experiment simply ordered the nodes alphabetically, without any other influence. This decreased the result of our baseline model by 2.0. Our second experiment was focused on fixing irregularities: if two nodes occur in a different order than they usually do (based on the full training set), we simply switch them around. This method did not change the order as considerably as the alphabetical ordering, but the result of the baseline model still decreased by 1.0. Hence we discarded both reordering techniques.

4.3.2 Introducing Super Characters

We are not necessarily restricted to *only* using characters as input. For example, we can view the AMR relations (e.g., :ARG0, :mod) as atomic instead of a set of characters. This ensures that the characters for relations (e.g., :, *m*, *o* and *d* for :mod) do not influence the general character embeddings of the concepts, which might improve performance. This way, we create a hybrid model that is a combination of word and character level input at the target side. An example of the AMR and sentence level input using super characters is shown in Figure 4.6 and Figure 4.7.

AMR, chars:	(t h i n g + : q u a n t + 1 + : p o l a r i t y + -)
AMR, super chars:	(t h i n g +:quant + 1 +:polarity + -)

Figure 4.6: Input for the AMR (t / thing :quant 1 :polarity -) representing the sentence *Not one thing*, with and without super characters. The +-symbols represent spaces.

sentence:	I + a m + n o t + t h a t + r i c h + .
sentence + POS:	I PRP + a m VBP + n o t RB + t h a t IN + r i c h JJ + .

Figure 4.7: Input for the sentence *I am not that rich*, without and with POS-tags. POS-tags are inserted as super characters. The +-symbols represent spaces (word boundaries).

Negative result We also tried various ways to explicitly encode the tree structure by using super characters. In our basic model, the parentheses ‘(’ and ‘)’ are simply characters. This means that the model cannot differentiate between a parenthesis that opens the full AMR and a parenthesis that opens, say, the fifth subtree of the AMR. One would expect it would help the model if it has this information explicitly encoded in the input. For example, in an experiment we replaced each parenthesis in the structure by a super character that also provides the subtree information (e.g., an opening parenthesis on the fifth level becomes *5*(, while a closing bracket on the third level becomes *3*). However this resulted in an F-score lower than the baseline and we discarded the technique.

4.3.3 Adding Part-of-Speech Information

We might still be able to benefit from syntactic information, even though we use a character-level neural semantic parser. To show this, we parse the sentences with the POS-tagger of the C&C tools (Clark et al., 2003), employing the Penn POS tagset. Each tag is represented as a single character and placed after the last character representation of the word that matches the tag (see Figure 4.7). Put differently, we create a new super character for each unique tag and add this to the input sentence. On the one hand, this will in-

crease the size of the input. On the other hand, just a single character will add a lot of general, potentially useful, information. For example, proper nouns correlate with the `:name` relation, while adjectives correlate with the `:mod` relation. Note that this makes us reliant on an external resource (a POS-tagger), which is what we intended to avoid. However, it is still interesting to know how much our model can benefit from such a resource.

4.3.4 Adding Silver Standard Data

A problem with semantic parsing approaches is data sparsity, since a lot of manual effort is required to create gold standard data. This is especially a problem for data-hungry neural models. Peng et al. (2017b) tried to overcome this by extensive generalization of the training data, but did not get near state-of-the-art results. Konstas et al. (2017) applied a similar method, but also used the GigaWord corpus to self-train their system. They use their own pre-trained parser to parse the previously unseen sentences and add those to the training data in a series of iterations. Ultimately, their system is trained on 20 million additional data AMR-sentence pairs and obtains an F-score of 62.1. Without this additional data, they obtain a score of 55.5, which is better than Peng et al. (2017b), but not close to state-of-the-art performance.

Our method of obtaining new training data mainly differs from Konstas et al. (2017) in two ways: (i) we use two off-the-shelf parsers to create the training data instead of self-training; (ii) we employ a method to exclude lower-quality AMRs instead of using all available data. We therefore refer to this data as *silver standard* data, by which we mean something in between unchecked automatically produced data and gold standard data.

Instead of self-training our parser, we use the off-the-shelf AMR parsers CAMR (Wang et al., 2015b) and JAMR (Flanigan et al., 2014) to create silver standard data for our system. Both are non-neural, syntax-based parsers. CAMR works by first generating a dependency tree for the English sentence, after which it uses a transition-based algorithm to create the AMR graph. JAMR is the first published AMR parser and does the parsing in two stages: first identifying the concepts by using a semi-Markov model, and then identifying the relations between these concepts by searching for the maximum spanning connected subgraph.

Both systems are trained on the LDC2015E86 AMR corpus, which contains 16,833 training instances. We parse 1,303,419 sentences from the Groningen Meaning Bank (Basile et al., 2012a), which mainly consists of newswire text. AMRs that are either invalid or include `null-tag` or `null-edge` (this is what the CAMR parser outputs when it is not able to find a suitable candidate parse) are removed. We do not simply add the other AMRs to our data set. To ensure that the AMRs are at least of decent quality, we compare the produced AMRs with each other using *Smatch* (Cai and Knight, 2013). If their pairwise score does not exceed 55.0, the AMRs are not considered for adding to our training set. This value was picked to filter out AMRs that would only hurt the training process, but to also still include a large variety of AMRs and sentences.⁶ Our final set contained 530,450 sentences, that have both a CAMR and JAMR parse.

We now have to determine which AMR to add to our silver data set. CAMR produces higher quality AMRs in general (64.0 vs 55.0 on the test set), but it might be beneficial to introduce some variety by also adding JAMR-parsed AMRs. We never add both CAMR and JAMR for the same sentence. We performed five experiments in which we added 100k silver AMRs, either containing 100%, 75%, 67%, 50% or 0% CAMR-parsed AMRs. The results of testing on the development set are shown in Table 4.4.

# CAMR AMRs	# JAMR AMRs	F-score
100,000	0	65.8
75,000	25,000	65.8
66,667	33,333	65.7
50,000	50,000	65.3
0	100,000	61.4

Table 4.4: F-scores on the dev set for adding different ratios of CAMR and JAMR parsed AMRs to our initial data set. All scores are without postprocessing improvement methods.

⁶We did not experiment with this value, but it did outperform adding CAMR or JAMR produced AMRs randomly (i.e., not having a minimal score).

As would be expected, we see that only adding CAMR scores is considerably better than only adding JAMR. However, the scores for adding 67% and 75% CAMR are very similar to adding 100% CAMR. But, since this does not indicate that adding JAMR actually helps performance, we only add the CAMR-parsed AMRs in our silver data experiments. We randomly selected 20k, 50k, 75k, 100k and 500k instances for these experiments.

4.3.5 Optimizing Training

Aside from the pre- and postprocessing methods described, we can also optimize the training process itself. The first method we employ is pre-training on our full data set including silver AMRs, after which the model is fine-tuned on the gold data only. Both phases use the same hyper-parameter settings. We experimented with different learning rates for the fine-tuning phase, but this resulted in lower performance. A similar procedure was used by Konstas et al. (2017) and in general this is a method widely used in Neural Machine Translation (Denkowski and Neubig, 2017).

The second method is averaging a set of models to decode the test set, instead of using a single model. This was first applied by Junczys-Dowmunt et al. (2016) as an alternative to the usual ensembling of models, which is known to give substantial improvements in Neural Machine Translation (Sutskever et al., 2014). Ensembling, however, is very resource intensive, since the predictions of different models are averaged at decoding time. This as opposed to averaging, where the parameters of models are averaged to create a single model. This means that averaging, say, four models is four times faster than ensembling four models, while also using only a quarter of the memory the ensemble method uses. We tested with both ensembling and averaging and obtained similar results on the development set, thus opting to only use averaging in our experiments.

4.4 Results and Discussion

Table 4.5 shows the results of our improvement methods in isolation, meaning that only that individual method is added to our baseline model. Re-ordering has a clear positive effect, both for using the best re-ordering (+2.0) and adding that re-ordering to the existing data set (+5.2). Constructing super characters and adding POS-tags both lead to a similar increase in performance. Pre-training and subsequently fine-tuning also results in a substantial improvement, but creating an average model only has a slight positive effect. The biggest improvement comes from adding silver standard data to our training set, reaching a maximum of 65.8 on the dev set. However, there is a limit with regards to adding silver data, since adding 500k silver AMRs performed worse than adding 50k, 75k or 100k silver AMRs. Finding the optimal number of silver AMRs is difficult due to the long training times and is therefore left for future work.

	Type	Dev	Diff	Test	Diff
Baseline	seq2seq	54.8		53.1	
AMR Re-ordering	Best	56.8	+ 2.0	55.1	+ 2.0
	Doubling	60.0	+ 5.2	58.3	+ 5.2
Super Characters	Relations	58.3	+ 3.5	57.4	+ 4.3
POS Tags	PTB	58.2	+ 3.4	57.2	+ 4.1
Training optimization	Averaging	54.9	+ 0.1	53.4	+ 0.3
	Pre-training	59.4	+ 4.6	58.6	+ 5.5
	Both	59.5	+ 4.7	58.7	+ 5.6
Silver Standard Data	Adding 20k	62.2	+ 7.4	60.0	+ 6.9
	Adding 50k	64.7	+ 9.9	62.9	+ 9.8
	Adding 75k	65.7	+ 10.9	63.7	+ 10.6
	Adding 100k	65.8	+ 11.0	64.0	+ 10.9
	Adding 500k	63.8	+ 9.0	62.1	+ 9.0

Table 4.5: Results of the improvements in isolation (without postprocessing).

Since the previous experiments were all in isolation, we now test whether a combination of our methods still increases performance. The tested combinations are shown in Table 4.6. Even after adding the silver data, the addition of POS-tags and super characters still increased the performance, albeit by a smaller margin. Interestingly, the best result (71.0) was not obtained by combining all improvement methods, since re-ordering the AMRs does not show an increase anymore after adding POS-tags and super characters. The best model without using any silver data obtains an F-score of 64.0, which is considerably higher than the AMR-only score (55.5) of Konstas et al. (2017).

Post-proc	+ 100k Silver	POS Tags	Super Chars	Best Order	Optimize Training	Dev	Test
✗	✗	✗	✗	✗	✗	54.8	53.1
✓	✗	✗	✗	✗	✗	57.3	55.5
✓	✗	✓	✓	✓	✓	65.2	64.0
✓	✓	✗	✗	✗	✗	68.0	66.4
✓	✓	✓	✗	✗	✗	68.9	67.3
✓	✓	✓	✓	✗	✗	70.4	69.0
✓	✓	✓	✓	✓	✗	69.0	68.0
✓	✓	✓	✓	✗	✓	71.9	71.0

Table 4.6: F-scores for our neural models, combining the different improvement methods.

Table 4.7 shows the results of the most notable previous AMR parsing systems. Our best model outperforms all these previous parsers and reaches state-of-the-art results. However, we are also the first approach that uses the LDC2017T10 data set, which contains slightly more than double the number of gold standard training instances compared to the LDC2015E86 data set.⁷ Therefore, we also trained the best performing model in Table 4.6 on the LDC2015E86 data set, while still applying all our improvement methods. This model still obtains an F-score of 68.5, outperforming all previous AMR parsers, except for the parser of Folland and Martin (2017).

⁷LDC2015E86 only contains 16,833 instances, as opposed to the 36,521 of LDC2017T10.

Authors	Model	Train set	F1
Flanigan et al. (2014)	JAMR-14	LDC2013	58.0
Damonte et al. (2017)	AMR-eager	LDC2015	64.0
Artzi et al. (2015)	CCG parsing	LDC2014	66.3
Wang et al. (2015b)	CAMR	LDC2015	66.5
Flanigan et al. (2016)	JAMR-16	LDC2015	67.0
Pust et al. (2015)	SBMT	LDC2015	67.1
Barzdins and Gosko (2016)	seq2seq (char)	LDC2015	43.0
Peng et al. (2017b)	seq2seq (word)	LDC2015	52.0
Konstas et al. (2017)	seq2seq (word)	LDC2015	55.5
Konstas et al. (2017)	seq2seq (word) + giga	LDC2015	62.1
Foland and Martin (2017)	5 bi-LSTMs (word)	LDC2015	70.7
This chapter	seq2seq (char) + silver	LDC2015	68.5
This chapter	seq2seq (char) + silver	LDC2017	71.0

Table 4.7: F-scores for AMR parsing. Comparison with previously published results on the test set.

Damonte et al. (2017) presented a way to evaluate system output in a more detailed way, by focusing on various aspects that are present in an AMR: role labelling, word sense disambiguation, named entity recognition, wikification, detecting negation, and so on. These detailed results of our best system are shown in Table 4.8, in which the results of the other parsers are taken from Damonte et al. (2017). Unfortunately, Foland and Martin (2017) did not publish these specific scores. As the table shows, our system scores higher than the other parsers on five of the eight metrics other than *Smatch*. In general, our system is quite conservative, obtaining a higher precision than recall for each metric. Given the results in Table 4.8, one would think that detecting negation and re-entrancy would be ways to get an improvement in accuracy. Note that the other parsers also score relatively bad at these metrics. Compared to the other systems, our system scores worse on concepts, named entities, and wikification. A possible method to increase performance in the first two of those metrics is to adopt an anonymization or generalization approach for named entities and concepts, similar to Peng et al. (2017b) or Konstas et al. (2017).

	CAMR	JAMR-16	AMR-eager	Our system		
Metric	F	F	F	Pr.	Rec.	F
Smatch	63	67	64	76	67	71
Unlabeled	69	69	69	79	70	74
No WSD	64	68	65	76	67	72
Re-entrancy	41	42	41	57	48	52
Concepts	80	83	83	87	78	82
Named entities	75	79	83	83	76	79
Wikification	0	75	64	82	54	65
Negations	18	45	48	67	58	62
SRL	60	60	56	70	62	66

Table 4.8: Comparison with previous parsers using the evaluation suite of Damonte et al. (2017). Our system also includes precision and recall scores.

4.5 Conclusions and Future Work

This chapter introduced a character-based neural sequence-to-sequence model for AMR parsing. It outperformed previous word-based sequence-to-sequence models, without being dependent on pretrained word embeddings or extensive anonymization techniques. Applying re-ordering of AMR branches, introducing super characters, and adding POS-tags are techniques that substantially improved performance. However, the biggest increase of performance is triggered by adding a large quantity of silver standard AMRs produced by existing (traditional) parsers. This is in line with the findings of Konstas et al. (2017), who used the Gigaword corpus to get extra training data, although their training method is different from ours.

The obtained results are promising. Our best model, with an F-score of 71.0, outperformed any known previously published result on AMR parsing. This is remarkable, for traditional approaches are often based on extensive, manually crafted lexicons using linguistic knowledge. It should be noted, of course, that we use some linguistic knowledge in the form of POS-tags in our best models, and that we employ existing parsers trained on extensive linguistic annotations. In fact, one could consider the use of silver standard AMR data as a disadvantage, as there is still a need of an existing high-quality

AMR parser to get the silver data in the first place. In our approach we rely even on two different off-the-shelf parsers. It would therefore be interesting to explore other opportunities, such as self-learning (Konstas et al., 2017).

We have the feeling that there are still a lot of techniques that one could try to increase the performance of neural AMR parsing. From a more esthetical perspective, it would be nice if one could eliminate the AMR repair strategies that are used to resolve unbalanced brackets. An interesting candidate that could master this problem would be the seq2tree model (Dong and Lapata, 2016). Similarly, a more principled approach to deal with co-occurring variables would be desirable. We will experiment with this in Chapter 5.

Another possible next step in semantic parsing is to change the target meaning representation. AMRs are unscoped meaning representations, and have no quantifiers. It would be challenging to transfer the techniques of neural semantic parsing to scoped meaning representations, such as those used in the Groningen Meaning Bank (Basile et al., 2012a) or the Parallel Meaning Bank (Abzianidze et al., 2017). The experiments in Chapters 7 and 8 will focus on the latter.

CHAPTER 5

Dealing with Coreference in Neural AMR Parsing

Linguistic phenomena like pronouns, control constructions, or coreference give rise to co-indexed variables in meaning representations. This is known to be a challenging problem for neural semantic parsers, as the variable names themselves are arbitrary and meaningless. We review three different methods for dealing with co-indexed variables for our character-level sequence-to-sequence model trained on abstract meaning representations: (a) copying concepts during training and restoring co-indexation in a post-processing step; (b) explicit indexing of co-indexation; and (c) using absolute paths to designate co-indexing. The second method gives the best results, outperforming our baseline model by 2.9 and our best model by 1.0 F-score points. These findings shed some light on the impact of different methods in handling co-indexed variables in neural semantic parsing.

Chapter adapted from:

van Noord, R. and Bos, J. (2017a). Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 41–49, Montpellier, France

5.1 Introduction

Semantic parsing is the task of mapping a natural language sentence into a meaning representation (a logical form). One of the problems a semantic parser has to deal with is co-indexed variables, which arise in antecedent-anaphor relations, proper name coreference, control constructions and other linguistic phenomena. Examples of such constructions are given in (1)–(4):

- (1) Bob_{*i*} likes himself_{*i*}.
- (2) Jack_{*i*} wants ϵ_i to buy an ice-cream.
- (3) Peter_{*i*} sold his_{*i*} car.
- (4) Sue_{*i*} saw Mary_{*j*}, but Mary_{*j*} did not see Sue_{*i*}.

We represent meanings using the formalism of Abstract Meaning Representation (AMR), as introduced by Banarescu et al. (2013). AMRs can be seen as graphs connecting concepts by relations. Each concept is represented by a named instance. Coreference is established by re-using these instances. For example, the AMRs corresponding to examples (1) and (2) above are given in Figure 5.1. Note that, due to the bracketing, the variable *b* encapsulates the whole entity `person :name "Bob"` and not just `person`, i.e., *b* stands for a person with the name Bob.

```
(l / like-01                                (w / want-01
  :ARG0 (b / person                          :ARG1 (p / person
    :name "Bob")                             :name "Jack")
  :ARG1 b)                                   :ARG3 (b / buy-01
                                              :ARG1 p
                                              :ARG2 (i / ice-cream)))
```

Figure 5.1: AMRs representing the meaning of examples (1) and (2).

That there is a lot to gain in this area can be seen by applying the AMR evaluation suite of Damonte et al. (2017), which calculates nine different metrics to evaluate AMR parsing, *re-entrancy* being one of them. Out of the four systems that made these scores available (all scores reported in Chapter 4), the re-entrancy metric obtained the lowest F-score for three of them.

Various methods have been proposed to automatically parse AMRs, ranging from syntax-based approaches (Flanigan et al., 2014; Wang et al., 2015b; Pust et al., 2015; Damonte et al., 2017) to the more recent neural approaches (Peng et al., 2017b; Buys and Blunsom, 2017; Konstas et al., 2017; Folland and Martin, 2017). Especially the neural approaches are interesting, since they all use some sort of linearization method and therefore need a predefined way to handle re-entrancy. Peng et al. (2017b) and Buys and Blunsom (2017) use a special character to indicate re-entrancy and restore co-referring variables in a postprocessing step. Konstas et al. (2017) simply replace re-entrancy variables by their co-referring concept in the input and never output co-referring nodes. Folland and Martin (2017) use the same input transformation as Konstas et al. (2017), but do try to restore co-referring nodes by merging all equal concepts into a single concept in a postprocessing step. This is similar as we did in Chapter 4. All these methods have in common that they are not very sophisticated, but more importantly, that it is not clear what the exact impact of these methods is on the final performance of the model. This makes it unclear what the best implementation is for future neural semantic parsers.

In this chapter we present three methods to handle re-entrancy for AMR parsing. The first two methods are based on the previous work described above, while the third is a new, more principled method. These methods are applied on the character-level sequence-to-sequence model as described in Chapter 4. The aim of this chapter is to find the best method to handle re-entrancy in neural semantic parsing and to show the specific impact that each of the methods have on general performance.

5.2 Abstract Meaning Representations

AMRs, as introduced by Banarescu et al. (2013), are acyclic, directed graphs that show a representation of the meaning of a sentence. There are three ways to display an AMR: as a graph, as a tree, or as a set of triples. The AMRs in Figure 5.1 are shown as trees, which is also the input for our neural semantic parser. AMR concepts (e.g., like, person) are relating to each other by the use of two-place relations (e.g., :ARG0, :ARG1, :name). For example, in the right AMR in Figure 5.1, *want* and *buy* are connected by the :ARG3 relation. See Section 2.1.1 for a more extensive overview of AMR.

For our experiments, we use the annotated AMR corpus LDC2017T10, which contains 36,521 training, 1,368 development and 1,371 test AMRs. Co-indexed variables occur frequently in this data set, as can be seen in Table 5.1. About half of the AMRs contain at least one co-indexed variable, while about 20% of the total number of triples¹ contains a variable that has at least one anaphor in the AMR. This is the number of triples that is used in the re-entrancy evaluation metric by Damonte et al. (2017). This number, however, also includes triples that would be present regardless of whether the variable is co-indexed (e.g., the instance triple of the antecedent). A more fair number might be the relation triples that contain an anaphor variable, which is exactly equal to the number of total co-indexed variables. This is still 3.4 to 4.2% of all triples in the data set.

	# AMRs	# AMRs w/ re-entrancy	# re-entrancies	Total triples	Triples w/ co-index
Train	36,520	17,589	40,582	968,512	189,426
Dev	1,368	706	1,590	46,737	8,704
Test	1,371	749	2,033	48,252	9,686
Silver	100,00	16,235	18,865	3,001,169	109,676

Table 5.1: Statistics about the co-indexed variables in the AMR data set.

5.3 Method

5.3.1 Variable-free AMRs

The actual names of the variables for AMR instances are unimportant. Hence, one can rename variables in an AMR as long as re-occurrences of variables are preserved. If variables are used only once in an AMR, they can therefore be completely eliminated. This insight was used by Barzdins and Gosko (2016) in the first approach to neural semantic parsing of AMRs, because particular names of variables are very hard to learn for a neural model given the limited amount of data available. We present three ways to encode coreference in variable-free AMRs.²

¹Triples are used to evaluate AMR parsers, see Section 2.1.1 and Cai and Knight (2013).

²Pre- and postprocessing scripts available at <https://github.com/RikVN/AMR>.

Method 1A: Baseline Note that if there are co-indexed variables, there is always exactly one instance of the variable that carries semantic information. In our baseline method, similar to Barzdins and Gosko (2016) and Konstas et al. (2017), we simply copy this semantic information, while removing the variables, as is shown below. This method never outputs re-entrancy nodes and therefore functions as a baseline. An example is shown in Figure 5.2.

```
(like-01                                (want-01
  :ARG0 (person                          :ARG1 (person
    :name "Bob")                          :name "Jack")
  :ARG1 (person                          :ARG3 (buy-01
    :name "Bob"))                        :ARG1 (person
                                          :name "Jack")
                                          :ARG2 (ice-cream)))
```

Figure 5.2: Baseline representations of examples (1) and (2).

Method 1B: Re-entrancy Restoring This method is created to restore re-entrancy nodes in the output of the baseline model. It operates on a very ad hoc principle: if two nodes have the same concept, the second one was actually a reference to the first one. We therefore replace each node that has already occurred in the AMR by the variable of the antecedent node. This approach was applied in Chapter 4 and also by Foland and Martin (2017).

The model thus never *learns* to output re-entrancy, but the co-indexed variables are restored in a postprocessing step. An example is shown in Figure 5.3. Note that this process can also erroneously insert re-entrancies when two separate entities would be correct. For example, if the sentence *Bob_i likes Bob_j* refers to two different Bobs, the initial AMR in Figure 5.3 would have been correct.

<pre>(1 / like-01 :ARG0 (p / person :name "Bob") :ARG1 (p2 / person :name "Bob"))</pre>	<pre>(1 / like-01 :ARG0 (p / person :name "Bob") :ARG1 p)</pre>
---	--

Figure 5.3: Example of the Re-entrancy Restoring method for example (1). Initial AMR on the left, re-entrancy restored AMR on the right.

<pre>(like-01 :ARG0 (*1* person :name "Bob") :ARG1 (*1*)</pre>	<pre>(want-01 :ARG1 (*1* person :name "Jack") :ARG3 (buy-01 :ARG1 *1* :ARG2 (ice-cream)))</pre>
--	---

Figure 5.4: Representations of examples (1) and (2) when applying the Indexing method.

Method 2: Indexing This method comprises of removing all variables from an AMR, except when they are co-indexed. The remaining variables are normalized by converting them to numbers, so that each unique co-indexed variable has a unique identifier. Similar approaches were applied by Peng et al. (2017b) and Buys and Blunsom (2017), and an example is shown in Figure 5.4. In this approach the model actually learns where it should output re-entrancy nodes, instead of restoring them in a postprocessing step.

However, we do still need some postprocessing, as is shown in Figure 5.5 (after variables are restored). In this AMR, there is an index that is never instantiated (***3***) and an instantiated index that is never referred to (***2*** b / buy). The superfluous ***2*** can simply be removed, but for ***3*** we have multiple options. Algorithm 1 shows how we handle these cases.

For each referent that was never instantiated, we first check if there is also an instantiated index that is never referred to. If that it is the case, we replace it by that variable. This assumes that the model merely mismatched the index symbols. This works for ***3*** in Figure 5.5, even though the resulting node is still incorrect. If that was not the case, we try to replace the referent by an instantiated index that already did have a referent. Since this

index already has a referent, we assume it is likely that a mismatched referent actually referred to this index (in Figure 5.5, $*3*$ would then be replaced by p). In the case that both previous options failed, we simply pick the variable of the concept that is most often a referent in the training set.

<pre>(w / want-01 :ARG1 (*1* p / person :name "Jack") :ARG3 (*2* b / buy-01 :ARG1 (*1*) :ARG2 (i / ice-cream) :ARG3 (*3*)))</pre>	<pre>(w / want-01 :ARG1 (p / person :name "Jack") :ARG3 (b / buy-01 :ARG1 p :ARG2 (i / ice-cream) :ARG3 b))</pre>
---	---

Figure 5.5: Possible output of the model for example (2) on the left, while the fixed AMR after applying Algorithm 1 is shown on the right.

Algorithm 1 Postprocessing algorithm used to replace indexes by variables.

R: all referent indexes in output (e.g., $*1*$, $*3*$)

X: all instantiated indexes that do not have a referent (e.g. $*2*$ b / buy-01)

Y: all instantiated indexes that have a referent (e.g., $*1*$ person :name "Jack")

C: all concept-variable pairs (e.g., w / want-01)

most_freq: function that, given a list of concepts, selects the concept that is most frequently a referent in the training set

```
for all ref in R do
  if ref  $\subseteq$  X then
    replace ref by X(ref)
  else if X  $\neq \emptyset$  then
    replace ref by most_freq(X)
  else if Y  $\neq \emptyset$  then
    replace ref by most_freq(Y)
  else
    replace ref by most_freq(C)
  end if
end for
```

Method 3: Absolute Paths In this method re-entrancy is established by replacing a co-indexed variable by an absolute path that describes a position within an AMR. Absolute paths start from the top (root) of the AMR. The examples in Figure 5.6 show an example of these paths. In the first AMR, the path `:ARG0` describes that the node refers to the value of the relation `:ARG0`. In the second AMR, the path `:ARG1` refers in a similar way to `:ARG1`. Note that although these examples are straightforward, the paths can become quite long for larger AMRs, e.g., `:op1 :ARG0 :ARG0-of :mod`. The longest path in the training set even contains 14 relations.

```
(like-01                                (want-01
  :ARG0 (person                          :ARG1 (person
    :name "Bob")                          :name "Jack")
  :ARG1 { :ARG0 })                       :ARG3 (buy-01
                                              :ARG1 { :ARG1 }
                                              :ARG2 (ice-cream)))
```

Figure 5.6: Representations of examples (1) and (2) when applying the Absolute Paths method.

This method is perhaps the most attractive from a theoretical point of view. Note, however, that not every node in an AMR can be described by a unique path, as ambiguities might occur when there are two edges with the same name (this can occur, for instance, when more than one modifier is present). To solve these ambiguities an index (e.g., `|1|`, `|2|`) is added to each relation in the path.³ This was necessary for 632 out of the 40,582 paths in the training set. In total, there are 5,760 unique paths, of which 3,447 only occur once. The most frequent path is `:ARG0|1|`, occurring 5,405 times.

Similar to the Indexing method, a problem with this approach is the fact that we have no control over what the model will output. Especially for larger AMRs it is likely that the model will output impossible paths without destination. For example, the model might output `:ARG2` instead of `:ARG0` in the left example in Figure 5.6. These impossible paths still need to be re-

³For clarity, this was omitted in the example in Figure 5.6.

placed by a variable to get a valid AMR. The strategy we opt for here is similar to one used in the Indexing method, namely replacing the path by the variable of the concept in the AMR that most frequently has a referent in the training set.

5.3.2 Neural Model

We implement a bi-LSTM sequence-to-sequence model with general attention that takes characters as input, using the OpenNMT software (Klein et al., 2017). This is the same model as in Chapter 4, with the same hyper-parameter settings (see Table 4.1). It is trained for 20 epochs, after which the model that performs best on the development set is used to decode the test set.

5.3.3 Experiments

We test the impact of the different methods on two of our earlier models, described in Chapter 4. The first is a simple baseline model that only takes the characters into account without any additional methods to improve performance. This model is referred to as the char-only model. The second is the approach that produced one of the best results so far in the literature. This model uses POS-tagged input, clusters together groups of characters (super characters) and exploits 100,000 “silver” AMRs that were obtained by using the off-the-shelf AMR parsers CAMR (Wang et al., 2015b) and JAMR (Flanigan et al., 2014).⁴ The added AMRs are all CAMR-produced. We must note that CAMR is not particularly keen on outputting coreference, as the 100,000 silver AMRs only produced 18,865 new re-entrancy nodes.

The second approach also employs the postprocessing methods Wikification and pruning. The Wikification step simply adds wiki links to `:name` nodes, since those links were removed in the input. Pruning is used to remove erroneously produced duplicate output. This is a common problem for sequence-to-sequence models, since the model does not keep track of what it has already output. No pretraining or ensemble methods are used for both approaches, as opposed to the approach of Chapter 4.⁵

⁴Please see Chapter 4 for a full explanation of this process.

⁵This is because the initial papers for Chapter 4 and this chapter partly overlapped in development. We thought it best to stick to the originally reported scores.

5.4 Results and Discussion

5.4.1 Main Results

The results of applying our three methods on the baseline and best model are shown in Table 5.2. All reported numbers are F-scores obtained by using `Smatch` (Cai and Knight, 2013). All three methods offer an improvement over the baseline, for both the baseline and the best model. Indexing is the highest scoring method, except for the test set of the best model, since Re-entrancy Restoring obtains the same F-score there. Explicitly encoding the absolute paths resulted in an increase over the baseline, but did not outperform both Re-entrancy Restoring and Indexing, although only by a small margin.

	Char-only		Best model	
	Dev	Test	Dev	Test
Baseline	54.8	53.1	69.3	68.0
Re-entrancy Restoring	55.7	54.2	70.0	69.0
Indexing	55.9	56.0	70.5	69.0
Absolute Paths	54.9	53.9	70.3	68.7

Table 5.2: Results of the different methods in comparison to the char-only and best model.

	Char-only				Best model			
	Coref		No coref		Coref		No Coref	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Baseline	52.1	50.3	61.6	60.6	66.9	65.1	73.8	74.8
Restoring	53.4	52.0	61.2	60.8	68.3	66.8	73.8	74.8
Indexing	54.2	53.5	60.5	61.6	68.6	66.5	74.8	74.8
Absolute Paths	52.8	51.5	59.6	59.2	68.2	66.3	74.6	74.5

Table 5.3: Results for each method on AMRs with (coref) and without (no coref) re-entrancy nodes.

It is interesting to look at whether we indeed improve on parsing AMRs that have co-indexed variables, in comparison to AMRs that do not contain them. This is shown in Table 5.3. We see that, in general, we indeed only improve on both baselines for AMRs that have co-indexed variables. This is the case for both the dev and the test set. This is the desired scenario: we improve on AMRs with re-entrancy, while performance does not decrease on AMRs without re-entrancy. Only applying the Absolute Paths method on the baseline model scores lower on the test set on AMRs without re-entrancy nodes. Similar to the results in Table 5.2, Indexing outperforms Re-entrancy Restoring for the baseline model, but has similar scores when applied to the best model.

5.4.2 Detailed Analysis

Table 5.4 shows the how often the model outputs possible paths for the Absolute Paths method. Unfortunately, about 50% of the time, the model output a path that did not lead to a possible referent, leaving us to rely on the frequency heuristic. This problem is not as severe for our best model, though, since about 75% of paths were actually possible. The addition of extra (silver) data thus helped the model in learning the paths, suggesting that the baseline results might merely be an effect of sparse data. However, the very long paths still proved to be challenging for the best model. This might mean that, even when adding more gold data, learning such sophisticated structures is too difficult for end-to-end sequence-to-sequence models in general.

	Char-only				Best model			
	Dev		Test		Dev		Test	
	Pos	Imp	Pos	Imp	Pos	Imp	Pos	Imp
Total paths produced	455	464	534	533	636	200	783	328
Paths of length 1	160	30	148	32	231	12	193	13
Paths of length 2	211	202	265	194	268	68	398	108
Paths of length 3	76	156	106	210	130	80	167	123
Paths of length >3	8	76	15	97	7	40	25	84

Table 5.4: Number of possible (pos) and impossible (imp) paths in the output, when using the Absolute Paths method.

Table 5.5 shows more detailed results of the Indexing method. In the majority of cases, the model does what it is supposed to do: first instantiating an index, then referring to it. However, for finding the correct variable replacement for the baseline model, we still have to fall back on heuristics in approximately 30% of the cases. Most of these instances are then solved by referring to the concept that is most frequently a referent in the training set. However, for our best model, this reliance is not there anymore. The model generally only outputs an index when it was instantiated first, which is exactly what was intended.

	Char-only		Best model	
Replace index by	# Dev	# Test	# Dev	# Test
Instantiated variable	749	984	770	1,122
Instantiated variable that was never referred to	25	28	0	0
Most frequent instantiated index	8	8	3	9
Variable of most frequent concept	248	217	3	10

Table 5.5: Number of times each step in Algorithm 1 was responsible for replacing an index in the output when using the Indexing method.

5.5 Conclusions

We proposed three methods to handle co-indexed variables for neural semantic (AMR) parsing. The best results were obtained by the Indexing method, which explicitly encodes co-indexing nodes in the data set. The perhaps theoretically most attractive Absolute Paths performed the worst, although it did still offer an improvement over the baseline. Perhaps an interesting direction for future research is to use relative instead of absolute paths, encoding the path relative to the re-entrancy node instead of starting at the top of the AMR, because this will make the paths shorter and more local. This is especially interesting for parsing Discourse Representation Structures, as they contain more variables as well as more re-entrancies.

PART III

Discourse Representation Structures

CHAPTER 6

An Evaluation Method for DRS Parsing

The task of Abstract Meaning Representation (AMR) parsing has a clear and well-defined method of evaluation, but there is no such method for the task of Discourse Representation Structure (DRS) parsing. In this chapter, we construct a similar tool as for AMR parsing by translating the DRSs to a set of clauses, which we can then use to calculate a precision, recall and F1-score based on the number of matching clauses in DRS pairs. To be able to evaluate this tool and to be able to later train supervised DRS parsing systems, we use a semantically annotated parallel corpus for English, German, Italian, and Dutch where sentences are annotated with DRSs. The formalism captures, among others, the semantics of negation, modals, quantification, and presupposition triggers. Using this data set, we show that our evaluation tool is both accurate and efficient, yielding F-scores between 43 and 54 for three baseline semantic parsers. Moreover, as this is a parallel corpus, we are able to perform a pilot study to automatically find changes in meaning by comparing meaning representations of translations. This comparison turns out to be an additional way of (i) finding annotation mistakes and (ii) finding instances where our semantic analysis needs to be improved.

Chapter adapted from:

van Noord, R., Abzianidze, L., Haagsma, H., and Bos, J. (2018a). Evaluating scoped meaning representations. In *LREC*, pages 1685–1693, Paris, France. ELRA

6.1 Introduction

Semantic parsing is the task of assigning meaning representations to natural language expressions. The availability of open-domain, wide coverage semantic parsers has the potential to add new functionality, such as detecting contradictions, verifying translations, and getting more accurate search results. Current research on open-domain semantic parsing focuses on supervised learning methods, using large semantically annotated corpora as training data. A popular corpus is the Abstract Meaning Representation corpus (AMR; Banarescu et al. 2013), which we experimented with in the previous chapters. However, we are now interested in whether our model can handle meaning representations based on formal semantics, such as Discourse Representation Structures (DRSs, Kamp and Reyle, 1993). In this chapter, we lay the groundwork for this approach by (i) presenting an annotated corpus of DRSs and (ii) introducing an evaluation tool for semantic DRS parsers that is able to evaluate the quality of machine-generated meaning representations by comparing them to gold standard annotations

Our corpus with annotated DRSs is a *parallel* corpus annotated with formal meaning representations for English, Dutch, German, and Italian. Our work shows many similarities with recent annotation and parsing efforts around AMR, in that we abstract away from syntax, use first-order meaning representations, and use an adapted version of `Smatch` (Cai and Knight, 2013) for evaluation. However, we deviate from AMR on several points: meanings are represented by scoped meaning representations (arriving at a more linguistically motivated treatment of modals, negation, presupposition, and quantification), and the non-logical symbols that we use are grounded in WordNet (concepts) and VerbNet (thematic roles), rather than PropBank (Palmer et al., 2005). We also provide a syntactic analysis in the annotated corpus, in order to derive the semantic analyses in a compositional way. We make two main contributions:

- Presenting the first release of an annotated parallel corpus of formal meaning representations for four languages (Section 6.3).
- A tool that is able to compare scoped meaning representations for the purpose of evaluation (Section 6.4 and Section 6.5).

24/3221: No one can resist.

k0 NOT **b2**
b2 REF **x1**
b2 person n.01 **x1**
b2 POS **b3**
b3 Agent **e1** **x1**
b3 REF **e1**
b3 resist v.02 **e1**

00/2302: È tutto nuovo.

k0 IMP **b2** **b3**
b2 REF **x1**
b2 thing n.12 **x1**
b3 REF **s1**
b3 Theme **s1** **x1**
b3 new a.01 **s1**
b3 Time **s1** **t1**
b4 REF **t1**
b4 time n.08 **t1**
b4 EQU **t1** "now"

Figure 6.1: Two examples of a PMB document with their scoped meaning representations and the corresponding clausal form. These two structures are basic DRSs. The English translation of the bottom sentence is *Everything is new*.

6.2 Scoped Meaning Representations

In this section, we first describe how we convert the scoped meaning representations (DRSs) to a clausal form, which can more easily be used during evaluation. This is followed by a detailed comparison with AMR in terms of number of clauses and variables.

6.2.1 Discourse Representation Structures

The backbone of the meaning representations in our annotated corpus is formed by the Discourse Representation Structures (DRS) of Discourse Representation Theory (DRT, Kamp, 1984; Kamp and Reyle, 1993). Our version of DRS integrates WordNet senses (Fellbaum, 1998), adopts a neo-Davidsonian analysis of events by employing VerbNet roles (Bonial et al., 2011), and includes an extensive set of comparison operators. Presuppositions are explicitly represented (Van der Sandt, 1992; Venhuizen et al., 2018), while discourse structure is modeled by Segmented DRT (Asher, 1993; Asher and Lascarides, 2003). For a more detailed overview of DRT and DRS, see Section 2.1.2.

DRSs can be visualized in different ways. While the compact linear format saves space, the box notation increases readability. In this chapter we use the latter notation. The examples of DRSs in the box notation are presented in Figure 6.1. However, for evaluation and comparison purposes, we convert a DRS into a flat clausal form, i.e., a set of clauses. This is carried out by using the labels for DRSs as introduced in Venhuizen (2015) and Venhuizen et al. (2018), and breaking down the recursive structure of DRS by assigning them a label of the DRS in which they appear. Let t , t' , and t'' be meta-variables ranging over DRSs or terms. Let $\mathcal{C}$ be a set of WordNet concepts, $\mathcal{T}$ a set of the thematic roles, and $\mathcal{O}$ the set of DRS operators (REF, NOT, POS, NEC, EQU, NEQ, APX, LES, LEQ, TPR, TAB, IMP, DIS, PRP, DRS). The resulting clauses are then of the form $t \ R \ t'$ or $t \ R \ t' \ t''$ where $R \in \mathcal{C} \cup \mathcal{T} \cup \mathcal{O}$. The result of translating DRSs to sets of clauses is shown in Figure 6.1. In a clausal form, it is assumed that different variables are represented with different variable names and vice versa. Due to this, before translating a DRS to a clausal form, different discourse referents in the DRS must be represented with different variable names. This assumption significantly simplifies the matching process between clausal forms (Section 6.4) and makes it possible to recover the original box notation of a DRS from its clausal form.

00/3008: Hij speelde piano en zij zong.

<table><tr><td>x_1</td><td>x_2</td><td>e_1</td><td>t_1</td></tr><tr><td colspan="4">male.n.02(x_1)</td></tr><tr><td colspan="4">play.v.03(e_1)</td></tr><tr><td colspan="4">Time(e_1, t_1)</td></tr><tr><td>$k_1 ::$</td><td colspan="3">Theme(e_1, x_2)</td></tr><tr><td></td><td colspan="3">Agent(e_1, x_1)</td></tr><tr><td colspan="4">time.n.08(t_1)</td></tr><tr><td colspan="4">$t_1 < \text{now}$</td></tr><tr><td colspan="4">piano.n.01(x_2)</td></tr></table>	x_1	x_2	e_1	t_1	male.n.02(x_1)				play.v.03(e_1)				Time(e_1, t_1)				$k_1 ::$	Theme(e_1, x_2)				Agent(e_1, x_1)			time.n.08(t_1)				$t_1 < \text{now}$				piano.n.01(x_2)				<table><tr><td>x_3</td><td>e_2</td><td>t_2</td></tr><tr><td colspan="3">female.n.02(x_3)</td></tr><tr><td colspan="3">time.n.08(t_2)</td></tr><tr><td>$k_2 ::$</td><td colspan="2">$t_2 < \text{now}$</td></tr><tr><td></td><td colspan="2">sing.v.01(e_2)</td></tr><tr><td colspan="3">Time(e_2, t_2)</td></tr><tr><td colspan="3">Agent(e_2, x_3)</td></tr></table>	x_3	e_2	t_2	female.n.02(x_3)			time.n.08(t_2)			$k_2 ::$	$t_2 < \text{now}$			sing.v.01(e_2)		Time(e_2, t_2)			Agent(e_2, x_3)		
x_1	x_2	e_1	t_1																																																							
male.n.02(x_1)																																																										
play.v.03(e_1)																																																										
Time(e_1, t_1)																																																										
$k_1 ::$	Theme(e_1, x_2)																																																									
	Agent(e_1, x_1)																																																									
time.n.08(t_1)																																																										
$t_1 < \text{now}$																																																										
piano.n.01(x_2)																																																										
x_3	e_2	t_2																																																								
female.n.02(x_3)																																																										
time.n.08(t_2)																																																										
$k_2 ::$	$t_2 < \text{now}$																																																									
	sing.v.01(e_2)																																																									
Time(e_2, t_2)																																																										
Agent(e_2, x_3)																																																										
CONTINUATION(k_1, k_2)																																																										

k0 DRS k1	k0 DRS k2
b1 REF x1	b4 REF x3
b1 male n.02 x1	b4 female n.02 x3
k1 REF e1	k2 REF e2
k1 play v.03 e1	k2 sing v.01 e2
k1 Agent e1 x1	k2 Agent e2 x3
k1 Theme e1 x2	b5 REF t2
k1 REF x2	b5 time n.08 t2
k1 piano n.01 x2	b5 TPR t2 "now"
b3 REF t1	k2 Time e2 t2
b3 time n.08 t1	k0 CONTINUATION k1 k2
b3 TPR t1 "now"	k1 Time e1 t1

Figure 6.2: An example of a segmented DRS, both in box format and clausal form. English: *He played the piano and she sang.*

6.2.2 Comparing DRSs to AMRs

Since DRSs in a clausal form come close to the triple notation of AMRs (Cai and Knight, 2013), and both aim to model meaning of natural language expressions, it is instructive to compare these two meaning representations. The main difference between AMRs and DRSs is that the latter ones have explicit scopes (boxes) and scopal operators such as negation. Due to the presence of scope in DRSs, their clauses are more complex than AMR triples.

Type	Description	Example	Total
REF	Discourse referent	b3 REF x2	7,592
NOT	Negation	b1 NOT b2	204
POS	Possibility ($\diamond$)	b4 POS b5	55
NEC	Necessity ($\square$)	b2 NEC b3	14
IMP	Implication ($\Rightarrow$)	b1 IMP b2 b3	104
PRP	Proposition ($:$)	b1 PRP x6	50
REL	Discourse relation	b1 CONTINUATION b2	71
DRS	DRS as a condition	b4 DRS b5	84
Compare	Comparison operators	x1 APX x2	2,100
Concept	WordNet senses	b2 hurt v.02 e3	7,545
Role	Semantic roles	b2 Agent e3 x4	7,516

Table 6.1: Distribution of clause types for 2,049 gold DRSs.

The length of DRS clauses varies from three to four, in contrast to the constant length of AMR triples. Additionally, DRS clauses contain two different types of variables, for scopes and discourse referents, whereas AMR triples have just one type. Lastly, DRS clauses can have three variables, while AMRs have two at most.

Unlike AMRs, DRSs model tense. In general, the tense-related information is encoded in a clausal form with three additional clauses, which express a WordNet concept, semantic role and a comparison operator. In order to give an intuition about the diversity of clauses in DRSs, Table 6.1 shows a distribution of various types of clauses in a corpus of DRSs (see Section 6.3). Since every logical operator carries a scope, their number represents a lower bound of the number of scopes in the meaning representations. In addition to logical operators, scopes are introduced by presupposition triggers like proper names or pronouns.

To make a meaningful comparison between AMRs and DRSs in terms of size, we compare the DRSs of 250,000 English sentences from the Parallel Meaning Bank (PMB; Abzianidze et al. 2017) to AMRs of the same sentences, produced by the AMR parser from Chapter 4. Note that the DRSs are also automatically generated for the most part (see Section 6.3). Statistics of the comparison are shown in Figure 6.3. On average, DRSs are about twice as large as AMRs, in terms of the number of clauses as well as the number of

Figure 6.3: Comparison of the number of triples/clauses and variables between AMRs and DRSs for sentences of different length.

unique variables. This is obviously due to the explicit presence of scope in the meaning representation. However, for both meaning representations the number of clauses and variables increase linearly with sentence length.

6.3 The Parallel Meaning Bank

The scoped meaning representations, integrating word senses, thematic roles, and the list of operators, form the final product of our semantically annotated corpus: the Parallel Meaning Bank. The PMB is a semantically annotated corpus of English texts aligned with translations in Dutch, German and Italian (Abzianidze et al., 2017). It uses the same framework as the Groningen Meaning Bank (Bos et al., 2017), but aims to abstract away from language-specific annotation models. There are five annotation layers present in the PMB: segmentation of words, multi-word expressions and sentences (Evang et al., 2013), semantic tagging (Bjerva et al., 2016b; Abzianidze and Bos, 2017), syntactic analysis based on CCG (Lewis and Steedman, 2014), word senses based on WordNet (Fellbaum, 1998), and thematic role labelling (Bos et al., 2012). The semantic analysis for English is projected on the other languages, to save manual annotation efforts (Evang, 2016; Evang and Bos, 2016). All the information provided by these layers is combined

	Documents	Sentences	Tokens
English	2,049	2,057	11,664
German	641	642	3,430
Italian	387	387	1,944
Dutch	394	395	2,268

Table 6.2: Statistics of the first PMB release.

into a single meaning representation using the semantic parser Boxer (Bos, 2015), in the form of Discourse Representation Structures. Note that the goal is to produce annotations that capture the most probable interpretation of a sentence; no ambiguities or under-specification techniques are employed.

At each step in this pipeline, a single component produces the automatic annotation for all four languages, using language-specific models. Human annotators can correct machine output by adding “Bits of Wisdom” (Basile et al., 2012b). These corrections serve as data for training better models, and create a gold standard annotated subset of the data. Annotation quality is defined per layer and language, at three levels: bronze (fully automatic), silver (automatic with some manual corrections), and gold (fully manually checked and corrected). If all layers are marked as gold, it follows that the resulting DRS can be considered gold standard, too.

The first public release¹ of the PMB contains gold standard scoped meaning representations for over 3,000 sentences in total (see Table 6.2). The release includes mainly relatively short sentences involving several semantic scope phenomena. A detailed distribution of clause types in the data set is given in Table 6.1. A larger amount of texts and more complex linguistic phenomena will be included in future releases.

In addition to the released data, the PMB documents are publicly accessible through a web interface, called the PMB explorer.² In the explorer, visitors can view natural language texts with several layers of annotations and compositionally derived meaning representations, and, after registration, edit the annotations. It is also possible to use a word or a phrase search to find certain words or constructions with their semantic analyses.

¹<http://pmb.let.rug.nl/data.php>

²<http://pmb.let.rug.nl/explorer>

6.4 Matching Scoped Representations

6.4.1 Evaluation by Matching

In the context of the Parallel Meaning Bank there are two main reasons to verify whether two scoped meaning representations capture the same meaning or not: (1) to be able to evaluate semantic parsers that produce scoped meaning representations by comparing gold-standard DRSs to system output; and (2) to check whether translations are meaning-preserving; a discrepancy in meaning between source and target could indicate a mistranslation.

The ideal way to compare two meaning representations would be one based on inference. This can be implemented by translating DRSs to first-order formulas and using an off-the-shelf theorem prover to find out whether the two meanings are logically equivalent (Blackburn and Bos, 2005). This method can compare meaning representation that have different syntactic structures but still are equivalent in meaning. The disadvantage of this approach is that it yields just a binary answer: if a proof is found the meanings are the same, else they are not. For semantic parsing, we prefer a method that can produce more fine-grained scores.

An alternative way of comparing meaning representations is comparing the corresponding clausal forms by computing precision and recall over matched clauses (Allen et al., 2008). The advantage of this approach is that it returns a score between 0 and 1, preferring meaning representations that better approximate the gold standard over those that are completely different. Since the variables of different clausal forms are independent from each other, the comparison of two clausal forms boils down to finding a (partial) one-to-one variable mapping that maximizes intersection of the clausal forms. For example, the maximal matching for the clausal forms in Figure 6.4 is achieved by the following partial mapping from the variables of the left form into the variables of the right one: $\{k0 \mapsto b0, e1 \mapsto v1, t1 \mapsto t1\}$.

For AMRs, finding a maximal matching is done using a hill-climbing algorithm called *Smatch* (Cai and Knight, 2013). This algorithm is based on a simple principle: it checks if a single change in the current mapping results in a better matching mapping. If this is the case, it continues with the

01/3445: He smiled.

x_1	e_1	t_1
male.n.02	(x_1)	
smile.v.01	(e_1)	
	Time	(e_1, t_1)
	Agent	(e_1, x_1)
time.n.08	(t_1)	
	t_1	$\prec$ now

```

b1 REF x1
b1 male n.02 x1
b3 REF t1
b3 TPR t1 "now"
b3 time n.08 t1
k0 Agent e1 x1
k0 REF e1
k0 Time e1 t1
k0 smile v.01 e1

```

00/3514: She fled Australia.

x_1	x_2	v_1	t_1
female.n.02	(x_1)		
flee.v.01	(v_1)		
	Time	(v_1, t_1)	
	Source	(v_1, x_2)	
	Theme	(v_1, x_1)	
time.n.08	(t_1)		
	t_1	$\prec$ now	
country.n.02	(x_2)		
	Name	$(x_2, \text{australia})$	

```

b1 REF x1
b1 female n.02 x1
b3 REF t1
b3 TPR t1 "now"
b3 time n.08 t1
b0 Theme v1 x1
b0 Source v1 x2
b0 REF v1
b0 Time v1 t1
b0 flee v.01 v1
b2 REF x2
b2 Name x2 "australia"
b2 country n.02 x2

```

Figure 6.4: The *Spar* DRS (Section 6.5.1) on the left matches the DRS of 00/3514 PMB document with an F-score of 54.5. If redundant REF-clauses are ignored, the F-score drops to 40. These results are achieved with the help of the mapping $\{k0 \mapsto b0, e1 \mapsto v1, t1 \mapsto t1\}$.

new mapping. Otherwise, if not a single change results in a better mapping, the algorithm stops and has arrived at the final mapping. This means that it can easily get stuck in local optima. To avoid this, *Smatch* does a predefined number of restarts of this process, where each restart starts with a new and random initial mapping. The first restart always uses a *smart* initial mapping, based on matching concepts.

Our evaluation system, called `Counter`³, is a modified version of `Smatch`. Even though clausal forms do not form a graph and clauses consist of either three or four components, the principle behind the variable matching is the same. The actual implementation differs, mainly because `Smatch` was not designed to handle clauses with three variables, e.g., $\langle k0 \text{ Agent } e1 \ x1 \rangle$.

In contrast to `Smatch`, `Counter` takes a set of clauses directly as input. `Counter` also uses two smart initial mappings, based on either role-clauses, like $\langle k0 \text{ Agent } e1 \ x1 \rangle$, or concept-clauses, like $\langle k0 \text{ smile } v.01 \ e1 \rangle$. Moreover, it makes sure that box variables can never match with discourse variables, while for AMR all variables can potentially match. Another addition of `Counter` is that it abstracts over WordNet synsets, i.e., a system is not penalized for predicting a sense that it is in the same synset as the gold standard sense. Moreover, in Chapter 7 we will add a component that checks DRSs for semantic well-formedness.

Also specific to this method is the treatment of REF-clauses in the matching process. Before matching two DRSs, redundant REF-clauses are removed. A REF-clause $\langle b1 \text{ REF } x1 \rangle$ is redundant if its discourse referent $x1$ occurs in some basic condition of the same DRS $b1$. Figure 6.4 shows some examples of redundant REF-clauses. Not removing these redundant clauses would lead to inflated matching scores since for each matched variable the corresponding REF-clause will also match. Comparison of the clausal forms in Figure 6.4 demonstrates this fact. Note that not all REF-clauses are redundant: if a discourse referent is declared outside the scope of negation or an other scope operator, the REF-clause is kept. This is very infrequent in our data, since only a single REF-clause was preserved in 2,049 examples.

6.4.2 Evaluating the Matching Procedure

As we showed in Figure 6.3, DRSs are about twice as large as AMRs. This increase in size might be problematic, since it increases the average runtime for comparing DRSs. Moreover, if there are more variables, more restarts might be needed to ensure a reliable score, again increasing runtime.

³Available at http://github.com/RikVN/DRS_parsing/

Figure 6.5: A clausal form obtained from an automatically generated AMR of the PMB document 14/0849.

Therefore, our goal is that `Counter` gets close to optimal performance in reasonable time. Since we want to be sure that this also holds for longer sentences, we use a balanced data set. We take 1,000 DRSs produced by the semantic parser `Boxer` for each sentence length from 2 to 20 (punctuation excluded), resulting in a set of 19,000 DRSs. To test `Counter` in a realistic setting, we cannot compare the DRSs to themselves or to a DRS of the translation, since those are too similar. Therefore, the 19,000 English sentences of the DRS are parsed by the AMR parser of Chapter 4 and subsequently converted into a DRS by a rule-based system, `Amr2drs`, as motivated by Bos (2016). An example of translating an AMR to a clausal form of a DRS is shown in Figure 6.5. We convert AMR relations to DRS roles by employing a manually created translation dictionary, including rules for semantic roles (e.g., `:ARG0` $\mapsto$ `Agent` and `:ARG1` $\mapsto$ `Patient`) and pronouns (e.g., `she` $\mapsto$ `female.n.02`). Since AMRs do not contain tense information, past tense clauses⁴ are produced for the first verb in the AMR (see the four tense-related clauses in Figure 6.5). Also, since AMRs do not use WordNet synsets, all concepts get a default first sense, except for concepts that are added by concept-specific rules, such as `female.n.02` and `time.n.08`.

⁴Past tense was chosen because it is the most frequent tense in the data set.

Restarts	P	R	F1	Time (h:m:s)
(random) 1	27.20	22.71	24.75	4:19
(smart concepts) 1	27.45	22.92	24.98	4:35
(smart roles) 1	27.27	22.76	24.81	4:37
5	30.25	25.25	27.53	19:33
10	30.65	25.59	27.89	37:08
20	30.84	25.75	28.07	1:10:13
30	30.90	25.80	28.12	1:41:43
50	30.94	25.83	28.16	2:41:38
75	30.96	25.85	28.17	3:53:01
100	30.97	25.85	28.18	5:01:25
Optimal	30.98	25.86	28.19	> 12h

Table 6.3: Results of comparing 19,000 Boxer-produced DRSs to DRSs produced by `Amr2drs`, for different number of restarts. For three or more restarts, we always use the smart role and concept mapping.

Restarts →	4	10	20	30	50	100
Baseline	54.56	54.82	54.97	55.03	55.09	55.14
Baseline + postprocessing	56.94	57.17	57.29	57.34	57.39	57.43
Best gold-only	65.06	65.19	65.24	65.26	65.28	65.30
Best model	71.83	71.89	71.92	71.93	71.94	71.95

Table 6.4: AMR parsing F-scores over different numbers of restarts for four selected models from Chapter 4. Each F-score is an average of 25 runs and is calculated using `Smatch`.

We compare the sets of DRSs using different numbers of restarts to find the best trade-off between speed and accuracy. The results are shown in Table 6.3. The optimal scores are obtained using a Prolog script that performs an exhaustive search for the optimal mapping. As expected, increasing the number of restarts benefits performance. Cai and Knight (2013) consider four restarts the optimal trade-off between accuracy and speed, showing no improvement in F-score when using more than ten restarts. However, in Table 6.4 we show that, in practice, `Smatch` still improves when using more restarts, especially for lower quality models.

Performance for `Counter` also still increases with more than 4 restarts. In our case, it is a bit harder to select an optimal number of restarts, since this number depends on the length of the sentence, as shown in Figure 6.6. We see that for long sentences, 5 and 10 restarts are not sufficient to get close to the optimal mapping, while for short sentences 5 restarts might be considered enough. In general, the best trade-off between speed and accuracy is approximately 20 restarts.

After the publication this chapter was based on (van Noord et al., 2018a), a new evaluation metric was proposed: D-scorer (Liu et al., 2020). This metric converts the DRSs to graphs, after which it calculates an F1-score based on n-gram overlap of partial graphs, similar to the BLEU metric (Papineni et al., 2002) for machine translation. It was mainly developed to replace `Counter` for DRS parsing on GMB data (e.g., Liu et al., 2019a), as the latter was too slow for multi-sentence DRSs. We evaluate our DRS parsing models using D-scorer in Chapter 9.

Figure 6.6: Comparison of the differences to the optimal F-score per sentence length for different number of restarts.

6.5 Counter in Action

6.5.1 Semantic Parsing

The first purpose of `Counter` is to evaluate semantic parsers for DRSs. Since this is a new task, there are no existing systems that are able to do this. Therefore, we show the results of three baseline systems `pmb-pipeline`, `Spar`, and `Amr2drs` (Subsection 6.4.2).⁵

The `pmb-pipeline` produces a DRS via the pipeline of the tools used for automatic annotation of the PMB.⁶ This means that it has no access to manual corrections, and hence it uses the most frequent word senses and default VerbNet roles. `Spar` is a trivial semantic “parser” which always outputs the DRS that is most similar to all other DRSs in the most recent PMB release (for 1.0.0, this is the left-hand DRS in Figure 6.4).

	Precision	Recall	F-score
<code>Spar</code>	53.1	36.6	43.3
<code>Amr2drs</code>	46.5	48.2	47.3
<code>pmb-pipeline</code>	53.0	54.8	53.9

Table 6.5: Comparison of three baseline DRS parsers to the gold-standard data set.

The results of the three baseline parsers are shown in Table 6.5. All parsers are far removed from optimal performance. The score of `pmb-pipeline`, though the highest of the three, shows that manual annotation is still required to obtain gold standard meaning representations. The surprisingly high score of `Spar` is explained by the fact that the first PMB release mainly contains relatively short sentences with little structural diversity. The average number of clauses per clausal form (excluding redundant REF-clauses) is 8.7, where a substantial share (approximately 3) comes from tense-related clauses. Due to this fact, guessing temporal clauses for short sentences has a big impact on F-score. This is illustrated

⁵`Spar` and `Amr2drs` are available at: https://github.com/RikVN/DRS_parsing/

⁶<http://pmb.let.rug.nl/software.php>

by the comparison of the clausal forms in Figure 6.4, where matching only temporal clauses results in an F-score of 40.

This points out a general weakness of the matching tool. All clauses carry equal weight during calculation of the score, while it is clear that, intuitively, certain clauses should have more impact on the score than others (e.g., negation versus tense). On the other hand, it is hard (if not impossible) to come up with a weight scheme that fairly evaluates all (types) of clauses regarding their perceived impact on the final semantic analysis. Moreover, evaluating these weight schemes only seems possible by asking human annotators to somehow judge the similarity of DRSs, who would need specific directions in *how* to judge them, which is what we wanted to test in the first place. Also, the current method at least ensures a transparent score for each DRS-pair, i.e., we can easily check what went wrong in any given parse and how we ended up at the final F-score.

In addition, though the scores are now perhaps inflated, this is not necessarily a problem. For one, we *know* the scores are inflated, so we can account for this in judging how far off we are from “solving” DRS parsing. In other words, an F-score of 70 for DRS parsing cannot be compared to an F-score of 70 for AMR parsing. Second, due to the specific clause matching, we can easily check whether some systems do better on clauses we consider (un)important. So, even if the final F-score has flaws, we can create sub-metrics that calculate scores for different (types) of clauses, similar as in Damonte et al. (2017) for AMR parsing. This is done in Chapter 7 and 9.

6.5.2 Comparing Translations

The second purpose of `Counter` is checking whether translations are meaning-preserving. As a pilot study, we compare the gold standard meaning representations of non-English translations in the release to their English counterparts. The results are shown in Table 6.6. The high F-scores indicate that the meaning representations are often syntactically very similar, if not identical. However, there is a considerable subset of meaning representations which are different from the English ones, indicating that there is at least a slight discrepancy in meaning for those translations.

	F-score	Docs	F<1.0	% total
German	98.4	579	61	10.5
Italian	97.6	341	46	13.5
Dutch	98.3	355	37	10.4

Table 6.6: Comparing meaning representations of English texts to those of German, Italian and Dutch translations.

Manual analysis of these discrepancies showed that there are several different causes for a discrepancy to arise. In most of the cases (38%), a human annotation error was made. In 34% of cases, a definite description was used in one language but not in the other. Examples are ‘has long hair’ with the Italian translation ‘ha i capelli lunghi’, and ‘escape from prison’ with the Dutch translation ‘vluchtte uit de gevangenis’. In 15% of cases proper names were translated (e.g., ‘United States’ and ‘Stati Uniti’). This is not accounted for, since we do not currently make use of grounding proper names to a unique identifier, for instance by wikification (Cucerzan, 2007), or by using a language-independent transliteration of names. In 13% of cases the translation was either non-literal or incorrect. Examples are *Tom lacks experience* with the Dutch translation *Tom heeft geen ervaring* (lit. *Tom has no experience*), *can’t use chopsticks* with the German *kann nicht mit Stäbchen essen* (lit. *cannot eat with sticks*), and *remove the dishes from the table* with the Dutch translation *ruimde de tafel af* (lit. *uncluttered the table*).

The mapping of clausal forms involving non-literal translations is illustrated in Figure 6.7. This preliminary analysis shows that this comparison of meaning representations provides an additional method for detecting mistakes in annotation. It also showed that there are cases where our semantic analysis needs to be revised and improved.

She removed the dishes from the table.

x_1	x_2	e_1	x_3	t_1
female.n.02(x_1)				
remove.v.01(e_1)				
Time(e_1, t_1)				
Source(e_1, x_3)				
Theme(e_1, x_2)				
Agent(e_1, x_1)				
time.n.08(t_1)				
$t_1 \prec \text{now}$				
dish.n.01(x_2)				
table.n.03(x_3)				

~~b1-REF x1~~
b1 female n.02 **x1**
~~b5-REF t1~~
b5 TPR **t1** "now"
b5 time n.08 **t1**
k0 Agent **e1** **x1**
~~k0-REF e1~~
k0 Theme **e1** **x2**
k0 Time **e1** **t1**
k0 remove v.01 **e1**
~~b2-REF x2~~
b2 dish n.01 **x2**
k0 Source **e1** **x3**
~~b4-REF x3~~
b4 table n.03 **x3**

Ze ruimde de tafel af.

x_1	x_2	e_1	t_1
female.n.02(x_1)			
unclutter.v.01(e_1)			
Time(e_1, t_1)			
Source(e_1, x_2)			
Agent(e_1, x_1)			
time.n.08(t_1)			
$t_1 \prec \text{now}$			
table.n.03(x_2)			

~~b1-REF x1~~
b1 female n.02 **x1**
~~b4-REF t1~~
b4 TPR **t1** "now"
b4 time n.08 **t1**
k0 Agent **e1** **x1**
~~k0-REF e1~~
k0 Source **e1** **x2**
k0 Time **e1** **t1**
k0 unclutter v.01 **e1**
~~b2-REF x2~~
b2 table n.03 **x2**

Figure 6.7: English and Dutch non-literal translations of the document 14/0849. Their clausal forms match each other (excl. redundant REF-clauses) with an F-score of 77.8. This matching is achieved by the mapping of variables $\{b5 \mapsto b4, b4 \mapsto b2\}$.

6.6 Conclusions and Future Work

Large semantically annotated corpora are rare. Within the Parallel Meaning Bank project, we aim to create a large, open-domain corpus annotated with formal meaning representations. We take advantage of parallel corpora, enabling the production of meaning representations for several languages at the same time. Currently, these are languages similar to English, two Germanic languages (Dutch and German) and one Romance language (Italian). Ideally, future work would include more non-Germanic languages.

The DRSs that we present are meaning representations with substantial expressive power. They deal with negation, universal quantification, modals, tense, and presupposition. As a consequence, semantic parsing for DRSs is a challenging task. Compared to Abstract Meaning Representations, the number of clauses and variables in a DRS is about two times larger on average. Moreover, compared to AMRs, DRSs rarely contain clauses with single variables. All non-logical symbols used in DRSs are grounded in WordNet and VerbNet (with a few extensions). This makes evaluation using matching computationally challenging, in particular for long sentences, but our matching system `Counter` achieves a reasonable trade-off between speed and accuracy.

Several extensions to the annotation scheme are possible. Currently, the DRSs for the non-English languages contain references to synsets of the English WordNet. Conceptually, there is nothing wrong with this (as synsets can be viewed as identifiers for concepts that are language-independent), but for practical reasons it makes more sense to provide links to synsets of the original language (e.g. Hamp and Feldweg, 1997; Postma et al., 2016). Other future work could focus on a more fine-grained matching regarding WordNet concepts, since the current evaluation tool normalizes over synsets, but does not take graded similarity into account. For example, producing the concept *cat* instead of the concept *tiger* should be rewarded over producing a less similar concept, such as *house* or *car*. Finally, now that we presented an annotated DRS corpus and evaluation tool, we can focus on DRS parsing. In the next chapter, we describe the first DRS parser trained on PMB data.

CHAPTER 7

Neural DRS Parsing

Neural methods have had several recent successes in semantic parsing, though they have yet to face the challenge of producing meaning representations based on formal semantics. We present a sequence-to-sequence neural semantic parser that is able to produce Discourse Representation Structures (DRSs) for English sentences with high accuracy, outperforming traditional DRS parsers. To facilitate the learning of the output, we represent DRSs as a sequence of flat clauses and introduce a method to verify that produced DRSs are well-formed and interpretable. We compare models using characters and words as input and see that the former performs better than the latter. We show that eliminating variable names from the output using De Bruijn-indices increases parser performance. Adding silver training data boosts performance even further. Our final model outperformed the semantic parser Boxer, which is a combination of rule-based and statistical approaches. However, a manual analysis showed that Boxer still outperformed our model on a set of semantic phenomena, such as scope ambiguity, pronoun resolution and identifying discourse relations.

Chapter adapted from:

van Noord, R., Abzianidze, L., Toral, A., and Bos, J. (2018b). Exploring neural methods for parsing discourse representation structures. *TACL*, 6:619–633

7.1 Introduction

Semantic parsing is the task of mapping a natural language expression to an interpretable meaning representation. Semantic parsing used to be the domain of symbolic and statistical approaches (Pereira and Shieber, 1987; Zelle and Mooney, 1996; Blackburn and Bos, 2005). Recently however, neural methods, and in particular sequence-to-sequence models, have been successfully applied to a wide range of semantic parsing tasks. These include code generation (Ling et al., 2016), question-answering (Dong and Lapata, 2016; He and Golub, 2016) and Abstract Meaning Representation parsing (Konstas et al., 2017). Since these models have no intrinsic knowledge of the structure (tree, graph, set) they have to produce, recent work also focused on structured decoding methods, creating neural architectures that always output a graph or a tree (Buys and Blunsom, 2017; Alvarez-Melis and Jaakkola, 2017). These methods often outperform the more general sequence-to-sequence models but are tailored to specific meaning representations.

This chapter will focus on parsing Discourse Representation Structures (DRSs) proposed in Discourse Representation Theory (DRT), a well-studied formalism developed in formal semantics (Kamp, 1984; Van der Sandt, 1992; Kamp and Reyle, 1993; Asher, 1993; Muskens, 1996; van Eijck and Kamp, 1997; Kadmon, 2001; Asher and Lascarides, 2003), dealing with many semantic phenomena: quantifiers, negation, scope ambiguities, pronouns, presuppositions, and discourse structure (see Figure 7.1). DRSs are recursive structures and form therefore a challenge for sequence-to-sequence models because they need to generate a well-formed structure and not something that looks like one but is not interpretable.

The problem that we try to tackle bears similarities with the recently introduced task of mapping sentences to an Abstract Meaning Representation (AMR, Banarescu et al. 2013). But there are notable differences between DRS and AMR. Firstly, DRSs contain scope, which results in a more linguistically motivated treatment of modals, quantification, and negation. And secondly, DRSs contain a substantially higher number of variable bindings (re-entrant nodes in AMR terminology), which are known to be challenging for learning (Damonte et al., 2017).

DRS parsing has been attempted already in the 1980s for small fragments of English (Johnson and Klein, 1986; Wada and Asher, 1986). Wide-coverage DRS parsers based on supervised machine learning emerged later (Bos, 2008b; Le and Zuidema, 2012; Bos, 2015; Liu et al., 2018a). The objectives of this chapter are to apply neural methods to DRS parsing. In particular, we are interested in answers to the following questions:

1. Are sequence-to-sequence models able to produce formal meaning representations (DRSs)?
2. What is better for input: sequences of characters or sequences of words; does tokenization help; and what kind of casing is best used?
3. What is the best way of dealing with variables that occur in DRSs?
4. Does adding silver data increase the performance of the neural parser?
5. What parts of semantics are learned and what parts of semantics are still challenging?

We make the following contributions to semantic parsing:¹ (a) The output of our parser consists of interpretable scoped meaning representations, guaranteed by a specially designed checking tool (Section 7.3); (b) We compare different methods of representing input and output in Section 7.4; (c) We show in Section 7.5 that employing additional, non-gold standard data can improve performance; (d) We perform a thorough analysis of the produced output and compare our methods to symbolic/statistical approaches (Section 7.6).

7.2 Discourse Representation Structures

The Structure of DRS

DRSs are meaning representations introduced by DRT (Kamp and Reyle, 1993). In general, a DRS can be seen as an ordered pair $\langle \mathcal{A}, l : B \rangle$, where $\mathcal{A}$ is a set of presuppositional DRSs, and B a DRS with a label l . The presuppositional DRSs $\mathcal{A}$ can be viewed as propositions that need to be anchored in the context in order to make the main DRS B true, where presuppositions

¹The code is available here: https://github.com/RikVN/Neural_DRS.

Raw input:

Tom isn't afraid of anything.

System output of a DRS in a clausal form:

b1 REF **x1**

b1 male "n.02" **x1**

b1 Name **x1** "tom"

b2 REF **t1**

b2 EQU **t1** "now"

b2 time "n.08" **t1**

b0 NOT **b3**

b3 REF **s1**

b3 Time **s1** **t1**

b3 Experiencer **s1** **x1**

b3 afraid "a.01" **s1**

b3 Stimulus **s1** **x2**

b3 REF **x2**

b3 entity "n.01" **x2**

The same DRS in a box format:

Figure 7.1: DRS parsing in a nutshell: given a raw text, a system has to generate a DRS in the clause format, a flat version of the standard box notation. The semantic representation formats are made more readable by using various letters for variables: the letters x , e , s , and t are used for discourse referents denoting individuals, events, states and time, respectively, while b is used for variables denoting DRS boxes.

comprise anaphoric phenomena too (Van der Sandt, 1992; Geurts, 1999; Beaver, 2002).

DRSs are either elementary DRSs or segmented DRSs. An elementary DRS is an ordered pair of a set of discourse referents and a set of conditions. There are basic conditions and complex conditions. A basic condition is a predicate applied to constants or discourse referents while a complex condition can introduce boolean operators ranging over DRSs (negation, condi-

tionals, disjunction). Segmented DRSs capture discourse structure by connecting two units of discourse by a discourse relation (Asher and Lascarides, 2003). Please see Section 2.1.2 for a more detailed overview of DRT and DRS.

Annotated Corpora

Despite a long tradition of formal interest in DRT, it is only since recently that textual corpora annotated with DRSs have been made available. The Groningen Meaning Bank (GMB) is a large corpus with DRS annotation for mostly short English newspaper texts (Basile et al., 2012a; Bos et al., 2017). The DRSs in this corpus are produced by an existing semantic parser and then partially corrected. The DRSs in the GMB are therefore not gold standard.

A similar corpus is the Parallel Meaning Bank (PMB), that provides DRSs for English, German, Dutch and Italian sentences based on a parallel corpus (Abzianidze et al. 2017; also see Chapter 6). The PMB, too, is constructed using an existing semantic parser, but a part of it is completely manually checked and corrected (i.e., gold standard). In contrast to the GMB, the PMB involves two major additions: (a) its semantics are refined by modelling tense and employing semantic tagging (Bjerva et al., 2016b; Abzianidze and Bos, 2017), and (b) the non-logical symbols of the DRSs corresponding to concepts and semantic roles are grounded in WordNet (Fellbaum, 1998) and VerbNet (Bonial et al., 2011) respectively.

These above-mentioned additions make the DRSs of the PMB more fine-grained meaning representations. For this reason we choose the PMB (over the GMB) as our corpus for evaluating our semantic parser. Even though the sentences in the current release of the PMB are relatively short, they contain many hard semantic phenomena that a semantic parser has to deal with: pronoun resolution, quantifiers, scope of modals and negation, multi-word expressions, word senses, semantic roles, presupposition, tense, and discourse relations. As far as we know, we are the first that employs the PMB corpus for semantic parsing.

Formatting DRSs with Boxes and Clauses

The usual way to represent DRSs is the well-known box-format. In order to facilitate reading a DRS with unresolved presuppositions, it can be depicted as a network of boxes, where a non-presuppositional (i.e., main) DRS $I : B$ is connected to the presuppositional DRSs A with arrows. Each box comes with a unique label and has two rows. In case of elementary DRSs these rows contain discourse referents in the top row and conditions in the bottom row (Figure 7.1). A segmented DRS has a row with labeled DRSs and a row with discourse relations (Figure 7.2).

The DRS in Figure 7.1 consists of a main box b_0 and two presuppositional boxes, b_1 and b_2 . Note that b_0 has no discourse referents but introduces negation via a single condition $\neg b_3$ with a nested box b_3 . The conditions of b_3 represent unary and binary relations over discourse referents that are introduced either by b_3 or the presuppositional DRSs.

A clausal form is another way of formatting DRSs. It represents a DRS as a set of clauses (see Figure 7.1 and 7.2). This format is better suitable for machine learning than the box-format as it has a simple, flat structure and facilitates partial matching of DRSs which is useful for evaluation (see Chapter 6). Conversion from the box-notation to the clausal form and vice versa is transparent: discourse referents, conditions, and discourse relations in the clausal form are preceded by the label of the box they occur in. Notice that the variable letters in the semantic representations are automatically set and they simply serve for readability purposes. Throughout the experiments described in this chapter, we employ clausal form DRSs.

7.3 Method

7.3.1 Annotated Data

We use the English DRSs from release 2.1.0 of the PMB (Abzianidze et al., 2017).² The release suggests to use the parts 00, 10, 20 and 30 as the development set, resulting in 3,998 train and 557 development instances. Basic

²<http://pmb.let.rug.nl/data.php>

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 7.2: An example of a segmented DRS. Discourse relations are formatted with uppercase characters.

statistics are shown in Table 7.1, while the number of occurrences of some of the semantic phenomena mentioned in Section 7.2 are given in Table 7.2.

Since this is a rather small training set, we tune our model using 10-fold cross-validation (CV) on the training set, instead of tuning on a separate development set. This means that we will use the suggested development set as a test set (and refer to it as such). When testing on this set, we train a model on all available training data. The employed PMB release also comes

	Sentences	Tokens	Avg tok/sent
Gold train	3,998	24,917	6.2
Gold test	557	3,180	5.7
Silver	73,778	638,610	8.7

Table 7.1: Number of documents, sentences and tokens for the English part of PMB release 2.1.0. Note that the number of tokens is based on the PMB tokenization, treating multi-word expressions as a single token.

Phenomenon	Train	Test	Silver
negation & modals	442	73	17,527
scope ambiguity	≈67	15	≈3,108
pronoun resolution	≈291	31	≈3,893
discourse rel. & imp.	254	33	16,654
embedded clauses	≈160	30	≈46,458

Table 7.2: Counts of relevant semantic phenomena for PMB release 2.1.0.³ These phenomena are described and further discussed in Section 7.6.3.

with “silver” data, namely, 71,308 DRSs that are only partially manually corrected. In addition, we employ the DRSs from the silver data but without the manual corrections, which makes them “bronze” DRSs following the PMB terminology. Our experiments will initially use only the gold standard data, after which we will employ the silver or bronze data to further push the score of our best systems.

7.3.2 Clausal Form Checker

The clausal form of a DRS needs to satisfy a set of constraints in order to correspond to a semantically interpretable DRS, i.e., translatable into a first-order logic formula without free occurrences of a variable (Kamp and Reyle, 1993). For example, all discourse referents need to be explicitly introduced with a `REF` clause to avoid free occurrences of variables.

³The phenomena are automatically counted based on clausal forms. The counting algorithm does not guarantee the exact number for certain phenomena, though it returned the exact counts of all the phenomena on the test data except for pronoun resolution (30).

We implemented a clausal form checker that validates the clausal form if and only if it represents a semantically interpretable DRS. Distinguishing box variables from entity variables is crucial for the validity checking, but automatically learned clausal forms are not expected to differentiate variable types. First, the checker separately parses each clause in the form to induce variable types based on the fixed set of comparison and DRS operators. After typing all the variables, the checker verifies whether the clauses collectively correspond to a DRS with well-formed semantics. For each box variable in a discourse relation, existence of the corresponding box inside the same segmented DRS is checked. For each entity variable in a condition, an introduction of the binder (i.e., accessible) discourse variable is found. The goal of these two steps is to prevent free occurrences of variables in DRSs. While binding the entity variables, necessary accessibility relations between the boxes are induced. In the end, the checker verifies the transitive closure of the induced accessibility relation on loops and checks existence of a unique main box of the DRS.

The checker is applied to every automatically obtained clausal form. If a clausal form fails the test, it is considered as ill-formed and will not have a single clause matched with the gold standard when calculating the F-score.

7.3.3 Evaluation

A DRS parser is evaluated by comparing its output DRS to a gold standard DRS using the Counter tool (see Chapter 6). Counter calculates an F-score over matching clauses by applying a hill-climbing search to find the best variable mapping between two sets of clauses. This is similar to the `Smatch` tool for AMR parsing.⁴ Counter generalizes over WordNet synsets, i.e., a system is not penalized for predicting a word sense that is in the same synset as the gold standard word sense. To calculate whether there is a significant difference between two systems, we perform approximate randomization (Noreen, 1989) with $\alpha = 0.05$, $R = 1000$ and $F(model_1) > F(model_2)$ as test statistic for each individual DRS pair.

⁴Counter ignores `REF` clauses in the calculation of the F-score since they are usually redundant and therefore inflate the final score (see Chapter 6).

7.3.4 Neural Architecture

We employ a recurrent sequence-to-sequence neural network (henceforth seq2seq) with two bidirectional LSTM layers and 300 nodes, implemented in OpenNMT (Klein et al., 2017). The network encodes a sequence representation of the natural language utterance, while the decoder produces the sequences of the meaning representation. We apply naive dropout (Srivastava et al., 2014; Zaremba et al., 2014) between both the recurrent encoding and decoding layers to prevent over-fitting, and use general attention (Luong et al., 2015) to selectively give more weight to certain parts of the input sentence. During decoding we perform beam search with length normalization, which in neural machine translation (NMT) is crucial to obtaining good results (Britz et al., 2017). This is the same model we used in Chapters 4 and 5, of which a detailed overview is given in Section 3.3. The hyper-parameters are different, though. We experimented with a wide range of hyper-parameter settings, of which the final settings can be found in Table 7.3.

We opted against trying to find the best hyper-parameter settings for each individual experiment (next to impossible in terms of computing time necessary as a single 10-fold CV experiment takes 12 hours on GPU), but selected settings that showed good performance for both the initial character and word-level representations (see Section 7.4 for details). The hyper-parameter search was performed using 10-fold CV on the training set. Training is stopped when there is no more improvement in perplexity on the validation set, which in our case occurred after 13–15 epochs.

A powerful, well-known technique in the field of NMT is to use an ensemble of models during decoding (Sutskever et al., 2014; Sennrich et al., 2016a). The resulting model averages over the predictions of the individual models, which can balance out some of the errors. In our experiments, we apply this method when decoding on the test set, but not for our experiments of 10-fold CV, as this would take too much computation time.

7.4 Experiments with Data Representations

This section describes the experiments we conduct regarding the data representations of the input (sentences) and output (DRSs) during training.

Setting	Value	Setting	Value
RNN-type	LSTM	dropout	0.2
encoder-type	brnn	dropout type	naive
optimizer	sgd	bridge	copy
layers	2	learning rate	0.7
nodes	300	learning rate decay	0.7
min freq source	3	max grad norm	5
min freq target	3	beam size	10
vector size	300	length normalization	0.9

Table 7.3: Hyper-parameters explored during training and testing with their final values. All other hyper-parameters have default values.

7.4.1 Between Characters and Words

We first try two (default) representations: character-level and word-level. Most semantic parsers use word-level representations for the input, but as a result are often dependent on pretrained word embeddings or anonymization of the input⁵ to obtain good results. Character-level models avoid this issue but might be at a higher risk of producing ill-formed output.

Character-based model In the character-level model, the input (an English sentence) is represented as a sequence of individual characters. The output (a DRS in clause format) is linearized, with special characters indicating spaces and clause separators. The semantic roles (e.g., *Agent*, *Theme*), DRS operators (e.g., *REF*, *NOT*, *POS*) and deictic constants (e.g., "now", "speaker", "hearer") are not represented as character sequences, but treated as compound characters, meaning that *REF* is not treated as a sequence of *R*, *E* and *F*, but directly as *REF*. All proper names, WordNet senses, time/date expressions, and numerals are represented as character sequences.

Word-based model In the word-level model, the input is represented as a sequence of words, using spaces as a separator (i.e., the original words

⁵This is done to keep the vocabulary small. An example is to change all proper names to *NAME* in both the sentence and meaning representation during training. When producing output, the original names are restored by switching *NAME* with a proper name found in the input sentence (Konstas et al., 2017).

are kept). The output is the same as for the character-based model, except that the character sequences are represented as words. We use pretrained GloVe embeddings (Pennington et al., 2014)⁶ to initialize the encoder and decoder representations. In the DRS representation, there are semantic roles and DRS operators that might look like English words, but should not be interpreted as such (e.g., *Agent*, *NOT*). These entities are removed from the set of pretrained embeddings, so that the model will learn them from scratch (starting from a random initialization).

Hybrid representations: BPE We do not necessarily have to restrict ourselves to using only characters or words as input representation. In NMT, byte-pair encoding (BPE, Sennrich et al. 2016b) is currently the *de facto* standard (Bojar et al., 2017). This is a frequency-based method that automatically finds a representation that is in between character and word-level. It starts out with the character-level format and then does a predefined number of merges of frequently co-occurring characters. Tuning this number of merges determines if the resulting representation is closer to character or word-level. We explore a large range of merges (1k–100k), while applying a corresponding set of pretrained BPE embeddings (Heinzerling and Strube, 2018). However, none of the BPE experiments improved on the character-level or word-level score (F-scores between 57 and 68), only coming close when using a small number of merges (which is very close to character-level anyway). Therefore this technique was disregarded for further experiments.

Combined char and word There is also a fourth possible representation of the input: combining the character and word-level representations. This is uncommon in NMT due to the large size of the embedding space (hence their preference for BPE), but possible here since the PMB data contains relatively short sentences. We simply add the word embedding vector after the sequence of character-embeddings for each word in the input and still initialize these embeddings using the pretrained GloVe embeddings. In Chapter 9 we will experiment more with other methods of combining character and word-level representations, including using contextual embeddings (Peters et al., 2018; Devlin et al., 2019).

⁶The Common Crawl version trained on 840 billion tokens, vector size 300.

Representation results The results of the experiments (10-fold CV) for finding the best representation are shown in Table 7.4. Character representations are clearly better than word representations, though the word-level representation produces fewer ill-formed DRSs. Both representations are maintained for our further experiments. Although the combination of characters and words did lead to a small increase in performance over characters only (Table 7.4), this difference is not significant. Hence, this representation is discarded in further experiments described in this chapter.

Model	Prec	Rec	F-score	% ill
Char	78.1	69.7	73.7	6.2
Word	73.2	65.9	69.4	5.8
Char + Word	78.9	69.7	74.0	7.5

Table 7.4: Evaluating different input representations. The percentage of ill-formed DRSs is denoted by % ill.

7.4.2 Tokenization

An interesting aspect of the PMB data is the way the input sentences are tokenized. In the data set, multi-word expressions are tokenized as single words, for example, “New York” is tokenized to “New~York”. Unfortunately, most off-the-shelf tokenizers (e.g., the Moses tokenizer) are not equipped to deal with this. We experiment with using Elephant (Evang et al., 2013), a tokenizer that can be (re-)trained on individual data sets, using the tokenized sentences of the published silver and gold PMB data set.⁷ Simultaneously, we are interested in whether character-level models need tokenization at all, which would be a possible advantage of this type of representation.

Results of the experiment are shown in Table 7.5. None of the two tokenization methods yielded a significant advantage for the character-level models, so they will not be employed further. The word-level models, however, did benefit from tokenization, but Elephant did not give us an advantage over the Moses tokenizer. Therefore, for word-level models, we will use Moses (Koehn et al., 2007) in our next experiments.

⁷Gold tokenization is available in the data set, but using this would not reflect practical applications of DRS parsing, as we want raw text as input for a realistic setting.

7.4.3 Representing Variables

So far we did not attempt to do anything special with the variables that occur in DRSs, as we simply tried to learn them as supplied in the PMB data set. Obviously, DRSs constitute a challenge for seq2seq models because of the high number of multiple occurrences of the same variables, in particular compared to AMR. AMR parsers do not deal well with this, since the re-entrancy metric (Damonte et al., 2017) is among the lowest metrics for all AMR parsers that reported them or are publicly available (see Chapter 4). Moreover, for AMR, only 50% of the representations contain at least one re-entrant node, and only 20% of the triples in AMR contain a re-entrant node (as we saw in Chapter 5), but for DRSs these are both virtually 100%. While seq2seq AMR parsers could get away with ignoring variables during training and reinstating them in a postprocessing step, for DRSs this is unfeasible.

b1 REF x1	\$1 REF @1	[NEW] REF <NEW>
b1 male "n.02" x1	\$1 male "n.02" @1	[0] male "n.02" <0>
b1 Name x1 "tom"	\$1 Name @1 "tom"	[0] Name <0> "tom"
b2 REF t1	\$2 REF @2	[NEW] REF <NEW>
b2 EQU t1 "now"	\$2 EQU @2 "now"	[0] EQU <0> "now"
b2 time "n.08" t1	\$2 time "n.08" @2	[0] time "n.08" <0>
b0 NOT b3	\$0 NOT \$3	[NEW] NOT [NEW]
b3 REF s1	\$3 REF @3	[0] REF <NEW>
b3 Time s1 t1	\$3 Time @3 @2	[0] Time <0> <-1>
b3 Experiencer s1 x1	\$3 Experiencer @3 @1	[0] Experiencer <0> <-2>
b3 afraid "a.01" s1	\$3 afraid "a.01" @3	[0] afraid "a.01" <0>
b3 Stimulus s1 x2	\$3 Stimulus @3 @4	[0] Stimulus <0> <1>
b3 REF x2	\$3 REF @4	[0] REF <NEW>
b3 entity "n.01" x2	\$3 entity "n.01" @4	[0] entity "n.01" <0>
(a) Standard	(b) Absolute	(c) Relative

Figure 7.3: Different methods of variable naming exemplified on the clausal form of Figure 7.1. For (c), positive numbers refer to introductions that have yet to occur, while negative numbers refer to known introductions. A zero refers to the previous introduction for that variable type.

However, since variable names are chosen arbitrarily, they will be hard for a seq2seq model to learn. We will therefore experiment with two methods of rewriting the variables to a more general representation, distinguishing between box variables and discourse variables. Our first method (absolute) traverses down the list of clauses, rewriting each new variable to a unique representation, taking the order into account. The second method (relative) is more sophisticated; it rewrites variables based on when they were introduced, inspired by De Bruijn index (de Bruijn, 1972). We view box variables as introduced when they are first mentioned, while we take the `REF` clause of a discourse referent as their introduction. The two rewriting methods are illustrated in Figure 7.3.

The results are shown in Table 7.5. For both characters and words, the relative rewriting method significantly outperforms the absolute method and the baseline, though the absolute method produces fewer ill-formed DRs. Interestingly, the character-level model still obtains a higher F1-score compared to the word-level model, even though it produces more ill-formed DRs.

	Char parser		Word parser	
	F1	% ill	F1	% ill
Baseline (bs)	73.7	6.2	69.4	5.8
Moses (mos)	74.1	4.8	71.8	5.8
Elephant (ele)	74.0	5.4	71.1	7.5
bs/mos + absolute (abs)	75.3	3.5	73.5	2.0
bs/mos + relative (rel)	76.3	4.2	74.2	3.1
bs/mos + rel + lowercase	75.8	3.6	74.9	3.1
bs/mos + rel + truecase	76.2	4.0	73.3	3.3
bs/mos + rel + feature	76.9	3.7	74.9	2.9

Table 7.5: Results of the 10-fold CV experiments regarding tokenization, variable rewriting and casing. *bs/mos* means that we use no tokenization for the character-level parser, while we use Moses for the word-level parser.

7.4.4 Casing

Casing is a writing device mostly used for punctuation purposes. On the one hand, it increases the set of characters (hence adding more redundant variation to the input). On the other hand, case can be a useful feature to recognize proper names as names of individuals are semantically analysed as pre-suppositions. Explicitly encoding uppercase with a feature could therefore prevent us from including a named-entity recognizer, often used in other semantic parsers. Although we do not expect dealing with case is a major challenge, we try out different techniques to find an optimal balance between abstracting over input characters and parsing performance. The results, in Table 7.5, show that the feature works well for the character-level model, but for the word-level model, it does not outperform lowercasing. These settings are used in further experiments.

7.5 Experiments with Additional Data

Since semantic annotation is a difficult and time-consuming task, gold standard data sets are usually relatively small. This means that semantic parsers (and data-hungry neural methods in particular) can often benefit from more training data. Some examples in semantic parsing are data recombination (Jia and Liang, 2016), paraphrasing (Berant and Liang, 2014) or exploiting machine-generated output (Konstas et al., 2017). However, before we do any experiments using extra training data, we want to be sure that we can still benefit from more gold training data. For both the character-level and word-level we plot the learning curve, adding 500 training instances at a time, in Figure 7.4. For both models the F-score clearly still improves when using more training instances, which shows that there is at least the potential for additional data to improve the score.

For DRSs, the PMB-2.1.0 release already contains a large set of silver standard data (71,308 instances), containing DRSs that are only partially manually corrected. We then train a model on both the gold and silver standard data, making no distinction between them during training. After training we take the last model and restart the training on only the gold data, in a similar process as described in Chapter 4.

Figure 7.4: Learning curve for different number of gold instances for both the character-level and word-level neural parsers (10-fold CV experiment for every 500 instances).

We are aware that there are many methods to obtain and employ additional data. However, our main aim is not to find the optimal method for DRS parsing, but to demonstrate that using additional data is indeed beneficial for neural DRS parsing. Since we are not further fine-tuning our model, we will show results on the test set in this section. Table 7.6 shows the results of adding the silver data. This results in a large increase in performance, for both the character and word-level models. We are still reliant on manually annotated data, however, since without the gold data (so training on only the silver data), we score even lower than our baseline model (68.4 and 68.1 for the char and word parser). Similarly, we are reliant on the fine-tuning procedure, as we also score below our baseline models without it (71.6 and 71.0 for the char and word parsers, respectively).

We believe there are two possible factors that could explain why adding silver data resulted in such a large improvement: (i) the fact that the data is silver instead of bronze or (ii) the fact that a different DRS parser (Boxer, see Section 7.6), is used to create the silver data instead of our own parser.

	Char parser		Word parser	
	F1	% ill	F1	% ill
Best gold-only	75.9	2.9	72.8	2.0
+ ensemble	77.9	1.8	75.1	0.9
Gold + silver	82.9	1.8	82.7	1.1
+ ensemble	83.6	1.3	83.1	0.7

Table 7.6: F1-score and percentage of ill-formed DRSs on the test set, for the experiments with the PMB-released silver data. The scores without using an ensemble are an average of five runs of the model.

	Char parser		Word parser	
Data	F1	% ill	F1	% ill
Silver (Boxer-generated)	83.6	1.3	83.1	0.7
Bronze (Boxer-generated)	83.8	1.1	82.4	0.9
Bronze (NN-generated)	77.9	2.7	74.5	2.2
without ill-formed DRSs	78.6	1.6	74.9	0.9

Table 7.7: Test set results of the experiments that analyse the impact of the silver data.

We conduct an experiment to find out the impact on performance of silver vs bronze and Boxer vs our parser. The results are shown in Table 7.7. Note that these experiments are performed to analyse the impact of the silver data, not to further push the score, meaning *Silver (Boxer-generated)* is our final model that will be compared to other approaches in Section 7.6.

For (i), we compare the performance of the model trained on silver and bronze versions of the exact same documents (so leaving out the manual corrections). Interestingly, we score slightly higher for the character-level model with bronze than with silver (though the difference is not statistically significant), meaning that the extra manual corrections are not beneficial (in their current format). This suggests that the silver data is closer to bronze than to gold standard.

	Prec	Rec	F-score
spar	48.0	33.9	39.7
sim-spar	55.6	57.9	56.8
amr2drs	43.3	43.0	43.2
Boxer	75.7	72.9	74.3
Neural Char	79.7	76.2	77.9
Neural Word	77.1	73.3	75.1
Neural Char + silver	84.7	82.4	83.6
Neural Word + silver	84.0	82.3	83.1

Table 7.8: Test set results of our best neural models compared to two baseline models and two parsers.

For (ii), we use our own best gold-only parser to parse the sentences in the PMB silver data release and use that as additional training data.⁸ Since the silver data contains longer and more complicated sentences than the gold data, our best parser produces more ill-formed DRSs (13.7% for char and 15.6% for word). We can either discard those instances or still maintain them for the model to learn from. For Boxer this is not an issue since only 0.3% of DRSs produced were ill-formed. We observe that a full self-training pipeline results in lower performance compared to using Boxer-produced DRSs. In fact, this does not seem to be beneficial over only using the gold standard data. Most likely, since Boxer combines symbolic and statistical methods, it learns very different things than our neural parsers, which in turn provides more valuable information to the model. A different explanation is that, possibly, self-training only works when the model can exploit larger quantities of bronze data. We will not be experimenting with this in this chapter, but will do so for AMR parsing in Chapter 9. A more detailed analysis on the difference in (semantic) output is performed in Section 7.6.2 and 7.6.3. Removing ill-formed DRSs before training leads to higher F-scores for both the char and word parser, as well as a lower number of ill-formed DRSs.

⁸Note that we cannot apply the manual corrections, so in PMB terminology, this data is bronze instead of silver.

7.6 Discussion

7.6.1 Comparison

In this section, we compare our best neural models (with and without silver data, see Table 7.6) to two baseline systems and to two DRS parsers: *Amr2drs* and *Boxer*. *Amr2drs* is a parser that obtains DRSs from AMRs by applying a set of rules (Chapter 6), in our case using AMRs produced by the AMR parser of Chapter 4. *Boxer* is an existing DRS parser using a statistical CCG parser for syntactic analysis and a compositional semantics based on λ -calculus, followed by pronoun and presupposition resolution (Curran et al., 2007; Bos, 2008b). *Spar* is a baseline parser that outputs the same (fixed) default DRS for each input sentence. We also implemented a second baseline model, *Sim-spar*, which outputs, for each sentence in the test set, the DRS of the most similar sentence in the training set. This similarity is calculated by taking the cosine similarity of the average word embedding vector (with removed stopwords) based on the Glove embeddings (Pennington et al., 2014).

Table 7.8 show the result of the comparison. The neural models comfortably outperform the baselines. We see that both our neural models outperform *Boxer* by a large margin when using the *Boxer* labeled silver data. However, even without this dependence, the neural models perform significantly better than *Boxer*. It is worth noting that the character-level model significantly outperforms the word-level model, even though it cannot benefit from pretrained word embeddings and from a tokenizer.

Concurrently with this work, a neural DRS parser has been developed by Liu et al. (2018a). They convert the DRSs to a tree representation and apply a customised neural encoder-decoder model with a structured decoder. Their system produces the DRS in three stages, similar to the coarse-to-fine approach of Dong and Lapata (2018). It first predicts the general (deep) structure of the DRSs, after which the conditions and referents are filled in. Unfortunately, they train and evaluate their parser on annotated (sentence-level) data from the GMB rather than from the PMB. The GMB does not contain word senses, tense information and the evaluation sets are not gold standard. This makes it difficult to compare the approaches. However, they do

	Char	Word	Boxer
All clauses	83.6	83.1	74.3
DRS Operators	93.2	93.3	88.0
VerbNet roles	84.1	82.5	71.4
WordNet synsets	79.7	79.4	72.5
nouns	86.1	88.5	82.5
verbs, adverbs, adj.	65.1	58.7	49.3
Oracle sense numbers	86.7	85.7	78.1
Oracle synsets	90.7	90.9	83.8
Oracle roles	87.4	87.2	82.0

Table 7.9: F-scores of fine-grained evaluation on the test set of the three semantic parsers.

compare to a similar model as the one we are using in this chapter, and outperform it by a large margin. They do not apply a format checker to check if the final DRSs are well-formed, though it is more likely that this would be a disadvantage for more vanilla sequence-to-sequence models, such as ours.

7.6.2 Analysis

An intriguing question is what our models actually learn, and what parts of meaning are still challenging for neural methods. We do this in two ways, by performing an automatic analysis and by doing a manual inspection on a variety of semantic phenomena. Table 7.9 shows an overview of the different automatic evaluation metrics we implemented with corresponding scores of the three models.

The character- and word-level systems perform comparably in all categories except for VerbNet roles, where the character-based parser shows a clear advantage (1.6% absolute). The score for WordNet synsets is similar, but the word-level model has more difficulty predicting synsets that are introduced by verbs than for nouns. It is clear that the neural models outperform Boxer consistently on each of these metrics (partly because Boxer picks the first sense by default). What also stands out is the impact of the word senses: with a perfect word sense disambiguation module (oracle senses) large improvements can be gained for all three parsers.

Figure 7.5: Performance of each parser for sentences of different length.

It is interesting to look at what errors the model makes in terms of producing ill-formed output. For both the neural parsers, only about 2% of the ill-formed DRSs are ill-formed because of a syntactic error in an individual clause (e.g., `b1 Agent x1`, where a fourth argument is missing), while all the other errors are due to a violated semantic constraint (see Section 7.3.2). In other words, the produced output is a syntactically well-formed DRS but is not interpretable.

To find out how sentence length affects performance, we plot in Figure 7.5 the mean F-score obtained by each parser on input sentences of different lengths, from 3 to 10 words.⁹ We observe that all the parsers degrade with sentence length. To find out whether any of the parsers degrades significantly more than any other, we build a regression model, in which we predict the F-score using as predictors the parser (char, word and Boxer), the sentence length and the number of clauses produced. According to the regression model, (i) the performance of all the three systems decreases with

⁹Shorter and longer sentences are excluded as there are fewer than 10 input sentences for any such length, e.g., there are only 3 sentences that have 2 words.

sentence length, thus corroborating the trends shown in Figure 7.5 and (ii) the interaction between parser and sentence length is not significant, i.e., none of the parsers decreases significantly more than any other with sentence length. The fact that the performance of the neural parsers degrades with sentence length is not surprising, since they are based on the seq2seq architecture, and models built on this architecture for other tasks, such as machine translation, have been shown to have the same issue (Toral and Sánchez-Cartagena, 2017).

7.6.3 Manual Inspection

The automatic evaluation metrics provide overall scores but do not capture how the models perform on certain semantic phenomena present in the DRs. Therefore, we manually inspected the test set output of the three parsers for the semantic phenomena listed in Table 7.2. Below we describe each phenomenon and explain how the parser output is evaluated on them.

The *negation & modals* phenomenon covers possibility (POS), necessity (NEC), and negation (NOT). The phenomenon is considered successfully captured if an automatically produced clausal form has the clause with the modal operator and the main concept is correctly put under the scope of the modal operator. For example, to capture the negation in Figure 7.1, the presence of **b0** NOT **b3** and **b3** afraid "a.01" **s1** is sufficient. *Scope ambiguity* counts nested pairs of scopal operators such as possibility (POS), necessity (NEC), negation (NOT), and implication (IMP). *Pronoun resolution* checks if an anaphoric pronoun and its antecedent are represented by the same discourse referent. *Discourse relation & implication* involves determining a discourse relation or an implication with a main concept in each of their scopes (i.e., boxes). For instance, to get the discourse relation in Figure 7.2 correctly, a clausal form needs to include **b0** CONTINUATION **b1** **b5**, **b1** play "v.03" **e1**, and **b5** sing "v.01" **e2**. Finally, the *embedded clauses* phenomenon verifies whether the main verb concept of an embedded clause is placed inside the propositional box (PRP). This phenomenon also covers control verbs: it checks if a controlled argument of a subordinate verb is correctly identified as an argument of a control verb.

Phenomenon	#	Char	Word	Boxer
negation & modals	73	0.90	0.81	0.89
scope ambiguity	15	0.73	0.57	0.80
pronoun resolution	31	0.84	0.77	0.90
discourse rel. & imp.	33	0.64	0.67	0.82
embedded clauses	30	0.77	0.70	0.87

Table 7.10: Manual evaluation of the output of the three semantic parsers on several semantic phenomena. Reported numbers are accuracies.

The results of the semantic evaluation of the parsers on the test set is given in Table 7.10. The character-level parser performs better than the word-level parser on all the phenomena except one. Even though both our neural parsers clearly outperformed Boxer in terms of F-score, they perform worse than Boxer on the selected semantic phenomena. Although the differences are not big, Boxer obtained the highest score for four out of five phenomena. This suggests that just the F-score is perhaps not good enough as an evaluation metric, or that the final F-score should perhaps be weighted towards certain clauses. For example, it is arguably more important to capture a negation correctly than tense. Our current metric only gives a rough indication about the contents, but not about the inferential capabilities of the meaning representation.

7.7 Conclusions and Future Work

We implemented a general, end-to-end neural seq2seq model that is able to produce well-formed DRSs with high accuracy (**RQ1**). Character-level models can outperform word-level models, even though they are not dependent on tokenization and pretrained word embeddings (**RQ2**). It is beneficial to rewrite DRS variables to a more general representation (**RQ3**). Obtaining and employing additional data can benefit performance as well, though it might be better to use an external parser instead of doing a full self-training pipeline (**RQ4**). F-score is only a rough measure for semantic accuracy: Boxer still outperformed our best neural models on a subset of specific semantic phenomena (**RQ5**).

Future work We think there are a lot of opportunities for future work. Since the sentences in the PMB data set are relatively short, it makes sense to investigate seq2seq models performing well for longer texts. There are a few promising directions here that could combat the degrading performance on longer sentences. First, the Transformer model (Vaswani et al., 2017) is an interesting candidate for exploration, a state-of-the-art neural model developed for MT that does not have worse performance for longer sentences. This will be explored in Chapter 9. Second, a seq2seq model that is able to first predict the general structure of the DRS, after which it can fill in the details, similar to Liu et al. (2018a), is something that could be explored. A third possibility is a neural parser that tries to build the DRS incrementally, producing clauses for different parts of the sentence individually, and then combining them to a final DRS. Fourth, the neural parser might benefit from using linguistic information, such as a syntactic analysis, which could help in parsing longer sentences. We will explore this in Chapter 8.

CHAPTER 8

Linguistic Information in Neural DRS Parsing

Recently, character-level sequence-to-sequence models have achieved impressive performance on a number of semantic parsing tasks. However, they often do not exploit available linguistic resources, while these, when employed correctly, are likely to increase performance even further. Research in neural machine translation has shown that employing this information has a lot of potential, especially when using a multi-encoder setup. We employ a range of semantic and syntactic resources to improve performance for the task of Discourse Representation Structure Parsing. This information could be especially helpful for parsing longer sentences, which were challenging for the DRS parser described in Chapter 7. We show that (i) linguistic features can be beneficial for neural semantic parsing and (ii) the best method of adding these features to the character-level model is by using multiple encoders. We also participated in the first shared task on DRS parsing, for which we improved model performance by creating a number of repairing strategies for ill-formed DRSs.

Chapter adapted from:

van Noord, R., Toral, A., and Bos, J. (2019). Linguistic information in neural semantic parsing with multiple encoders. In *IWCS - Short Papers*, pages 24–31, Gothenburg, Sweden

van Noord, R. (2019). Neural Boxer at the IWCS shared task on DRS parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden

8.1 Introduction

Sequence-to-sequence neural networks have shown remarkable performance in semantic parsing (Ling et al., 2016; Jia and Liang, 2016; Konstas et al., 2017; Dong and Lapata, 2018; Liu et al., 2018a). As we also observed in the previous chapters, this architecture is able to learn meaning representations for a range of semantic phenomena, usually without resorting to any linguistic information such as part-of-speech or syntax. Though this is an impressive feat in itself, there is no reason to abandon these resources. Even in machine translation, where models can be trained on relatively large data sets, it has been shown that sequence-to-sequence models can benefit from external syntactic and semantic resources (Sennrich and Haddow, 2016; Aharoni and Goldberg, 2017) and a multi-source approach has proven particularly successful for adding syntactic information (Currey and Heafield, 2018). This could be an effective method for increasing the performance on longer sentences, which are often challenging for neural semantic parsers. The current approaches in neural semantic parsing either include (some) linguistic information in a single encoder (POS-tags in Chapter 4 and 5, lemmas in Liu et al. 2018a), or use multiple encoders to represent multiple languages rather than different sources of linguistic knowledge (Duong et al., 2017; Susanto and Lu, 2017). To our knowledge, we are the first to investigate the potential of exploiting linguistic information in a multi-encoder setup for (neural) semantic parsing.

Specifically, the aims of this chapter are to investigate (i) whether exploiting linguistic information can improve neural semantic parsing and (ii) whether it is better to include this linguistic information in the same encoder or in an additional one. We take as baseline the character-level neural semantic parser for Discourse Representation Structures (DRS, Kamp and Reyle, 1993) developed in Chapter 7. During encoding we add linguistic information in a multi-encoder setup, including various wide-spread automatic linguistic analyses for the input texts, such as lemmatization, POS-tagging, syntactic analysis and semantic tagging. We then empirically determine whether using a multi-encoder setup is preferable over merging all input features in a single encoder. We also discuss our submission to

the first shared task on DRS parsing (Abzianidze et al., 2019), based on this method and in which we apply a few postprocessing methods to improve the scores. We hope the insight gained from these experiments will provide suggestions to improve future neural semantic parsing for DRSs and other semantic formalisms.

8.2 Data and Methodology

8.2.1 Discourse Representation Structures

DRSs are formal meaning representations based on Discourse Representation Theory (Kamp and Reyle, 1993). We use the version of DRT provided in the Parallel Meaning Bank (PMB, Abzianidze et al. 2017), a semantically annotated parallel corpus, with texts in English, Italian, German and Dutch. DRSs are rich meaning representations containing quantification, negation, reference resolution, comparison operators, discourse relations, concepts based on WordNet, and semantic roles based on VerbNet. For a more extensive overview, see Section 2.1.2 and Chapter 6.

All experiments are performed using the data of the PMB. In our experiments, we only use the English texts and corresponding DRSs. We use PMB release 2.2.0, which contains gold standard (fully manually annotated) data of which we use 4,597 as train, 682 as dev and 650 as test instances. It also contains 67,965 silver (partially manually annotated) and 120,662 bronze (no manual annotations) instances. Most sentences are between 5 and 15 tokens in length. Since we will compare our results mainly to the sequence-to-sequence model of Chapter 7, we will only employ the gold and silver data.

8.2.2 Representing Input and Output

We represent the source and target data in the same way as in Chapter 7, i.e., the source sentence as a sequence of characters, with a special character indicating uppercase characters. The target DRS is also represented as a sequence of characters, with the exception of DRS operators, thematic roles and DRS variables, which are represented as *super characters* (see Chapter 4), i.e., individual tokens. The DRS variables are rewritten to a more general representation, using the De Bruijn index (de Bruijn, 1972). In a postprocessing step, the original clause structured is restored.

(a)	(b)	(c)
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <div style="border: 1px solid black; padding: 2px; margin-bottom: 2px;"> $e_1 \ x_1 \ t_1$ </div> <div style="border: 1px solid black; padding: 2px;"> $\text{time.n.08}(t_1)$ $t_1 = \text{now}$ $\text{work.v.02}(e_1)$ $\neg \text{Time}(e_1, t_1)$ $\text{Co} - \text{Agent}(e_1, x_1)$ $\text{Agent}(e_1, \text{speaker})$ $\text{male.n.02}(x_1)$ $\text{Name}(x_1, \text{tom})$ </div> </div>	<div style="border-top: 1px solid black; border-bottom: 1px solid black; padding: 5px;"> b1 REF t1 b1 EQU t1 "now" b1 time "n.08" t1 b2 Time e1 t1 b1 NOT b2 b2 REF e1 b2 Agent e1 "speaker" b2 work "v.02" e1 b2 Co-Agent e1 x1 b3 REF x1 b3 Name x1 "tom" b3 male "n.02" x1 </div>	<div style="border-top: 1px solid black; border-bottom: 1px solid black; padding: 5px;"> b2 REF x1 b2 EQU x1 "now" b2 time "n.08" x1 b1 Time x2 x1 b2 NOT b1 b1 REF x2 b1 Agent x2 x3 b1 work "v.01" x2 b1 Goal x2 x3 b1 REF x3 b1 Name x3 "tom" </div>

Figure 8.1: DRS in box format (a), gold clause representation (b) and example system output (c) for *I am not working for Tom*, with precision of 4/8 and recall of 4/9, resulting in an F-score of 47.1.

To include morphological and syntactic information, we apply a lemmatizer, POS-tagger and dependency parser using Stanford CoreNLP (Manning et al., 2014), similar to Sennrich and Haddow (2016) for machine translation. The lemmas and POS-tags are added as a token after each word. For the dependency parse, we add the incoming arc for each word. We also apply the easyCCG parser of Lewis and Steedman (2014), using the supertags.¹ Finally, we exploit semantic information by using semantic tags (Bjerva et al., 2016b; Abzianidze and Bos, 2017). Semantic tags are language-neutral semantic categories, which get assigned to a word in a similar fashion as part-of-speech tags. Semantic tags are able to express important semantic distinctions, such as negation, modals and types of quantification. We train a semantic tagger with the TnT tagger (Brants, 2000) on the gold and silver standard data in the PMB release.² Examples of the input to the model for each source of information are shown in Table 8.1.

¹We segment the supertags, e.g., $(S|NP)(S|NP)$ is represented as $(S | NP) | (S | NP)$

²This tagger is also used in the PMB pipeline, see Abzianidze and Bos (2017). It outperformed an ngram-based CRF-tagger (Lafferty et al., 2001) we also tried, obtaining an accuracy of 94.4% on the dev set.

Sent.	I	am	not	working	for	Tom
POS	PRP	VBP	RB	VBG	IN	NNP
SEM	PRO	NOW	NOT	EXG	REL	PER
LEM	I	be	not	work	for	Tom
DEP	nsubj	aux	neg	ROOT	case	nmod
CCG	NP	(S\NP) / (S\NP)	(S\NP) \ (S\NP)	(S\NP) / PP	PP / NP	N

Table 8.1: Example representations for each source of input information.

8.2.3 Neural Architecture

We employ a recurrent sequence-to-sequence neural network with attention (Bahdanau et al., 2015) and two bi-LSTM layers, similar to the models in Chapters 4 and 7. However, those models were trained with *OpenNMT* (Klein et al., 2017), which does not support multiple encoders. Therefore, we switch to a sequence-to-sequence framework that implements multiple encoders, *Marian* (Junczys-Dowmunt et al., 2018). We use model-type *s2s* (for a single encoder) or *multi-s2s* (for multiple encoders).

This architecture differs from the previous in four main ways: (i) the initial decoder hidden state is defined as the mean of all encoder states, (ii) an extra linear layer is added between this mean encoder state and the initial decoder state, (iii) a *conditional* LSTM is used instead of a regular LSTM during decoding and (iv) an extra linear layer is added after each decoder state.

Specifically, given a source sequence $(v_1, \dots, v_k)$ of length k , and a target sequence $(t_1, \dots, t_l)$ of length l , let e_i be the embedding of source symbol i , let s_i be the encoder hidden state at source position i and let d_j be the decoder state at target position j . A single forward encoder state is obtained as follows: $\vec{s}_i = \text{LSTM}(\vec{s}_{i-1}, e_i)$. The final state is obtained by concatenating the forward and backward hidden states, $s_i = [\vec{s}_i; \overleftarrow{s}_i]$. The decoder is initialized with the average over all encoder states:

$$\mathbf{c} = \left(\sum_{i=1}^l \mathbf{s}_i \right) / k \quad (8.1)$$

$$\mathbf{d}_0 = \tanh(\mathbf{W}_{\text{init}} \mathbf{c})$$

The conditional LSTM with attention is defined as follows:

$$\begin{aligned} \mathbf{d}'_j &= \text{LSTM}_1(\mathbf{d}_{j-1}, \mathbf{y}_{j-1}) \\ \mathbf{a}_j &= \text{ATT}(\mathbf{C}, \mathbf{d}'_j) \\ \mathbf{d}_j &= \text{LSTM}_2(\mathbf{d}_{j-1}, \mathbf{a}_j) \end{aligned} \quad (8.2)$$

Here, $\mathbf{C}$ is the set of all encoder states and ATT is the attention function (see Section 3.3). When using two encoders, this means that multiple inputs are encoded separately by an identical RNN, without sharing parameters. We simply concatenate the resulting (averaged) context vectors before feeding it to the decoder:

$$\mathbf{d}_0 = \tanh(\mathbf{W}_{\text{init}} [\mathbf{c}_{\text{enc1}}; \mathbf{c}_{\text{enc2}}])$$

During decoding, the calculation of final attention vector $\mathbf{a}_j$ is now dependent on both sets of encoder states:

$$\begin{aligned} \mathbf{a}_j^{\text{enc1}} &= \text{ATT}(\mathbf{C}^{\text{enc1}}, \mathbf{d}'_j) \\ \mathbf{a}_j^{\text{enc2}} &= \text{ATT}(\mathbf{C}^{\text{enc2}}, \mathbf{d}'_j) \\ \mathbf{a}_j &= [\mathbf{a}_j^{\text{enc1}}; \mathbf{a}_j^{\text{enc2}}] \end{aligned} \quad (8.3)$$

Figure 8.2: Schematic overview of our neural architecture when using two encoders, in this case characters and POS-tags.

Setting	Value	Setting	Value	Setting	Value
RNN type	LSTM	Dropout RNN	0.2	Learning rate (LR)	0.002
Enc type	bi-direc	Dropout src/tgt	0.0	LR decay	0.8
Enc/dec layers	2	Batch size	12	LR decay strategy	epoch
Embedding size	300	Optimization crit	ce-mean	LR decay start	9
RNN size	300	Vocab size src/tgt	80/150	Clip normalization	3
Epochs	15	Optimizer	adam	Beam size	10
Length norm	0.9	Label smoothing	0.1	Skip connections	True
Layer norm	True				

Table 8.2: Hyper-parameter settings for our model, found after a search on the development set. Settings not mentioned are left at default.

See Figure 8.2 for a schematic overview of our model. Our hyper-parameter settings, found after a search on the development set, can be found in Table 8.2. When only using gold data, training is stopped after 15 epochs. For gold + silver data, we stop training after 6 epochs, after which we restart the training process from that checkpoint to fine-tune on only the gold data, also for 6 epochs.³

We will compare our method of using extra encoders to represent linguistic information to adding the linguistic information in a single encoder. The common method of adding this information in one encoder is to add it to the token representation before feeding this larger vector to the LSTM, e.g., $\mathbf{e}_j = [\mathbf{e}_{\text{word}}; \mathbf{e}_{\text{pos}}]$ for adding POS-tags. This is also how the linguistic information is added in Sennrich and Haddow (2016). However, since we have a character-level model there is no single tag per individual character. Therefore, we merge the representations in the text itself, similar as we added POS-tags in Chapter 4. Table 8.3 shows examples of how the input is structured for using one or more encoders. Experiments showed that using more than two encoders drastically decreased performance. Therefore, we merge all the linguistic information in a single encoder if we want to use multiple sources of linguistic information in a multi-encoder model (see last row of Table 8.3).⁴

³We empirically found the best number of epochs during tuning, by looking at performance on the development set for the baseline model.

⁴Concatenating the linguistic information on token-level in a second encoder, e.g., $\mathbf{e}_{\text{ling}} = [\mathbf{e}_{\text{pos}}; \mathbf{e}_{\text{sem}}]$, is not supported by Marian, but we experiment with this in Chapter 9.

Source	Enc.	Representation
POS - 1 enc	Enc 1	I PRP am VBP not RB working VBG for IN Tom NNP . .
POS - 2 enc	Enc 1	I am not working for Tom .
	Enc 2	PRP VBP RB VBG IN NNP .
POS + Sem (1 enc)	Enc 1	I PRP PRO am VBP NOW not RB NOT working VBG EXG for IN REL Tom NNP PER . . NIL
	Enc 2	PRP PRO VBP NOW RB NOT VBG EXG IN REL NNP PER . NIL

Table 8.3: Example representation when using one or two encoders, for either a single source of information (POS) or multiple sources (POS + Sem) for the sentence *I am not working for Tom*. For readability purposes we show the word-level instead of character-level representation of the source words.

8.2.4 Evaluation Procedure

Produced DRSs are compared with the gold standard representations by using `Counter`. An example of the matching procedure is shown in Figure 8.1, while more details can be found in Chapter 6. The produced DRSs go through a strict syntactic and semantic validation process, as is described in Chapter 7. If a produced DRS is invalid, it is replaced by a dummy DRS, which gets an F-score of 0.0. We check whether two systems differ significantly by performing approximate randomization (Noreen, 1989), with $\alpha = 0.05$, $R = 1000$ and $F(model_1) > F(model_2)$ as test statistic for each DRS pair. This is the same procedure we applied in Chapter 7.

8.3 Results

We perform all our experiments twice: (i) only using gold data for training and (ii) with both gold (fully manually annotated) and silver (partially manually annotated) data. We differentiate between two types of experiments: *single type* experiments in which only a single source of information is added, and *stacking* experiments in which we add multiple sources of information, one by one, based on how much they improved the scores in the *single type* experiments.

Gold-only The results of the gold-only experiments are shown in Table 8.4. We clearly see that using an additional encoder for the linguistic information is superior to merging all the information in a single encoder. For two encoders, the scores increase by at least 0.7 for each source of information individually. Lemmatization shows the highest improvement, most likely because the DRS concepts that need to be produced are often lemmatized versions of the source words. When we stack the linguistic features, we observe an improvement for each addition, resulting in a final 2.7 point F-score increase over the baseline.

Gold + silver If we also employ silver data, we again observe that the multi-encoder setup is preferable over a single encoder (Table 8.5), for both isolating and stacking the linguistic features. For single type, the results are similar to only using gold data, with the exception of the semantic tags, which even hurt the performance now. Interestingly, when stacking the linguistic features, there is no improvement over only using the lemma of the source words. The lemmas give us a significant improvement of 0.9 over the baseline. Since we only use lemmas, we unfortunately do not expect that this model is particularly better at parsing longer sentences than our baseline model. In Section 8.4.3 (Figure 8.4) we find that this is indeed not the case. In summary, linguistic information mainly helps when there is limited training data, though can still be helpful when more training data is available.

(a) Gold only: single type			(b) Gold only: stacking		
Model	1 enc	2 enc	Model	1 enc	2 enc
Baseline	78.6 $\pm$ 0.6	NA	Baseline	78.6 $\pm$ 0.6	NA
POS-tags	79.5 $\pm$ 0.8	79.3 $\pm$ 0.6	+ Lemma	78.6 $\pm$ 0.4	79.9 $\pm$ 0.4
Semantic tags	79.0 $\pm$ 0.9	79.3 $\pm$ 0.4	+ Semantic tags	79.4 $\pm$ 0.6	80.5 $\pm$ 0.6
Lemma	78.6 $\pm$ 0.4	79.9 $\pm$ 0.4	+ POS tags	79.4 $\pm$ 0.3	80.8 $\pm$ 0.3
Dep. parse	78.9 $\pm$ 0.7	79.3 $\pm$ 0.8	+ CCG supertags	79.4 $\pm$ 0.6	81.0 $\pm$ 0.6
CCG supertags	78.6 $\pm$ 1.1	79.4 $\pm$ 0.9	+ Dep. parse	78.8 $\pm$ 0.7	81.3 $\pm$ 0.9

Table 8.4: Tables (a) and (b) show the results of adding *a single type* of linguistic information or *stacking* the linguistic information for using **gold** data only. Reported scores are F-scores on the development set, averaged over 5 runs of the system, with confidence scores.

(c) Gold + silver: single type			(d) Gold + silver: stacking		
Model	1 enc	2 enc	Model	1 enc	2 enc
Baseline	84.5 $\pm$ 0.3	NA	Baseline	84.5 $\pm$ 0.3	NA
POS tags	84.8 $\pm$ 0.3	84.9 $\pm$ 0.4	+ Lemma	84.0 $\pm$ 0.2	85.6 $\pm$ 0.4
Semantic tags	83.5 $\pm$ 0.6	84.0 $\pm$ 0.4	+ POS-tags	84.3 $\pm$ 0.4	85.5 $\pm$ 0.3
Lemma	84.0 $\pm$ 0.2	85.6 $\pm$ 0.4	+ CCG supertags	84.5 $\pm$ 0.2	85.6 $\pm$ 0.6
Dep. parse	83.9 $\pm$ 0.4	84.6 $\pm$ 0.3	+ Dep. parse	84.5 $\pm$ 0.2	85.4 $\pm$ 0.4
CCG supertags	83.8 $\pm$ 0.3	84.8 $\pm$ 0.5	+ Semantic tags	83.7 $\pm$ 0.4	85.1 $\pm$ 0.2

Table 8.5: Table (c) and (d) show the results of adding a *single type* of linguistic information or *stacking* the linguistic information for using **gold + silver** data. Reported scores are F-scores on the development set, averaged over 5 runs of the system, with confidence scores.

Comparison We now compare our best models to previous parsers introduced in Chapter 7⁵: the two baseline systems *Spar* and *Sim-spar*, *Boxer* (Bos, 2015) and our previous sequence-to-sequence model without linguistic features. *Boxer* (Bos, 2008b, 2015) is a DRS parser that uses a statistical CCG parser for syntactic analysis and a compositional semantics based on λ -calculus, followed by pronoun and presupposition resolution. *Spar* is a baseline system that outputs the same DRS for each test instance⁶, while *Sim-spar* outputs the DRS of the most similar sentence in the training set, based on a simple word embedding metric.⁷ The results are shown in Table 8.6. Our model clearly outperforms the previous systems, even when only using gold standard data. When compared to our previous sequence-to-sequence model (Chapter 7), retrained on the same data used in our systems, the largest improvement (3.6 and 3.5 for dev and test) comes from the improved neural architecture (see Section 8.2.3). However, the linguistic features are clearly still beneficial when using only gold data (increase of 2.7 and 1.9 for dev and test), and also still help when employing additional silver data (1.1 and 0.3 increase for dev and test, both significant).

⁵Since Liu et al. (2018a) used data from the Groningen Meaning Bank instead of the PMB, we cannot make a comparison.

⁶For PMB release 2.2.0 this is the DRS for *Tom voted for himself*.

⁷See Section 6.5 for an explanation of the high baseline scores.

	Dev			Test		
	Prec	Rec	F-score	Prec	Rec	F-score
Spar	42.3	37.9	40.0	44.4	37.8	40.8
Sim-spar	52.4	54.2	53.3	57.0	58.4	57.7
Boxer (Bos, 2015)	72.5	72.0	72.2	72.1	72.3	72.2
Seq2seq (Chapter 7)	83.5	78.5	80.9	85.0	81.4	83.2
This chapter: gold only	81.9	75.6	78.6 $\pm$ 0.6	85.1	78.1	81.5 $\pm$ 0.2
This chapter: gold only + all ling	84.3	78.5	81.3 $\pm$ 0.9	86.6	80.4	83.4 $\pm$ 0.4
This chapter: gold + silver	85.9	83.2	84.5 $\pm$ 0.3	87.4	86.0	86.7 $\pm$ 0.2
This chapter: gold + silver + lemma	86.5	84.8	85.6 $\pm$ 0.4	87.6	86.3	87.0 $\pm$ 0.4

Table 8.6: Results on the test set compared to a number of baseline parsers and our previous seq2seq model in OpenNMT (Chapter 7). Our scores are averages of 5 runs, with confidence scores.

8.4 Shared Task Participation

We also participated with our system in the first shared task on DRS parsing (Abzianidze et al., 2019). This motivated us to try and push the scores by looking at the learning curves of the model and by applying a few extra post-processing steps. Moreover, we analyse which documents were particularly easy or hard for our model, compared to the other shared task participants. These extra analyses coupled with the final shared task results are described below.

8.4.1 Learning Curves

Since manual DRS annotation is a hard and time consuming task, it is interesting to know how much we can still benefit from extra silver and gold standard examples. Concurrently, we can also observe the impact of the linguistic features across different amounts of training data. We show the learning curves of two models: the baseline model and the best model. The best model employs the linguistic features in a separate encoder.

How much gold is needed? The left graph of Figure 8.3 shows the performance of our two models when only using gold training data. It is clear that additional gold data still improves performance. This is no surprise, of

Figure 8.3: Performance of our baseline and best system for training on different amounts of gold data on the left. A similar comparison, but now for varying amounts of silver data is shown on the right.

course, since even silver data still helped improve performance. The linguistic features are beneficial for all amounts of data. Moreover, the improvements do not seem to diminish for larger amounts of gold training data.

How much silver is needed? The effect of varying the amount of silver data is shown in the right graph of Figure 8.3. The initial addition of silver data is clearly beneficial. However, the effect seems to diminish a bit for the best model after 40,000 silver instances. The baseline model, though, still improves after 40,000 instances. In general, it seems that additional silver data could be beneficial, though the extra benefit is likely to be small. Here, we also find the linguistic features (only lemmas in this case) to be beneficial. Interestingly, the linguistic features seem to have more impact for larger amounts of silver data. Presumably, the model can only learn accurate representations for the lemmas if enough data is available.

8.4.2 Improvement Methods

There are a number of syntactic and semantic requirements for a set of clauses to be considered a well-formed DRS (see Chapter 7). Among others, there should be a single main box and there should be no loop in the subordination of boxes. Since we do not restrict our model when producing clauses, these errors can occur. However, they can often be (easily) fixed, by chang-

ing a single clause or a set of clauses. We apply a few simple methods to fix these ill-formed DRSs. Moreover, we improve performance on word sense disambiguation, by changing senses that never occurred in the training set to its most frequent sense in the training set. Details of these methods, along with specific improvements per method, are shown in Appendix A. Applying all these methods on the development set gave us an improvement of 0.8, while test set performance improved by 0.4. The total number of ill-formed DRSs decreased from $\sim 1\%$ to less than 0.1%. All in all, these methods give us a modest improvement, having slightly less impact than adding sources of linguistic information. Note that, preferably, we have a model that does not need these methods, and will always produce well-formed DRSs. On the other hand, it seems unfair to punish our model with a score of 0.0 for an ill-formed DRS, if the DRS can easily be fixed.

8.4.3 Competition Results

The results on the final shared task evaluation set are shown in Table 8.7. Aside from our own two systems, there were three participating teams. Fancellu et al. (2019) proposed a graph-aware recurrent sequence model that can explicitly model graphs instead of the string representations. This reduced the number of ill-formed DRSs (especially without postprocessing), though it was quite far from state-of-the-art performance.⁸ Evang (2019) aimed to find a middle-ground between traditional symbolic approaches and the recent

	Prec.	Rec.	F1
Amr2drs	36.7	42.2	38.8
Spar	44.3	35.4	39.4
Sim-spar	55.7	53.0	54.3
Bi-LSTM graph decoder (Fancellu et al., 2019)	71.9	64.1	67.8
Stack-LSTMs (Evang, 2019)	71.9	69.9	70.9
Seq2seq bi-LSTM (Chapter 7)	80.8	78.6	79.7
Seq2seq bi-LSTM + ling (this chapter)	85.5	83.6	84.5
Seq2seq Transformer (Liu et al., 2019b)	84.8	84.8	84.8

Table 8.7: Official results of the shared task for the 5 participating systems.

⁸Their submitted system here is not the final version described in Fancellu et al. (2019), though also there they do not outperform our model from Chapter 7.

neural (sequence-to-sequence) models. They employed a transition-based parser that relied on explicit word-meaning pairs that are found in the training set. Parsing decisions are made based on vector representations of parser states, which are encoded using stack-LSTMs. Lastly, Liu et al. (2019b) created a system based on our model of Chapter 7, but swapped the bi-LSTM architecture for a Transformer (Vaswani et al., 2017) and also trained on bronze data, which increased performance. This is the highest scoring system, though the difference between their work and our current model is not significant.

Figure 8.4: Performance of the 5 shared task systems over document length. Noord18 is the model of Chapter 7, while Noord19 is our current work.

We deem it interesting to compare the participating systems over document length. One could hypothesise that the more principled approaches of Fancellu et al. (2019) and Evang (2019) do better on longer documents, while the character-level seq2seq models have trouble handling longer sequences. The comparison shown in Figure 8.4 (taken from Abzianidze et al., 2019) tells us that, so far, this is not the case. All systems seem to degrade more or less equally over document length. On the other hand, a document of 12 tokens (the largest length for which at least 8 documents were available) is still quite short, so perhaps for longer documents we would see a different trend.

Relatively good performance	Relatively poor performance
(a) Tom died when he was 97.	(f) These bananas are not ripe.
(b) I read comic books.	(g) A book about dancing is lying on the desk.
(c) We should drink 64 ounces of fluids a day.	(h) Approximately seven billion people inhabit our planet.
(d) I look down on liars and cheats.	(i) The trip will take approximately five hours.
(e) You can't run away.	(j) Tom was too tired to speak.

Table 8.8: Sentences for which our model performed well and poor.

8.4.4 Error Analysis

Participating systems were provided with five sentences in the evaluation set for which their model did relatively well, and five for which the model performed relatively poor. The sentences for our system are shown in Table 8.8.⁹ Interestingly, the model does well on sentences containing numbers, (a) and (c), but fails to capture the correct interpretation of *approximately* in (h) and (i). For (b) and (d), our model correctly identified the multi-word expressions `comic_book` and `look_down_on`, perhaps due to the character-level input. For (e), our model produced a perfect DRS, while other approaches had trouble either producing `run_away`, or the fact that the sentence was addressed to a hearer. Sentence (j) is interpreted in the gold standard as *Tom could not speak because he was tired*, but our model produced a DRS more similar to the meaning of *Tom spoke and was tired*.

8.5 Conclusions

In this chapter we have shown that a range of linguistic features can improve performance of character-level sequence-to-sequence models for the task of parsing Discourse Representation Structures. We have shown empirically that the best method of adding these features is by using a multi-encoder setup, as opposed to merging the sources of linguistic information in a single encoder. This method can potentially also be beneficial for other semantic parsing tasks in which sequence-to-sequence models do well. We also participated in the first shared task on DRS parsing, for which we improved our model by devising a number of repairing strategies for ill-formed DRSs. We ended second out of five teams in the competition.

⁹DRSs available at: <https://urd2.let.rug.nl/~rikvannoord/DRS/IWCS/>

PART IV

Putting Characters into Context

CHAPTER 9

Combining Characters with Contextual Embeddings

The introduction of large, pretrained contextual language models has had an enormous impact on the field of Natural Language Processing (NLP). The use of these representations improved performance on virtually all tasks they were tried on. This leads us to ask the question: are character-level representations still useful in this new era of NLP? In this chapter we argue that the answer to this question is a clear yes. We combine character-level and contextual language model representations to improve performance on Discourse Representation Structure and Abstract Meaning Representation parsing. Character representations can easily be added in a sequence-to-sequence model in either one encoder or as a fully separate encoder, with improvements that are robust to different formalisms, language models, languages and data sets. For English DRS parsing, these improvements are larger than adding individual sources of linguistic information or adding non-contextual embeddings. A new method of analysis based on semantic tags demonstrates that the character-level representations improve performance on a subset of selected semantic phenomena.

Chapter is an extended version of:

van Noord, R., Toral, A., and Bos, J. (2020). Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *EMNLP*, pages 4587–4603, Online. ACL

9.1 Introduction

Character-level models have obtained impressive performance on a number of natural language processing (NLP) tasks, ranging from the classic POS-tagging (Santos and Zadrozny, 2014) to complex tasks such as open domain semantic parsing, as we saw in the previous chapters. However, this was before the introduction of large pretrained language models (Peters et al., 2018; Devlin et al., 2019). These models solved the issue of static word embeddings always assigning the same vector to a source word, independent of the context this source word occurred in. Moreover, these models are not just useful for initializing the word vectors of a system. The trained weights of the models themselves can be used during training of a different task, in a process that is called *fine-tuning* (Howard and Ruder, 2018).

It is hard to understate the impact of these models on the field of NLP. The most famous model `Bert`¹ (Devlin et al., 2019) already outperformed the state-of-the-art on 11 NLP tasks by a wide margin, and in general, state-of-the-art performance is currently virtually always obtained by fine-tuning a pre-trained language model (e.g., Conneau et al., 2020). It inspired a new line of research that tries to understand what these models actually learn (BERTology, see Rogers et al. 2020), as well as multiple even larger models, with better performance (e.g., Yang et al. 2019; Raffel et al. 2020).

Does this mean that, despite a long tradition of being used in language-related tasks (see Section 3.5), character-level representations are no longer useful? We try to answer this question by looking at semantic parsing, specifically Discourse Representation Structure (DRS, Kamp and Reyle, 1993) and Abstract Meaning Representation (AMR, Banarescu et al., 2013) parsing. We aim to answer the following research questions:

1. Do pretrained language models (LMs) outperform character-level models for DRS and AMR parsing?
2. Can character and LM representations be combined to improve performance, and if so, what is the best method of combining them?
3. How do these improvements compare to adding linguistic features?

¹To give an impression: `Bert` has already been cited over 8,500 times (as of August 2020) since it first got posted on arXiv in October 2018.

4. Are the improvements robust across different pretrained language models, languages, and data sets?
5. On what type of sentences do character-level representations specifically help?

Why semantic parsing? Semantic parsing is the task of automatically mapping natural language utterances to interpretable meaning representations. The produced meaning representations can then potentially be used to improve downstream NLP applications (Issa et al., 2018; Song et al., 2019; Mihaylov and Frank, 2019), though the introduction of large pretrained language models has shown that explicit formal meaning representations might not be a necessary component to achieve high accuracy. However, it is now known that these models lack reasoning capabilities, often simply exploiting statistical artifacts in the data sets, instead of actually *understanding* language (Niven and Kao, 2019; McCoy et al., 2019). Moreover, Ettinger (2020) found that the popular `Bert` model completely failed to acquire a general understanding of negation. Related, Bender and Koller (2020) contend that meaning cannot be learned from form alone, and argue for approaches that focus on grounding the language (communication) in the real world. We believe formal meaning representations therefore have an important role to play in future semantic applications, as semantic parsers produce an explicit model of a real-world interpretation.

9.2 Background

9.2.1 Transformers & BERT

The LSTM-based sequence-to-sequence model we used in the previous chapters has a clear bottleneck: we can only calculate the model's internal state s_t *after* we calculated s_{t-1} . This is not very efficient, while also making long-range dependencies harder to model. This problem was addressed by the Transformer (Vaswani et al., 2017), which has no recurrent (or convolutional) layers. Transformers are sequence-to-sequence models that mainly rely on *self-attention* to do the learning, which can be calculated fully in parallel. The remaining issue was that this architecture was insensitive to the order of the input sequence (quite a problem when processing language),

but this was solved by adding positional embeddings, which only purpose is to indicate the position of a word in the sentence. We will compare the performance of this architecture with our bi-LSTM model.²

The Transformer is also the backbone of the `Bert` contextual language model (Devlin et al., 2019). `Bert` is trained using Transformers on masked language modelling, i.e., it is tasked to predict words that are obscured in the input sentence. This method of language modelling allowed them to train a bi-directional model, which, in ablation experiments, was shown to be an important factor for the improved performance. They combine the masked language modelling task with *next sentence prediction*, i.e., the model is given two sentences *A* and *B* and has to predict whether *B* followed *A*.³ In essence, these tasks were picked so that the model learns representations of language in general, which can be useful for a number of downstream applications. A 3.3 billion word corpus (books and Wikipedia) was used for training the model. In fact, they trained two models: one with 12 Transformer layers (`Bert-base`) and one with 24 layers (`Bert-large`).

There are two ways to use these trained `Bert` models. For one, we can use the representations as *features*, that is, we run the full model on our input sentence, and take the representations of the last hidden layer as the input vectors for our sequence-to-sequence model. These representations are *contextual*, i.e., the vector of a word can differ based on the context of the sentence it occurs in. We can go one step further, though, and also keep the full model (with all the weights) during the training of our own task-specific model. The pretrained `Bert` weights themselves are then also updated based on the current loss, instead of only the new weights of our own model. This process is called *fine-tuning* and is the method that resulted in the best performance in the original `Bert` paper. Using this fine-tuning method, they considerably improved performance on a large variety of NLP tasks. We will experiment with both methods in this chapter.

²We will not provide a detailed description of the Transformer model in this thesis. For such a description, we refer to the original paper (Vaswani et al., 2017) or this excellent blog post: <http://www.peterbloem.nl/blog/transformers>

³A more efficient alternative for masked language modelling might be replaced token detection (Clark et al., 2020), while sentence order prediction was shown to be preferable over next sentence prediction (Lan et al., 2020).

9.2.2 Character-level Models

As we saw in Section 3.5, the power of character-level representations has long been known in the field. In earlier work, they were successfully used across a range of tasks in statistical models (e.g., Sejnowski and Rosenberg, 1987; Church, 1993; Klein et al., 2003). More recently, they also proved useful as input representations for neural networks, starting with success in general language modelling (Sutskever et al., 2011), but also for a variety of other tasks, including tokenization (Evang et al., 2013), POS-tagging (Santos and Zadrozny, 2014), dependency parsing (Ballesteros et al., 2015) and neural machine translation (Chung et al., 2016a).

In previous chapters, we successfully trained a character-level end-to-end sequence-to-sequence-model for AMR and DRS parsing. However, in semantic parsing, character-level representations are also often used in combination with non-contextual word-level representations (e.g., Ballesteros and Al-Onaizan, 2017; Groschwitz et al., 2018; Cai and Lam, 2019). In recent work, they are also employed in combination with `Bert` (Zhang et al., 2019a,b; Cai and Lam, 2020), by using a char-CNN (Kim et al. 2016, also see Section 9.3.1). However, only Zhang et al. (2019a) provide an ablation score without the characters, and it is not clear if their small improvement is significant.

9.2.3 Discourse Representation Structures

DRSs are formal meaning representations introduced by Discourse Representation Theory (Kamp and Reyle, 1993) with the aim to capture the meaning of texts (Figure 9.1). Many variants of DRS have been proposed throughout the years. We adopt Venhuizen et al. (2018)’s version of DRT, which is close to Kamp’s original ideas, but has a neo-Davidsonian view of event semantics and explicitly represents presuppositions. The Groningen Meaning Bank (GMB, Basile et al., 2012b; Bos et al., 2017) and the Parallel Meaning Bank (PMB, Abzianidze et al., 2017) are releases of open domain texts annotated with DRS. We will be using the PMB data in this chapter, since it contains gold standard evaluation sets and covers languages for which the distinction between characters and words is meaningful (English, German, Italian and Dutch). See Section 2.1.2 for a more detailed description of DRS and Chapter 6 for more information on the PMB.

b1 NEGATION b2	b3 REF x1
b1 REF t1	b3 Name x1 "boston"
b1 TPR t1 "now"	b3 PRESUPPOSITION b2
b1 time "n.08" t1	b3 city "n.01" x1
b2 REF e1	b2 Start e1 t2
b2 Theme e1 "speaker"	b2 REF t2
b2 Time e1 t1	b2 time "n.08" t2
b2 be "v.03" e1	b2 Location e1 x1
b2 YearOfCentury t2 "2013"

Figure 9.1: Example DRS in clause (top) and box (bottom) representation.

DRS parsing Early approaches to DRS parsing employed rule-based systems for small English texts (Johnson and Klein, 1986; Wada and Asher, 1986; Bos, 2001). The first open domain DRS parser is Boxer (Bos, 2008b, 2015), which is a combination of rule-based and statistical models. Le and Zuidema (2012) used a probabilistic parsing model that used dependency structures to parse GMB data as graphs. More recently, Liu et al. (2018a) proposed a neural model that produces (tree-structured) DRSs in three steps by first learning the general (box) structure of a DRS, after which specific conditions and referents are filled in. In follow-up work (Liu et al., 2019a) they extend this work by adding an improved attention mechanism and constraining the decoder to ensure well-formed output. This model achieved impressive performance

on both sentence-level and document-level DRS parsing based on GMB data. Fu et al. (2020) in turn improve on this work by employing a Graph Attention Network during both encoding and decoding.

The introduction of gold standard DRSs in the PMB enabled a principled comparison of approaches. In Chapter 7, we showed that sequence-to-sequence models can successfully learn to produce DRSs, with characters as the preferred representation. In Chapter 8, we improved on these scores by adding linguistic information. The first shared task on DRS parsing (Abzianidze et al., 2019) sparked more interest in the topic, with a system based on stack-LSTMs (Evang, 2019) and a neural graph-based system (Fancellu et al., 2019). The best system (Liu et al., 2019b) used a similar approach as our model of Chapter 7, but swapped the bi-LSTM encoder for a Transformer. We will compare our approach to these models in Section 9.4.

9.2.4 Abstract Meaning Representations

Abstract Meaning Representation (AMR, Banarescu et al. 2013) is a broad-coverage semantic formalism that aims to capture sentence level semantics. AMRs are rooted, directed, acyclic graphs that encode, among others, information about semantic relations, coreference, named entities (often grounded with Wikipedia links) and negation. In AMR graphs, the nodes are *concepts*, while the edges are the (semantic) *relations*. There is no specific alignment between concepts in the graph and words in the sentence. An example AMR is shown in Figure 9.2 (left). For a more detailed overview of AMR, see Section 2.1.1.

Initial AMR parsers either used two-stage graph-based methods to split the task in relation and concept identification (Flanigan et al., 2014; Werling et al., 2015; Flanigan et al., 2016; Zhou et al., 2016) or a transition-based parser based on shift-reduce dependency parsing (Wang et al., 2015b; Damonte et al., 2017). These two approaches are still dominant, though commonly extended by using powerful neural networks. Lyu and Titov (2018) use bi-LSTM networks in their graph-based method and extend the model by explicitly modelling alignment between concepts and words. Zhang et al. (2019a) also use a two-stage method that treats AMR parsing as sequence-

Figure 9.2: Example AMR in graph format on the left. The same AMR after applying graph recategorization and anonymization on the right (see Section 9.3.2).

to-graph transduction, which is not reliant on alignments. Transition-based methods were improved by using stack-LSTMs (Ballesteros and Al-Onaizan, 2017; Naseem et al., 2019), better alignment methods (Liu et al., 2018b) and using a different, more compact graph representation (Guo and Lu, 2018).

Groschwitz et al. (2018) introduced a new graph-based method based on the AM-algebra (Groschwitz et al., 2017). This work views AMRs as consisting of atomic graphs, which can be combined by using linguistically motivated operations. These structures can be seen as dependency trees, allowing them to borrow techniques from neural dependency parsing to produce the AMRs. Follow-up work improved on their scores by using multi-task learning on different graph-banks and using contextual word embeddings (Lindemann et al., 2019). The current best approaches combine parts of the two-stage and transition-based methods by doing incremental sequence-to-graph transduction (Zhang et al., 2019b; Cai and Lam, 2019, 2020; Zhou et al., 2020). In these models, the final graph is built incrementally in a top-down fashion, by spanning one node at every step, each decision being informed by previous decisions of the model. We will compare to these systems in Section 9.4.2.

9.3 Method

In this section we describe our neural architectures: the baseline sequence-to-sequence model and the two methods of adding character-level representations. We then describe the data sets under consideration, along with the sources of linguistic information we will be employing. Lastly, we outline the evaluation metrics and statistical tests used in our experiments. All our code is released at https://github.com/RikVN/Neural_DRS.

9.3.1 Neural Architecture

As our baseline system, we use a sequence-to-sequence model with attention (Bahdanau et al., 2015), with modifications based on the Nematus (Sennrich et al., 2017) and Marian (Junczys-Dowmunt et al., 2018) architectures. This is described in more detail in Chapter 8, but shortly: (i) we initialize the decoder hidden state with the mean of all encoder states, (ii) we add an extra linear layer between this mean encoder state and the initial decoder state and (iii) we add an extra linear layer after each decoder state. We re-implement this model in AllenNLP (Gardner et al., 2017), as there is no option to use contextual embeddings in Marian.

More formally, given a source sequence $(s_1, \dots, s_l)$ of length l , and a target sequence $(t_1, \dots, t_k)$ of length k , let $\mathbf{e}_i$ be the embedding of source symbol i , let $\mathbf{h}_i$ be the encoder hidden state at source position i and let $\mathbf{d}_j$ be the decoder state at target position j . A single forward encoder state is obtained as follows: $\vec{\mathbf{h}}_i = \text{LSTM}(\vec{\mathbf{h}}_{i-1}, \mathbf{e}_i)$. The final state is obtained by concatenating the forward and backward hidden states, $\mathbf{h}_i = [\vec{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$. The decoder is initialized with the average over all encoder states: $\mathbf{c}_{\text{tok}} = (\sum_{i=1}^l \mathbf{h}_i) / l$ and $\mathbf{d}_0 = \tanh(\mathbf{W}_{\text{init}} \mathbf{c}_{\text{tok}})$. We will experiment with adding character information in either one or two encoders. The implementations are outlined below.

Characters in one encoder For one encoder, we use char-CNN (Kim et al., 2016), which runs a Convolutional Neural Network (LeCun et al., 1990) over the characters for each token. It applies convolution layers for certain *widths*, which in essence select n -grams of characters. For each width, it does this a predefined number of times, referred to as the number of *filters*. An overview of the network for encoding the word *have* is shown in Figure 9.3.

Figure 9.3: Overview of the char-CNN encoder, encoding the word *have* with bigrams (width = 2) for n filters.

A width of 2 selects the bigrams *ha*, *av* and *ve*, returning a scalar for each bigram operation, which in turn form a vector f_1 for filter 1. We then take the max value of this vector to obtain the first value of our width 2 (w_2) char-CNN embedding e_{w2_1} . The final vector e_{w2} is thus of length n . However, we usually do not look at only a single width, but at a range of widths, e.g., $[1, 2, 3, 4, 5]$. In that case, we simply concatenate the resulting vectors to obtain our final char-CNN embedding: $e_{\text{char}} = [e_{w1}; e_{w2}; e_{w3}; e_{w4}; e_{w5}]$. Each width-filter combination has independent learnable parameters, which get updated by the loss and optimization function, similar to all other network parameters. Finally, the char-CNN embedding is concatenated to the token-level representation, which is fed to the encoder: $e_i = [e_{\text{tok}_i}; e_{\text{char}_i}]$.

This method has two main advantages. For one, it is able to specifically model n -grams of characters instead of only individual characters. Second, the characters do not influence the total number of encoder states (still token-level), which can grow quite large for character-level encoders. On the other hand, it introduces a lot of new parameters, which increase the size and complexity of the model. Moreover, it increases the number of hyper-parameters (filters, widths, character embedding size), which is a costly extra tuning step, and might make the model more prone to overfitting.

Characters in two encoders Our two encoder set-up is the same as described in Chapter 8, but with a contextual language model as first encoder, and the characters in a second encoder. We run separate (but structurally identical) bi-LSTM encoders over the tokens and characters, and concatenate the resulting context vector before we feed it to the decoder:

$$\mathbf{d}_0 = \tanh(\mathbf{W}_{\text{init}} [\mathbf{c}_{\text{tok}}; \mathbf{c}_{\text{char}}])$$

In the decoder, we replace the LSTM with a doubly-attentive LSTM, based on the doubly-attentive GRU (Calixto et al., 2017). We apply soft-dual attention (Junczys-Dowmunt and Grundkiewicz, 2017) to be able to attend over both encoders in the decoder (also see Figure 9.4):

$$\begin{aligned} \mathbf{d}'_j &= \text{LSTM}_1(\mathbf{d}_{j-1}, \mathbf{e}_{t_{j-1}}) \\ \mathbf{a}_j &= [\text{ATT}(\mathbf{C}_{\text{tok}}, \mathbf{d}'_j); \text{ATT}(\mathbf{C}_{\text{char}}, \mathbf{d}'_j)] \\ \mathbf{d}_j &= \text{LSTM}_2(\mathbf{d}'_j, \mathbf{a}_j) \end{aligned}$$

Here, $\mathbf{e}_{t_{j-1}}$ is the embedding of the previously decoded symbol t , C the set of encoder hidden states for either the tokens or characters, ATT the attention function (dot-product) and $\mathbf{d}_j$ the final decoder hidden state at step j . This model can easily be extended to more than two encoders, which we will experiment with in Section 9.4.

This type of multi-source model is commonly used to represent different languages, e.g., in machine translation (Zoph and Knight, 2016; Firat et al., 2016) and semantic parsing (Susanto and Lu, 2017; Duong et al., 2017), though it has also been successfully applied in multi-modal translation

Figure 9.4: Schematic overview of our neural architecture when using two encoders (Bert and characters).

(Libovický and Helcl, 2017), multi-framework semantic parsing (Stanovsky and Dagan, 2018) and adding linguistic information (Currey and Heafield, 2018 and our own work in Chapter 8). To the best of our knowledge, we are the first to represent the characters as a source of extra information (when combined with a contextual language model) in a multi-source sequence-to-sequence model.

Transformer We also experiment with the Transformer model (Vaswani et al., 2017), using the stacked self attention model as implemented in AllenNLP. A possible advantage of this model is that it might handle longer sentences and documents better. However, it might be harder to tune (Popel and Bojar, 2018)⁴ and its improved performance has mainly been shown for large data sets, as opposed to the generally smaller semantic parsing data sets (Section 9.3.3). Indeed, we cannot outperform the LSTM architecture (see Section 9.4), even when tuning more extensively. We therefore do not experiment with adding character-level representations to this architecture, though the char-CNN can be added similarly as for the LSTM model.

⁴Also see: https://twitter.com/srush_nlp/status/1245825437240102913

	DRS parsing			AMR parsing	
Setting	LSTM	Transf.	Range	LSTM	Range
Hidden RNN size	300	NA	200 - 600	300	300 - 700
Decoder RNN size	300	NA	300	300	300
Num heads	NA	6	2, 4, 6, 10	NA	NA
hidden_dim	NA	300	300 - 600	NA	NA
ff_hidden_dim	NA	900	300 - 1,200	NA	NA
dropout: layer	NA	0.1	0.1, 0.2	NA	NA
residual	NA	0.2	0.1, 0.2	NA	NA
attention	NA	0.1	0.1, 0.2	NA	NA
target emb dim	300	300	300 (GloVe)	300	300 (GloVe)
max src tokens	125	50	30 - no max	100	100
max tgt tokens	1,160	560	300 - no max	300	300
layers	1	6	1-3 LSTM, 1-10 Tr.	1	1, 2, 3
max_norm	3	3	3, 4, 5	3	3
scale_grad_by_freq	False	False	True/False	False	True/False
label_smoothing	0.0	0.1	0.0, 0.05, 0.1, 0.2	0.1	0.0 - 0.2
beam size	10	10	10	10	10
max decoding steps	1,000	500	500, 1000	300	300
scheduled sampling	0.2	0.0	0.0, 0.1, 0.2, 0.3, 0.4	0.0	0.0, 0.1, 0.2
batch size	48	32	12, 24, 32, 48, 64, 128	32	16 - 128
optimizer	adam	adam	adam, sgd	adam	adam, sgd
learning rate	0.001	0.0002	0.0001 - 0.01	0.001	0.0001 - 0.01
grad norm	0.9	0.9	0.7 - 0.95	0.9	0.7 - 0.95
min target occ	3	3	1, 3, 5, 10, 20	5	1, 3, 5, 10, 20

Table 9.1: An overview of the hyper-parameters used for the LSTM and Transformer architecture, that use the `Bert-base` representations. Settings not specified are left at their default value.

Hyper-parameters To make a fair comparison, we conduct an independent hyper-parameter search on the development set for all eight input text representations (see Section 9.3.2) across the two neural architectures, starting from the settings we found in Chapter 8. We found that the best settings were very close for all systems, with the only notable difference that the learning rate of the Transformer models is considerably smaller than for the bi-LSTM models (0.0002 vs 0.001). Table 9.1 gives an overview of the hyper-parameters we used and/or experimented with in the tuning stage for both formalisms. This table only gives an overview of the settings for the `Bert-base` model, though the settings for the other representations (de-

scribed in 9.3.2) are very similar.⁵ We performed manual tuning, with between 10 and 40 tuning runs for each representation type and model combination (see Table 9.4).

For the char-CNN model, we use 100 filters, a character embedding size of 75 and n-gram filter sizes of [1, 2, 3] for English and [1, 2, 3, 4, 5] for German, Italian and Dutch. For experiments where we add characters or linguistic features, the only extra search we do is the size of the hidden vector of the RNN encoder (300 – 600 for DRS, 500 – 700 for AMR), since this vector now has to contain more information, and could potentially benefit from a larger size. Note that (possible) improved performance is not simply due to larger model capacity, since during tuning of the baseline models a larger RNN hidden size did not result in better performance.

9.3.2 Representations

We will experiment with five well-known pretrained language models: `Elmo` (Peters et al., 2018), `Bert` base/large (Devlin et al., 2019) and `Roberta` base/large (Liu et al., 2019c).⁶ `Elmo` calculates contextual embeddings based on a bi-LSTM language model, while `Roberta` was designed as an improved version of `Bert` by making small changes in its training regime. The performance of these five large LMs is contrasted with results of a character-level model and a token-based model with either non-contextual `GloVe` (Pennington et al., 2014) or `FastText` (Grave et al., 2018) embeddings.

DRS parsing Pre- and postprocessing of the DRSs is done using the steps described in Chapter 7. The DRSs are linearized, after which the variables are rewritten to a relative representation based on the order of their introduction. The character-level model has character representations for the DRS concepts and constants, but not for variables, roles and operators. For all word-level models, the DRS concepts are initialized with `GloVe` embeddings, while the other target tokens are learned from scratch.

⁵Specific configuration files are available here: https://github.com/RikVN/Neural_DRS/

⁶We are aware that there exist several other large pretrained language models (e.g., Yang et al., 2019; Raffel et al., 2020; Clark et al., 2020), but we consider that the models we used have had the largest impact on the field.

AMR parsing The AMRs are pre- and postprocessed by the methods described in Chapter 4, extended by using the indexing method from Chapter 5 to handle coreference. We use character-level representations, with super characters at the target side. Wiki links are removed before training and restored in a postprocessing step, following Bjerva et al. (2016a). There are a few differences with the work in Chapter 4. We do not use ensembling or averaging of models. Also, we do not add POS-tags as features. Lastly, since the target side is now word-based, we separate the concepts from its sense during training to make learning easier, e.g., *say-01* is represented as *say* **SPLIT** -01. The senses are then restored in a postprocessing step.

AMR recategorization Previous AMR parsers were often reliant on *graph recategorization* to achieve high accuracy (e.g., Werling et al., 2015; Konstas et al., 2017; Lyu and Titov, 2018; Zhang et al., 2019a). This method merges specific subgraphs of an AMR to a new single compound node. Often, this method also *anonymizes* parts of the graph and input sentence to deal with the sparsity of named entities, e.g., changing *Germany* to *Country_1*. These recategorizations are then restored in a postprocessing step. An example recategorized graph (with corresponding sentence) is shown in Figure 9.2 (right). This approach depends on a large number of handcrafted rules, which are non-trivial and differ between approaches, making it hard to distinguish modelling advancements from simply using a better set of rules (Cai and Lam, 2019). We would prefer a model that is not dependent on these methods, but we will test with both settings to assess their impact. We use the graph recategorization method of Zhang et al. (2019a).⁷

BERT specifics For the `Bert` models, we obtained the best performance by only keeping the vector of the first WordPiece per original token (e.g., only keep `play out of play ##ing`). For `Roberta`, it was best to use the WordPiece tokenization as is. Since linguistic features are added on token level, we duplicate the semantic tags for multi-piece tokens of `Roberta` in Table 9.6. We found that for both `Bert` and `Roberta`, it was best to keep the pretrained weights frozen (`requires_grad = False`).⁸ This was not a small difference:

⁷<https://github.com/sheng-z/stog>

⁸Using the last hidden layer as input features. We did not experiment with taking a different layer or taking a (pooled) mix of different layers.

models using fine-tuning obtained scores in the range of 45–60. Interestingly, the above findings were the same for both DRS and AMR parsing.

9.3.3 Data and Evaluation

DRS data We use PMB releases 2.2.0 and 3.0.0⁹ in our DRS parsing experiments (see Table 9.2). The latter is a larger and more diverse extension of 2.2.0, which will be used for most of our experiments. We use 2.2.0 to compare to previous work and to verify that our results are robust across data sets. The PMB releases contain DRSs for four languages (English, German, Italian and Dutch) for three levels of annotation: gold (fully manually checked), silver (partially manually corrected) and bronze (no manual corrections). To make a fair comparison to previous work, we only employ the gold and silver data, by pretraining on gold + silver data and subsequently fine-tuning on only the gold data. If there is no gold train data available, we train on silver + bronze and fine-tune on silver. Unless otherwise indicated, our results are on the English development set of release 3.0.0.

		Gold			Silver	Bronze
		Train	Dev	Test	Train	Train
2.2.0	English	4,597	682	650	67,965	120,662
	German	0	727	747	4,235	102,998
	Italian	0	374	400	2,515	61,504
	Dutch	0	370	341	1,051	20,554
3.0.0	English	6,620	885	898	97,598	146,371
	German	1,159	417	403	5,250	121,111
	Italian	0	515	547	2,772	64,305
	Dutch	0	529	483	1,301	21,550

Table 9.2: Document statistics over the four languages, for the two PMB releases considered.

AMR data Similar as in Chapters 4 and 5, we use AMR release LDC2017T10 in our experiments. It contains 36,521 training, 1,368 development and 1,371 test AMRs. This release only contains English data.

⁹<https://pmb.let.rug.nl/data.php>

Data filtering We filtered ill-formed DRSs from the PMB data sets, which only occurs for silver and bronze data ($< 0.1\%$ of DRSs). Similarly, we filter ill-formed AMRs when using silver data, which is only $< 0.01\%$ of AMRs. For the bi-LSTM models, the filtering of source and target tokens (see Table 9.1) only filters out three very large documents from training for DRS parsing. This was done for efficiency and memory purposes; it did not make a difference in terms of F1-score. For the Transformer model, this improved F1-score by around 0.5. We apply a similar filtering for AMR parsing.

Linguistic features We want to contrast our method of character-level information to adding sources of linguistic information. We employ the same five sources as in Chapter 8: part-of-speech tags (POS), dependency parses (DEP), lemmas (LEM), CCG supertags (CCG) and semantic tags (SEM). For the first three sources, we use Stanford CoreNLP (Manning et al., 2014) to parse the documents. The CCG supertags are obtained by using easyCCG (Lewis and Steedman, 2014). For semantic tagging, we train our own trigram-based tagger using TnT (Brants, 2000). Table 9.3 shows a tagged example sentence for all five sources of information. Moreover, we also include non-contextual GloVe and FastText embeddings as an extra source of information.

We add these sources of linguistic information in the same way as we add the character-level information, in either one or two encoders (see Section 9.3.1). In two encoders, we use the exact same architecture. For one encoder, we (obviously) do not use the char-CNN, but learn a separate embedding for the tags (of size 200), that is then concatenated to the token-level representa-

Sent	I	have	n't	been	to	Boston	since	2013
POS	PRP	VBP	RB	VBN	TO	NNP	IN	CD
SEM	PRO	NOW	NOT	EXT	REL	GPE	REL	YOC
LEM	I	have	not	be	to	Boston	since	2013
DEP	nsubj	aux	neg	cop	case	ROOT	case	nmod
CCG	NP	VP\VP	VPVP	VP/PP	PP/NP	N	(VP\VP)/NP	N

Table 9.3: Example representation for each source of linguistic information (PMB document p00/d1489).

tion, i.e., $\mathbf{e}_i = [\mathbf{e}_{\text{tok}_i}; \mathbf{e}_{\text{ling}_i}]$. If we use two encoders with a LM, characters *and* linguistic information (e.g., Table 9.5), the characters are added separately in the second encoder, while the LM and linguistic information representations are added in the first encoder.

DRS evaluation We compare the produced DRSs to the gold standard using `Counter`¹⁰ (see Chapter 6), which calculates micro precision, recall and F1-score based on the number of matching clauses. We use an updated version of Referee (see Chapter 7) to ensure that the produced DRSs are syntactically and semantically well-formed (i.e., no free variables, no loops in subordinate relations) and form a connected graph. All shown scores are averaged F1-scores over five training runs of the system, in which the same five random seeds are used.¹¹ Reported scores are calculated using `Counter`, unless otherwise indicated. For significance testing we use approximate randomization (Noreen, 1989), with $\alpha = 0.05$ and $R = 1000$.

Moreover, we also evaluate our final models using the recently proposed metric D-scorer (Liu et al., 2020). This metric converts the DRSs to graphs, after which an F1-score is calculated based on n-gram overlap, similar to BLEU (Papineni et al., 2002). Since there are no human judgements for DRS pairs available we do not know if this metric better captures DRS similarity than `Counter`, but it will be interesting to know whether our findings will be robust across different metrics.

We also introduce and release **drs-jury**.¹² This program provides a detailed overview of the performance of a DRS parser, but can also compare experiments, possibly over multiple runs. Features include significance testing, semantic tag analysis (Section 9.5.1), sentence length plotting (Section 9.5.2), new detailed `Counter` scores (Appendix B), and analysing which DRSs were relatively easy or hard for the respective models (Appendix C). We hope this is a step in the direction of a more principled way of evaluating DRS parsers.

AMR evaluation The produced AMRs are evaluated by using `Smatch` (Cai and Knight, 2013), which also calculates micro precision, recall and F1-score, in this case based on matching triples (see Section 4.2.1). Note, though, that

¹⁰https://github.com/RikVN/DRS_parsing/

¹¹Standard deviations are omitted for brevity, though usually in range 0.1 – 0.6.

¹²Available at https://github.com/RikVN/Neural_DRS/

a few bugs were fixed in `Smatch` over the years, resulting in slightly higher scores now than reported in previous papers. In Table 9.10, we report both the reported score as well as the current score, wherever possible. Moreover, we evaluate our final models based on four new metrics. First, we look at SemBleu (Song and Gildea, 2019), also a metric based on BLEU, that can catch higher-order relations instead of only node-to-node correspondences. Second, we use `s2match` (Opitz et al., 2020), an extension to `Smatch` that can account for graded similarity of concepts by using static word embeddings. Lastly, we compare our models with `smatch-weighted` and `smatch-core` (Cai and Lam, 2019), which either gives higher weight to triples that compose the core idea of the graph (`smatch-weighted`), or only takes those triples into account (`smatch-core`). These methods were introduced to evaluate models on their ability to capture the *core semantics* of a given sentence. We apply the same method of significance testing as for DRS parsing.

9.4 Results

In this section we will discuss the results of adding character-level representations to contextual language models on DRS parsing and AMR parsing. For DRS parsing, we compare the impact of characters to using linguistic features and test the robustness across multiple languages. For AMR parsing, we analyze the impact of adding more silver standard data, as well as different sources of silver standard data.

9.4.1 DRS Results

LMs vs char-level models DRS parsing is no exception to the general trend in NLP: it is indeed the case that pretrained language models outperform models based only on characters (Table 9.4). Interestingly, the Transformer model has worse performance for all representations.¹³ Surprisingly, we find that `Bert-base` is the best model, though there is no significant difference with `Bert-large`. We choose to use `Bert-base` in further experiments (referred to as `Bert`).

¹³The Transformer models were even tuned longer, since they were more sensitive to small hyper-parameter changes.

	bi-LSTM	Transformer
char	86.1	79.7
GloVe	85.4	84.6
FastText	85.5	84.0
Elmo	87.3	84.3
Bert-base	87.6	85.4
Bert-large	87.5	84.7
Roberta-base	87.0	82.7
Roberta-large	86.8	81.9

Table 9.4: **DRS parsing**: Baseline model for the eight input representations considered, for the bi-LSTM and Transformer architectures. Best score in each column shown in bold.

	No chars		+ characters		
	1-enc	2-enc	1-enc	2-enc	3-enc
Bert	87.6	–	88.1	88.1	–
Bert + GloVe	87.9	87.2	88.1	88.0	86.9
Bert + FastText	87.8	87.7	87.9	87.9	87.0
Bert + pos	87.6	87.6	87.4	87.6	87.8
Bert + sem	87.9	88.0	88.0	88.4	88.1
Bert + lem	87.8	88.0	88.1	88.0	87.4
Bert + dep	87.9	87.5	88.0	87.8	87.8
Bert + ccg	87.8	87.3	87.9	87.8	87.6

Table 9.5: **DRS parsing**: Results for adding characters, linguistic information and a combination of the two to the bi-LSTM Bert-base model.

Adding characters to BERT We can see the impact of adding characters to Bert in the first row of results in Table 9.5. For both methods, it results in a clear and significant improvement over the Bert-only baseline, 87.6 versus 88.1. Of course, in an absolute sense these improvements are modest, but given that the Bert model improved only by 1.5 over the char-only model, we consider an improvement of 0.5 to be a meaningful difference.

Adding linguistic features to BERT However, another common method of improving performance is adding linguistic features to the token-level representations. We try a range of linguistic features (described in Section 9.3.3), that are added in either one or two encoders. We see in the first two columns of results of Table 9.5 that even though linguistic information sources indeed

do improve performance (up to 0.4 absolute), there is no single source that can beat adding just the character-level representations (88.1).

Combining characters and linguistic features An obvious follow-up question is whether we still see improvements for character-level models when also adding linguistic information. In a single encoder, adding characters (third column of results in Table 9.5) is beneficial for 6 out of 7 linguistic sources (i.e., compare to the first column of results). The scores are, however, not higher than simply adding characters on their own, suggesting that linguistic features are not always beneficial if character-level features are also included. For two encoders, the pattern is less clear, but we do find our highest score thus far when we combine characters and semantic tags (88.4).¹⁴ Since using three encoders did not yield clear improvements over two encoders, we do not experiment with using more than three encoders.

Robustness to different LMs We want to verify that the character improvements are robust to using different language models (Table 9.6). We see that adding characters results in improvement for all the LMs under consideration, even for `Elmo`, which already incorporates characters in creating the initial embeddings. Moreover, combining characters and semantic tags also results in an improvement over just using characters for all the LMs considered.

	No char	+char (1 enc)	+char (2 enc)	+char +sem (2 enc)
Elmo	87.3	87.6	87.8	88.0
Bert-base	87.6	88.1	88.1	88.4
Bert-large	87.5	88.2	87.8	88.3
Roberta-base	87.0	87.3	87.8	88.0
Roberta-large	86.8	86.8	87.0	87.3

Table 9.6: **DRS parsing**: Results on 3.0.0 English dev of five LMs for adding characters and both characters and semtags.

¹⁴With gold semantic tags (ceiling performance) we score 88.6.

Figure 9.5: **DRS parsing**: Dev and test scores (F1) for the four models we trained for three languages (German, Italian and Dutch). For 2.2.0, we compare our results to Fancellu et al. (2019).

Robustness across languages We train DRS parsing systems for German, Italian and Dutch for four models: char-only, Bert-only, Bert + char in 1 encoder, and Bert + char in two encoders.¹⁵ The Bert model we use is bert-multilingual-uncased. The results for both releases are shown in Figure 9.5. For all languages, adding characters leads to a clear improvement for both one and two encoders, though for Dutch the improvement is smaller than for German and Italian. For 3.0.0, the char-only model is even on par with the Bert-only model for German and Italian. Interestingly, the two-encoder setup seems to be generally preferable for these smaller, non-English data sets. For 2.2.0, we outperform the system of Fancellu et al. (2019) for German and Italian and obtain competitive scores for Dutch. An interesting direction for future work is to check whether these conclusions hold on language-specific Bert models (e.g., de Vries et al., 2019).

Comparison to previous work To check whether the improvements hold on unseen data, we run our best models on the test set and compare the scores to previous work (first two columns with content in Table 9.7).¹⁶ We see that adding the character-level information has similar (significant) improvements for dev and test on both data sets. The addition of semantic tags might be questionable: for 2.2.0, both the Bert + char models outperform this model, while for 3.0.0 the 0.1 improvement over Bert + char in one encoder is not significant. Despite this, we reach state-of-the-art performance on both data sets, significantly outperforming the previous best scores by Liu et al. (2019b) and our model from Chapter 8. We also compare to the semantic parser Boxer, which needs input for 6 different PMB layers (Abzianidze et al., 2017). *Amateur Boxer* is trained with internal PMB taggers, while *Pro Boxer* uses the output of a neural multi-task learning system based on Bert (van der Goot et al., 2020). Even though this is an unfair comparison to our system, since the rule-based components of Boxer are (partly) optimized on the dev and test sets, our best model still improves slightly over *Pro Boxer* (significantly on test).

¹⁵We do not train a model that uses semantic tags as features, since there is not enough gold semantic tag data available to train a good tagger for any of these languages.

¹⁶See Appendix B for the detailed Counter scores.

Model	Counter				D-scorer			
	2.2.0		3.0.0		2.2.0		3.0.0	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Amateur Boxer	72.2	72.2	78.2	78.8	53.1	57.5	61.1	66.9
Pro Boxer	–	–	88.2	88.9	–	–	76.8	82.1
Fancellu et al. (2019)	–	76.4	–	–	–	–	–	–
Evang (2019)	74.4	74.4	–	–	63.7	65.8	–	–
Char seq2seq (Chapter 7)	81.2	83.3	84.3	84.9	70.0	74.6	74.1	76.2
Char seq2seq + ling (Chapter 8)	86.5	86.8	86.8	87.7	74.7	79.5	78.0	80.1
Liu et al. (2019b)	85.5	87.1	–	–	–	80.9	–	–
This chapter - Bert	85.4	87.9	87.6	88.5	76.3	81.7	79.5	82.0
This chapter - Bert + char (1 enc)	86.1	88.3	88.1	89.2	77.4	81.9	80.4	82.6
This chapter - Bert + char (2 enc)	85.6	88.1	88.1	89.0	77.1	81.7	80.0	82.7
This chapter - Best model	85.5	87.7	88.4	89.3	76.8	81.2	80.6	82.8

Table 9.7: **DRS parsing**: comparison of our four main models to previous work for PMB 2.2.0 and 3.0.0 (English only). We use the metrics Counter (Chapter 6) and D-scorer (Liu et al., 2020).

We also compare performance of the models using the D-scorer metric (Liu et al., 2020). Generally, we see that the results hold up: using `Bert` improves performance over just characters, while using characters in combination with `Bert` results in modest improvements as well. The main difference is that on the 2.2.0 test set, combining `Bert`, characters and semtags has worse performance than just using `Bert`.

DRS parsing on GMB data The most recently developed neural DRS parsers (Liu et al., 2018a, 2019a; Fu et al., 2020) only evaluate on the GMB, making a comparison difficult. However, we consider it highly likely that their more structured models (based on the coarse-to-fine approach of Dong and Lapata, 2018) would outperform our more vanilla sequence-to-sequence models. Therefore, even though we reach state-of-the-art PMB performance, we consider the improvement of character-level representation the main takeaway of these experiments. Since the GMB models also employ an encoder-decoder framework, our character-level method could potentially be used to improve those systems.

9.4.2 AMR Results

Let us now turn to AMR parsing. Here, we also first select the best language model out of the `Elmo`, `Bert` and `Roberta` models (shown in Table 9.8). These experiments only used gold standard data. Interestingly, we see the same pattern as for DRS parsing: `Bert-base` is the best model, while the `Roberta` models have worse performance. The models also clearly outperform the char-only model from Chapter 4, as well as the word-level models combined with static embeddings. We will use the `Bert-base` model in further experiments. We did a lot of tuning experiments with the Transformer model for both `Bert-base` and `Roberta-base`, but could not score higher than 62.0 (`Bert-base`). Therefore, we do not consider running more Transformer experiments a good use of our resources, and only focus on the bi-LSTM architecture for the rest of our experiments.

Representation	bi-LSTM
char-only (Chapter 4)	65.2
GloVe	62.1
FastText	62.2
Elmo	66.7
Bert-base	69.5
Bert-large	68.5
Roberta-base	67.1
Roberta-large	66.2

Table 9.8: Baseline model scores for **AMR parsing** with our bi-LSTM sequence-to-sequence model, when trained only on the gold standard data.

Adding characters to BERT We now test our two methods of adding character-level representations here as well. We do this for our baseline model as well as for models that exploit additional training data, both with and without the recategorization method of Zhang et al. (2019a). This is shown in Table 9.9. In general, adding characters in one encoder significantly improves performance for all models, except for the self-training model with graph recategorization. This was the case even when the model had access to additional training data. Moreover, adding this data does not

Model	Bert-base baseline			Bert-base + recat		
	Bert	+char (1 enc)	+char (2 enc)	Bert	+char (1 enc)	+char (2 enc)
Gold-only	69.5	70.3	69.6	71.5	71.9	71.6
Gold + 100k self-training	70.5	71.0	70.4	72.1	72.3	71.7
Gold + 100k (Wang et al., 2015b)	72.0	72.5	71.4	74.6	74.9	74.5
Gold + 100k (Lindemann et al., 2019)	73.2	73.6	72.4	76.0	76.3	75.9
Gold + 100k (Zhang et al., 2019a)	72.9	73.1	72.3	75.2	75.7	75.4

Table 9.9: **AMR parsing**: F-scores for using characters in combination with our seq2seq Bert-base model. Scores are compared for gold data only and different sources of silver data.

seem to diminish the effect of adding characters, with consistent improvements of 0.2 – 0.5 for the one encoder model. The recategorization method clearly helps as well, with improvements between 1.3 and 3.5 F-score points. Again, even with this method, we still see an improvement for adding character-level representations in one encoder. On the other hand, using two encoders does not seem to improve performance over the baseline model. This might be due to the longer AMR sentences as compared to the shorter sentences in the PMB.

Self-training We also employ *self-training*, i.e., we obtain 100,000 silver AMRs by parsing them with our best gold-only model (both baseline and re-categorized). The advantage of this approach is that we are not dependent on an external parser. Using self-training does not work as well as using silver data from a different source, unfortunately, though we still get a meaningful and significant improvement over the gold-only baseline. Using silver AMRs from higher quality AMR parsers (Zhang et al., 2019a; Lindemann et al., 2019) is preferable over using the Wang et al. (2015b) AMRs from Chapter 4, though the difference is smaller than one would perhaps expect (see Table 9.9). Notably, we do not outperform the models that provided us with the silver data, as opposed to the model in Chapter 4. Therefore, we will discard the models that are dependent on previous models, and only consider the self-training models going forward.

Figure 9.6: **AMR parsing**: F-scores for self-training with different amounts of silver data for the *Bert-base* bi-LSTM with recategorization. Scores are on the development set and averaged over 5 runs.

Limits of silver data First, we want to know whether we can improve the self-training process by employing more silver data. If so, we want to know if adding characters is still beneficial, or if this improvement disappears. We add silver data in batches of 50k until no further improvement is reached. This is shown in Figure 9.6. The model can clearly still benefit from more silver data. We reach the highest baseline score for 350k silver data (73.2), at which point adding characters in one encoder significantly improves the score by 0.3 (73.5). We consider this model as our final model. In general, adding characters is beneficial for different amounts of silver data, though the differences are small.

Performance on test set In the bottom 8 rows of Table 9.10 we show the performance of 8 of our models on the test set. We see that the improvements of using character-level representations hold up when using silver data, graph recategorization, or both. These improvements, though small, are also consistent across the different metrics under consideration. Our best model obtains a final *Smatch* F-score of 73.9.

Model	GR	S-paper	S-cur	S-w	S-core	s2match	SemBleu
Cai and Lam (2020)	✓	80.2	80.6	78.1	77.0	81.6	64.1
Cai and Lam (2020)	✗	78.7	79.0	77.5	76.2	80.0	63.1
Zhou et al. (2020)	✓	77.5	77.9	74.6	73.5	79.0	61.3
Zhang et al. (2019b)	✓	77.0	77.3	74.1	72.9	78.9	60.7
Zhang et al. (2019a)	✓	76.3	76.6	73.4	72.3	77.9	59.7
Lindemann et al. (2019)	✓	76.3	76.7	73.6	72.7	77.8	56.7
Naseem et al. (2019)	✓	75.5	75.9	72.3	71.5	77.3	57.0
Lyu and Titov (2018)	✓	74.4	75.2	67.1	67.3	76.3	53.6
Cai and Lam (2019)	✗	73.2	73.6	71.3	70.3	75.0	54.0
Groschwitz et al. (2018)	✓	71.0	71.4	67.9	67.0	72.7	50.9
Char seq2seq + 100k (Chapter 4)	✗	71.0	71.3	69.0	67.7	72.6	52.3
BERT seq2seq gold-only	✗	-	68.9	67.7	68.9	70.1	46.8
BERT seq2seq gold-only + char (1e)	✗	-	69.7	68.7	66.7	70.9	47.3
BERT seq2seq gold-only	✓	-	71.3	69.4	67.8	72.7	50.0
BERT seq2seq gold-only + char (1e)	✓	-	71.8	69.8	68.1	73.0	50.4
BERT seq2seq + 350k	✗	-	70.7	70.0	68.0	71.9	47.1
BERT seq2seq + 350k + char (1e)	✗	-	71.0	70.4	68.2	72.2	47.4
BERT seq2seq + 350k	✓	-	73.6	71.9	70.3	74.7	52.9
BERT seq2seq + 350k + char (1e)	✓	-	73.9	72.0	70.5	75.0	53.3

Table 9.10: **AMR parsing**: comparison of our best model to the previous best systems. **S-paper** is the F-score reported in the respective papers, **S-cur** the score for the current version of `Smatch`. **S-w** is `smatch-weighted`, while **GR** is short for graph recategorization.

Comparison to previous work We compare our final models to the best previous systems on the test set (Table 9.10).¹⁷ Our `Bert` based `seq2seq` model does not come close to state-of-the-art performance. However, if we do not want to rely on the rule-based graph recategorization, our system is only outperformed by the recent work of Cai and Lam (2019, 2020). Interestingly, the end-to-end character-level `seq2seq` model from Chapter 4 outperforms the Lyu and Titov (2018) model for `smatch-weighted` and `smatch-core`, even though there is a considerable difference in general `Smatch` score. We see a similar trend when comparing with

¹⁷Lindemann et al. (2019) reported an F-score of 75.3 in the paper, but later fixed a bug in postprocessing that improved their score to 76.3. We report the latter score.

Groschwitz et al. (2018): no difference in original `Smatch` score, but a clear difference for the “core semantics” metrics. In other words, even though the `seq2seq` model does not make linguistically informed decisions, that does not necessarily mean it will do worse on capturing the core meaning of a sentence. A possible explanation is that the extensive graph recategorization mainly helps in capturing triples that are not vital to the full meaning representation (e.g., named entities), which is not reflected in the general `Smatch` score.

9.5 Analysis

In this section we analyze on what (semantic) phenomena the characters improve performance the most by introducing a new analysis method based on semantic tags (Section 9.5.1). Moreover, we evaluate the performance of the parsers over document length, and try to estimate performance over longer documents. For DRS parsing, we evaluate on a newly constructed set of higher quality silver standard documents (Section 9.5.2).

9.5.1 Semantic Tag Analysis

We deem it insightful to find out on what type of sentences and semantic phenomena the character representations are the most beneficial. We introduce a novel method of analysis: selecting subsets of sentences based on the

Modality	NOT	NEC	POS																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										</
----------	-----	-----	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

Table 9.11: Semantic tags that were used to select sentences that contain a certain phenomenon. The example sentence in Table 9.3 is included in the categories *Modality*, *Pronouns*, *Named Entities* and *Numerals*.

occurrence of certain semantic tags. In the PMB release, each token is also annotated with a semantic tag, which indicates the semantic properties of the token in the given context (Abzianidze and Bos, 2017). This allows us to easily select all sentences that contain certain (semantic) phenomena and evaluate the performance of the different models on those sentences. For AMR parsing, we use the semantic tagger described in Section 9.3.2 to get the list of semantic tags per document.

Table 9.11 shows which semtags were used per phenomenon. The corresponding F-scores per phenomenon for our best models are shown in Table 9.12. Our best DRS parsing model (+ch+sem) has the best performance on six of the seven phenomena selected, even though the differences are small. For AMR parsing, adding characters in one encoder outperformed the baseline model on all phenomena selected. In general, the character-level representations seem to help across the board: the +char models improve on the baseline (Bert) in almost all instances.

	DRS parsing					AMR parsing			
	# Docs	Bert	+char (1 enc)	+char (2 enc)	+ch+sem (2 enc)	# Docs	Bert	+char (1 enc)	+char (2 enc)
All sentences	1,783	88.1	88.7	88.5	88.8	2,739	73.2	73.5	73.1
Modality	188	86.8	+0.1	+0.1	+0.4	697	69.4	+0.7	+0.5
Negation	156	88.8	+0.2	-0.1	+0.4	469	67.2	+0.5	+0.7
Possibility	38	81.3	0.0	+1.0	+1.5	251	69.5	+1.2	+1.0
Necessity	13	74.5	-1.6	+1.4	-0.2	103	67.3	+1.0	+0.2
Logical	449	86.3	+0.7	+0.2	+0.5	1,676	71.2	+0.4	0.0
Pronouns	996	88.9	+0.4	+0.4	+0.6	1,123	68.6	+0.3	+0.3
Attributes	1,063	87.6	+0.7	+0.4	+0.8	2,576	72.9	+0.3	0.0
Comparatives	45	84.5	+1.6	+0.2	-0.2	576	68.5	+0.8	+0.5
Named entities	673	88.1	+0.5	+0.3	+0.6	1,890	74.0	+0.2	0.0
Numerals	186	85.8	+1.1	+1.2	+1.5	1,001	72.7	+0.4	+0.2

Table 9.12: F-scores on subsets of sentences that contain a certain phenomenon, based on semantic tags. For DRS parsing, scores are for the combined English dev and test set of PMB release 3.0.0. For AMR parsing, the scores are also for the combined dev and test set. Absolute scores shown for Bert and relative differences for the remaining systems.

For *Numerals* and *Named Entities* we expected the characters to help specifically, since (i) Bert representations might not be as optimal for all individual numerals (Wallace et al., 2019), and (ii) the character representations might attend more to capital letters, which often indicates the presence of a named entity. Relatively speaking, the character representations clearly help for *Numerals*, but less so for *Named Entities*.

9.5.2 Sentence Length Analysis

We are also interested in finding out which model performs well on longer documents. When the Transformer model was introduced, one of the advantages was less decrease in performance for longer sentences (Vaswani et al., 2017). Also, since Boxer is partly rule-based and not trained in an end-to-end fashion, it might be able to handle longer sentences better. Figure 9.7 shows the performance over sentence length for seven of our DRS parsing systems.

Figure 9.7: **DRS parsing**: F-scores over document length (tokens) on the combined English dev and test set of 3.0.0. X-axis shows the document length (top) and the number of documents for that length (bottom).

We see a similar trend for all models: a decrease in performance for longer sentences. We also create a regression model that predicts F-score, with as predictors parser and document length in tokens, similar to the procedure described in Chapter 7. We do not find a significant interaction of any model with sentence length, i.e., none of the models decreases significantly less or more than any other model.

To get some idea how well our models would do on longer (possibly multi-sentence) documents, we create a new evaluation set for DRS parsing, as all gold standard documents are relatively short. We select all silver documents with 15 or more and less than 51 tokens that have at least the semtagging or CCG layer marked as gold standard. This resulted in a set of 128 DRSs, which should contain the higher quality silver documents. We retrain our models with those sentences removed and plot the performance over sentence length in Figure 9.8. Note that in this case we do not perform fine-tuning on

Figure 9.8: **DRS parsing:** F-scores over document length (tokens) on the silver standard evaluation set of longer documents. X-axis shows the sentence length bins (top) and the number of documents for that length (bottom).

the gold standard data. Also, we run Counter with the `-default-sense` setting (not punishing models that get the word sense wrong), since the word senses of the evaluation set are not gold standard. This has a similar increase of around 1.0 for all models. We see that performance still decreases for longer sentences, though not as much after 30 tokens per document. The Transformer model does not seem to catch up with the bi-LSTM models, even for longer documents. The addition of characters is still beneficial for longer documents, though only in one encoder.

For AMR parsing, we do not have to create a new evaluation set, as the development and test set contain quite a few longer sentences. The performance over sentence length is plotted in Figure 9.9. Interestingly, the character representations seem to help *more* for longer documents, though the differences are relatively small. Moreover, there is no indication that the longer sequence length hurts performance of the two-encoder model.

Figure 9.9: **AMR parsing:** F-scores over document length (tokens) on the combined dev and test set. X-axis shows the sentence length bins (top) and the number of documents for that length (bottom).

9.5.3 Discussion

We found that character-level representations generally improved performance, though we did not find a clear preference for either the one-encoder or two-encoder model. We believe that given the better performance of the two-encoder DRS parsing model on the fairly short documents of the non-English languages (see Figure 9.5), this model is likely the most useful in semantic parsing tasks with single sentences, such as SQL parsing (Zelle and Mooney, 1996; Iyer et al., 2017; Finegan-Dollak et al., 2018). The one encoder char-CNN model, on the other hand, has more potential for tasks with longer sentences/documents, such as UCCA (Abend and Rappoport, 2013) and DRS parsing on GMB data (Bos et al., 2017; Liu et al., 2018a, 2019a). The latter model also has more potential to be applicable for other (semantic parsing) systems as it can be applied to all systems that form a token-level representation from a document. In this sense, we hope that our findings here are also applicable for other, more structured, encoder-decoder models developed for semantic parsing (e.g., Yin and Neubig, 2017; Krishnamurthy et al., 2017; Dong and Lapata, 2018; Liu et al., 2019a).

A curious finding is that, for both AMR and DRS parsing, `Bert-base` outperformed the larger `Bert-large` and `Roberta` models. In both cases, it was also preferable to use `Bert` only as initial token embedder, instead of fine-tuning using the full model. This serves as some indication that certain NLP tasks cannot be solved by simply training ever larger language models. Moreover, the Transformer model did not improve performance for both tasks, while being harder to tune as well. We are a bit hesitant with drawing strong conclusions here, though, since we only experimented with a vanilla Transformer, while other extensions (e.g., Dehghani et al., 2019; Guo et al., 2019; Press et al., 2020) might be more promising for smaller data sets.

9.6 Conclusions

We performed a range of experiments on Discourse Representation Structure and Abstract Meaning Representation Parsing using neural sequence-to-sequence models, in which we vary the neural representation of the input documents. We show that, not surprisingly, using pretrained contextual language models is better than simply using characters as input (**RQ1**). However, characters can still be used to improve performance, in both a single encoder and two encoders (**RQ2**). The improvements are higher than using individual sources of linguistic information, and performance still improves in combination with these sources (**RQ3**). The improvements are also robust across different formalisms, languages models, languages and data sets (**RQ4**) and improve performance across a range of semantic phenomena (**RQ5**). These methods should be applicable to other semantic parsing and perhaps other natural language analysis tasks.

PART V

Conclusions

CHAPTER 10

Conclusions

Semantic parsing is the task of automatically assigning formal, structured meaning representations to natural language utterances. These representations can then potentially be used to improve human-computer communication. What are now considered traditional semantic parsing approaches rely on linguistic resources, such as manually constructed grammars, syntactic analyses and predefined lexicons. In this thesis, we used a neural network *sequence-to-sequence* model that is not dependent on any such resource; it can automatically produce meaning representations by only looking at the input text. We performed a large number of experiments on the graph-based semantic formalism Abstract Meaning Representation (AMR, Banarescu et al., 2013) and the more expressive Discourse Representation Structures (DRS, Kamp and Reyle, 1993). Specifically, we posed seven research questions, which we are now ready to answer.

RQ1: Can neural sequence-to-sequence models be used to produce accurate meaning representations?

The answer to this question is a clear yes. Throughout our semantic parsing experiments on AMR and DRS, our sequence-to-sequence models achieved good performance, even without being dependent on any additional resources.

RQ2: How can we best represent the input texts for these models: characters or words?

Perhaps surprisingly, we found characters to be the preferred representation in our experiments. For AMR there was quite a margin to word-level models, while for DRS this was modest, but still significant. Moreover, experiments using a representation that is in between words and characters (byte-pair encoding) did not achieve better performance than simply using characters. However, this was before the introduction of contextual language models (see **RQ7**).

RQ3: To what extent can we improve performance by employing silver standard data?

Again, we obtained similar results for AMR and DRS: additional silver data was clearly beneficial in both cases. This was especially the case when the data came from a different source than our own system, but also when using a self-training pipeline. Moreover, we found it to be a vital component for reaching state-of-the-art performance for AMR parsing. We believe this puts some work on multi-task semantic parsing (e.g., Peng et al., 2017a; Hershovich et al., 2018; Lindemann et al., 2019) in perspective, as multi-task learning is in essence also a data augmentation technique. Perhaps the often modest benefits of multi-task learning will diminish if a parser has access to large amounts of silver data.

RQ4: How can we best deal with the variables in the meaning representations?

The variables in a meaning representation represent a significant challenge to sequence-to-sequence models. While the variable names are arbitrary and meaningless, they are vital to correctly modelling semantic phenomena such as coreference, pronouns and control constructions. For AMR, we proposed a novel method to handle variable re-entrancies, but found a more simple method to be preferable (Chapter 5). For DRS (Chapter 7), we obtained best performance by rewriting the variables to a more general representation based on De Bruijn indexing (de Bruijn, 1972).

RQ5: AMR has a well-established evaluation method. Can we construct a similar method to evaluate Discourse Representation Structures?

In Chapter 6 we presented our DRS evaluation metric that is based on the AMR evaluation metric (Cai and Knight, 2013). This metric calculates an F-score based on the number of matching clauses and is the current standard metric for DRS parsing (Abzianidze et al., 2019). The clause matching system also allowed for the calculation of a number of more detailed scores, which can give more insight in parser performance than only a single F-score. This is outlined in Chapters 7 and 9. The downside of this metric is that it assigns equal weight to all clauses, while, intuitively, certain clauses should carry more weight than others (e.g., negation versus tense).

RQ6: Can we improve neural semantic parsers by injecting linguistic knowledge?

The injection of linguistic information might help our model achieve better performance, especially on longer sentences. This is not straightforward for character-level neural models, however, as we cannot simply concatenate the information at token-level. In Chapter 8, we presented a neural method that is able to exploit linguistic information in a second encoder. In our DRS parsing experiments, this outperformed simply adding the information in the input text. This information was mainly beneficial for the model that only used gold standard data.

RQ7: Can we combine character-level representations with representations from a contextual language model to improve neural semantic parsing? If so, what is the best method of combining them?

The rise of contextual language models in NLP has been undeniable. Not surprisingly, we found that using representations based on these models resulted in better performance than only using character-level representations (Chapter 9). However, for both AMR and DRS, we found that character-level representations can still be useful. We found modest, but consistent, improvements across different formalisms, language models, languages and data sets. In a detailed analysis, we found that character-level representations improved performance across a range of semantic phenomena.

Reflection

This thesis took a different approach than most research in semantic parsing. Instead of developing a new model for a specific formalism, we took a successful, generally applicable model and applied it to two considerably different open domain semantic parsing tasks. We then mostly focused on how to improve the accuracy of this model, with experiments regarding input representation, variable representation, silver standard data and linguistic knowledge. We hope that, this way, our conclusions will be more generally applicable (perhaps also outside semantic parsing) and will continue to be relevant even if a newer, fancier model has conquered NLP. To an extent, this already happened throughout the course of this thesis, with the introduction of the Transformer model, though the vanilla variant of this model did not outperform our model yet (Chapter 9).

Research on AMR parsing is now at a critical stage. The most accurate AMR parser (Cai and Lam, 2020) more or less reached human-level performance. There seems to be no need to develop an even better parser. It is time to look at the original goal of AMR: being useful for downstream applications. There is exploratory work that showed the potential of AMR in this regard for summarization (Hardy and Vlachos, 2018; Liao et al., 2018) and paraphrase detection (Issa et al., 2018), though they did not make use of contextual language models (LMs). Even if it turns out to be hard to improve on using these LMs, these downstream tasks will make for an interesting *black box* (Bos, 2008a) evaluation suite for AMR, which could tell us more than a single F-score.

This focus on downstream applications could also shed light on what semantic formalism is more generally useful for representing meanings of natural language texts. We can list the respective advantages and disadvantages of AMR, DRS and other semantic formalisms (e.g., Abend and Rappoport, 2013; White et al., 2016), but at the end of the day, we do not *know* which formalism is leading us in the most promising direction. Evaluating semantic parsers on downstream applications will give us some indication of the general usefulness of a formalism, at the same time providing insight in which formalism is best suited for which type of task.

Appendices

A Postprocessing Improvement Methods

In Chapter 8, we proposed a number of postprocessing methods to improve performance on DRS parsing. They are described in detail below.

Removing clauses One of the problems of the neural model is that it can get stuck in a loop producing (more or less) the same output. It does not happen often, but if it does, it can hurt performance quite a bit, since all the wrongly produced clauses hurt precision. We apply two straightforward methods to fix these instances. For one, we remove all clauses after clause number 75. Second, we remove clauses of concepts, roles and operators (except `REF`) if they occur more than three times. We only look at the second argument of a clause (the identifier), for example, if a full DRS contains five `Theme` clauses, we remove the last two, no matter the other values in those clauses.

No main box found If there is no main box found, this means that there are multiple independent boxes. For two independent boxes, we first try to remedy this by changing a single discourse variable in one of these boxes. We change a discourse variable that is unique in **b1** to a unique variable in **b2** (and vice versa), to establish a connection between the boxes. For each of these possible changes we check whether the DRS is well-formed now, and if so, return the new DRS. For multiple independent boxes, and if the previous method failed for two independent boxes, we start merging boxes together (e.g. replace each occurrence of **b1** by **b2**), until we find a well-formed DRS. If this does not result in a well-formed DRS, we return a dummy DRS.

Subordinate relation has a loop This error can occur if boxes indirectly subordinate themselves, e.g. **b0** subordinates **b1**, **b1** subordinates **b2** and **b2** subordinates **b0**. To solve this, we first try to merge the offending box with each of the other boxes in the DRS. If this does not work, we try to remove the offending box from the DRS. If the DRS is still ill-formed, we start the process again if the error is *Subordinate relation has a loop* (but now with the offending box removed) or apply the previously described fix for *No main box found*. A non-matching dummy DRS is returned if the DRS is still ill-formed after these steps.

Fixing senses In Chapter 7 we showed that the neural model often produced the wrong word sense for the correct concept. It even often outputs senses that were never observed in the training set. We apply a simple method to fix these instances. If a concept + sense is not present in the gold standard training set, we replace the sense by the most frequent sense for this concept in the training set. For example, we change `grow.v.01` to `grow.v.07` and `fast.n.02` to `fast.a.02`. Note that this method does not influence whether a DRS is well-formed or not.

Negative result We also tried to exploit the n-best list the parser outputs when using beam search. We simply took the first well-formed DRS in the list and considered that to be the final DRS. We found that the repairing strategies performed better, with almost no improvement for this method. Therefore, this method was discarded in further experiments.

Improvement Results We can check by how much the scores in Section 8.3 improve if we apply these methods. The results of adding the improvement methods incrementally are shown in Table A.1. We see that simply removing clauses returns only modest gains, but fixing ill-formed DRSs gives a substantial improvement, even for our best model. The improvements are similar with and without using linguistic information. The method for fixing word senses also proved quite effective, improving the final F-score by 0.2 to 0.4. When we apply these improvements on the test set, our best model improves from 87.0 to 87.4.

	Initial		+ Removing clauses		+ Solving ill-formed		+ Fixing senses	
	ill (%)	F1	ill (%)	F1	ill (%)	F1	ill (%)	F1
Gold-only baseline	2.6	78.6	2.6	78.8	0.2	79.5	0.2	79.8
Gold-only + ling	2.7	81.3	2.7	81.4	0.1	82.2	0.1	82.4
Gold + silver	1.5	84.5	1.5	84.5	0.0	85.1	0.0	85.4
Gold + silver + ling	0.9	85.6	0.9	85.7	0.0	86.1	0.0	86.4

Table A.1: Impact of incrementally adding the new postprocessing methods on the development set results of Section 8.3.

B Detailed Scores

Tables B.1, B.2, B.3, B.4, B.5, B.6, B.7 and B.8 show the detailed F-scores for the English, German, Italian and Dutch development and test sets of release 2.2.0 and 3.0.0. *Infreq. sense* is the F-score on all concept clauses that were not the most frequent sense for that word in the training set (e.g., `be.v.01`, `like.v.03`), which can only be reliably calculated for English. This gives us an idea of how well the model does on word-sense disambiguation. *Perfect sense*, on the other hand, is the F-score when we ignore word senses during matching, i.e. `be.v.01` can match with `be.v.02`. The last 9 rows are not in the original detailed Counter scores, but are produced by **drs-jury**. We observe that character-level representations help to produce fewer ill-formed and more perfect DRSs, especially on 3.0.0.

In Tables B.9, B.10 and B.11 we show F-scores per individual clause type, for the English development and test of set of release 3.0.0, as calculated by **drs-jury**.¹ The models perform quite well on all operators (Table B.9), without any considerable differences between the models. For the individual roles in Table B.10, it is interesting to note that the models have some trouble with `Theme`, even though it is the second most frequent role in the data set. On the other hand, the scores for `Degree` and `Unit` are surprisingly high, given their moderate frequency. Here, we can also observe an effect of the character-level representations: the models that employ them get considerably higher scores on `ClockTime` and `YearOfCentury`, likely because the characters can represent individual numbers. Lastly, in Table B.11 we see the F-scores of individual concepts. The models indeed learn to perform word-sense disambiguation, as the scores for the three senses of `be` are still relatively high. An effect of the characters might be visible for the scores of `measure.n.02`, perhaps because measurements are often indicated by using numbers.

¹Scores for the non-English languages and release 2.2.0 can be easily generated by following the instructions here: https://github.com/RikVN/Neural_DRS/blob/master/DRS_jury.md

	PMB release 2.2.0 - English							
	Development set				Test set			
	bert	+ch (1e)	+ch (2e)	+ch +sem	bert	+ch (1e)	+ch (2e)	+ch +sem
Prec	87.3	87.8	87.4	87.6	89.8	89.9	89.9	89.5
Rec	83.6	84.4	83.6	83.5	86.2	86.7	86.4	86.0
F1	85.4	86.1	85.5	85.5	87.9	88.3	88.1	87.7
Operators	94.7	95.2	94.7	94.4	94.8	94.7	94.4	94.7
Roles	88.0	88.4	88.2	88.0	90.3	90.3	90.5	89.8
Concepts	83.9	84.5	84.0	84.8	87.4	87.9	87.6	87.4
Nouns	90.8	91.5	91.1	91.4	92.4	92.8	92.4	92.5
Verbs	65.6	65.4	64.8	67.6	75.7	76.4	76.3	75.5
Adjectives	70.4	74.0	72.7	71.5	70.9	72.3	70.8	71.5
Adverbs	90.0	67.7	83.3	63.3	70.0	71.7	73.3	61.0
Events	66.7	67.3	66.5	68.4	74.8	75.7	75.4	74.7
Perfect sense	87.3	88.1	87.6	87.4	89.3	89.7	89.5	89.1
Infreq. sense	50.5	50.5	46.7	52.3	57.2	58.3	58.8	59.1
F1 std dev	0.30	0.30	0.17	0.05	0.22	0.22	0.16	0.19
F1 confidence interval	85.0	85.6	85.2	85.4	87.6	88.0	87.9	87.5
	85.8	86.5	85.7	85.5	88.2	88.6	88.3	88.0
# illformed	0.4	0.0	0.2	0.2	0.2	0.0	0.2	0.0
# perfect (avg)	235.4	237.4	239.0	239.8	267.0	265.8	266.4	267.2
# perfect (all 5)	180	187	183	188	206	213	212	205
# zero (avg)	4.4	3.4	4.2	4.2	1.6	1.8	1.2	1.8
# zero (all 5)	4	3	3	3	1	1	0	1
# same (all 5)	368	398	379	384	356	368	361	352

Table B.1: Detailed Counter scores for our models on the **English** dev and test sets of release 2.2.0. All scores are averages of 5 runs.

	PMB release 2.2.0 - German					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	71.3	73.5	74.1	73.7	75.2	76.3
Rec	62.6	65.6	65.9	64.9	66.9	67.8
F1	66.6	69.3	69.8	69.0	70.8	71.8
Operators	73.1	78.2	81.0	74.0	76.3	79.2
Roles	69.9	71.3	71.3	73.9	73.7	74.7
Concepts	72.1	74.0	74.4	73.3	76.3	76.8
Nouns	82.0	82.8	83.1	83.2	84.5	85.0
Verbs	40.6	45.2	45.9	45.5	50.8	50.8
Adjectives	49.5	57.4	57.4	45.4	56.5	59.1
Adverbs	0.0	0.0	0.0	56.7	50.0	86.7
Events	42.2	47.5	48.0	45.5	51.9	52.5
Perfect sense	68.3	71.3	71.8	75.4	76.9	78.0
F1 std dev	0.51	0.32	0.51	0.55	0.23	0.54
F1 conf interval	65.9	68.9	69.1	68.2	70.5	71.0
interval	67.4	69.8	70.5	69.8	71.1	72.5
# illformed	0.0	1.4	0.6	1.4	1.8	0.4
# perfect (avg)	53.2	73.6	73.0	68.8	75.2	79.2
# perfect (all 5)	30	50	50	43	49	53
# zero (avg)	17.8	17.6	15.6	17.6	13.8	15.2
# zero (all 5)	8	12	10	7	8	9
# same (all 5)	254	307	331	276	310	330

Table B.2: Detailed Counter scores for our models on the **German** dev and test sets of release 2.2.0. All scores are averages of 5 runs.

	PMB release 2.2.0 - Italian					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	76.1	78.2	78.3	77.3	79.6	80.7
Rec	68.0	70.0	70.1	69.7	73.2	73.8
F1	71.9	73.8	74.0	73.3	76.3	77.1
Operators	81.3	86.9	84.3	79.9	86.3	86.9
Roles	76.2	77.3	78.3	77.6	79.6	80.5
Concepts	74.2	75.9	76.1	76.0	77.2	78.6
Nouns	82.6	82.9	82.9	84.6	84.2	85.9
Verbs	45.8	53.4	55.0	47.4	53.8	56.1
Adjectives	57.4	55.2	57.5	58.8	61.6	60.4
Adverbs	0.0	0.0	0.0	0.0	0.0	0.0
Events	48.4	53.8	55.6	50.2	55.7	57.1
Perfect sense	73.4	75.5	75.7	75.1	78.1	79.1
F1 std dev	0.49	0.34	0.16	0.49	0.29	0.38
F1 confidence	71.2	73.4	73.8	72.6	75.9	76.6
interval	72.5	74.3	74.2	74.0	76.7	77.7
# illformed	0.2	0.4	0.0	0.0	0.4	0.0
# perfect (avg)	38.6	49.4	47.4	53.6	61.6	69.4
# perfect (all 5)	19	29	30	40	48	50
# zero (avg)	10.0	10.0	11.6	8.6	6.8	5.2
# zero (all 5)	6	5	6	5	5	3
# same (all 5)	139	163	172	185	193	212

Table B.3: Detailed Counter scores for our models on the **Italian** dev and test sets of release 2.2.0. All scores are averages of 5 runs.

	PMB release 2.2.0 - Dutch					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	70.5	70.6	71.3	69.2	70.8	71.1
Rec	62.0	62.5	62.9	58.9	60.5	61.0
F1	66.0	66.3	66.8	63.7	65.2	65.7
Operators	75.6	76.4	77.3	68.0	71.0	73.4
Roles	67.4	66.8	68.3	66.3	67.6	67.8
Concepts	71.6	72.3	71.9	72.9	74.0	73.6
Nouns	78.5	79.1	79.1	81.7	81.4	81.7
Verbs	45.2	47.0	43.1	42.2	50.0	45.8
Adjectives	62.7	64.7	69.5	51.2	48.0	57.4
Adverbs	0.0	0.0	80.0	0.0	40.0	80.0
Events	49.2	50.8	48.6	44.1	49.7	48.1
Perfect sense	67.9	68.1	68.9	65.3	67.0	67.5
F1 std dev	0.68	0.13	0.6	0.63	0.56	0.17
F1 confidence	65.0	66.1	66.0	62.8	64.4	65.5
interval	66.9	66.5	67.7	64.5	66.0	65.9
# illformed	0.0	1.0	0.0	0.2	0.0	0.0
# perfect (avg)	31.8	30.6	29.4	25.2	29.4	29.2
# perfect (all 5)	22	24	23	20	21	25
# zero (avg)	6.8	10.0	10.6	8.6	11.0	9.4
# zero (all 5)	5	7	7	7	9	7
# same (all 5)	143	161	160	140	167	145

Table B.4: Detailed Counter scores for our models on the **Dutch** dev and test sets of release 2.2.0. All scores are averages of 5 runs.

	PMB release 3.0.0 - English							
	Development set				Test set			
	bert	+ch (1e)	+ch (2e)	+ch +sem	bert	+ch (1e)	+ch (2e)	+ch +sem
Prec	88.8	88.9	89.3	89.5	90.0	90.6	90.3	90.5
Rec	86.4	87.3	86.9	87.2	87.1	87.9	87.6	88.0
F1	87.6	88.1	88.1	88.4	88.5	89.2	88.9	89.3
Operators	95.0	95.4	95.4	95.7	95.7	95.7	95.7	96.1
Roles	89.0	89.0	89.2	89.9	89.4	90.1	89.9	90.0
Concepts	84.7	84.9	85.6	85.4	87.3	87.9	87.4	87.7
Nouns	90.6	91.0	91.4	91.5	92.0	92.5	91.8	92.5
Verbs	69.1	68.9	70.4	69.2	75.3	76.0	76.4	75.3
Adjectives	76.1	75.3	76.6	75.5	75.8	77.5	76.2	76.0
Adverbs	78.1	77.7	78.7	80.1	88.0	88.2	87.7	88.9
Events	70.8	70.5	71.9	70.7	75.4	76.3	76.4	75.4
Perfect sense	89.6	90.3	90.2	90.4	91.6	92.2	92.0	92.1
Infreq. sense	54.9	57.6	56.5	56.0	62.0	62.8	62.7	63.1
F1 std dev	0.19	0.25	0.30	0.34	0.26	0.24	0.29	0.22
F1 confidence	87.3	87.8	87.7	87.9	88.2	88.9	88.5	89.0
interval	87.9	88.5	88.5	88.8	88.9	89.5	89.4	89.6
# illformed	3.2	0.8	2.8	2.0	4.6	3.0	2.8	2.0
# perfect (avg)	336.2	350.6	352.4	352.8	358.0	372.4	365.0	367.8
# perfect (all 5)	212	238	229	226	242	255	239	241
# zero (avg)	6.6	3.6	5.0	3.6	5.0	3.2	3.6	2.6
# zero (all 5)	2	2	1	1	0	0	0	0
# same (all 5)	347	387	386	365	364	378	361	361

Table B.5: Detailed Counter scores for our models on the **English** dev and test sets of release 3.0.0. All scores are averages of 5 runs.

	PMB release 3.0.0 - German					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	82.8	84.4	84.9	81.3	83.5	84.0
Rec	77.8	79.6	80.2	77.7	79.1	80.1
F1	80.2	81.9	82.4	79.5	81.3	82.0
Operators	90.2	90.8	91.6	88.4	89.6	90.6
Roles	84.4	84.7	85.1	81.2	82.5	82.7
Concepts	76.5	79.8	80.3	76.7	80.6	80.8
Nouns	86.4	88.4	88.3	84.8	87.0	87.6
Verbs	51.2	58.5	59.8	55.4	61.5	61.5
Adjectives	52.2	55.4	59.4	46.8	62.6	60.2
Adverbs	64.7	59.4	61.4	98.0	98.2	100.0
Events	51.5	57.6	59.7	53.8	61.7	61.2
Perfect sense	81.5	83.3	83.8	80.5	82.6	83.2
F1 std dev	0.37	0.4	0.55	0.2	0.28	0.25
F1 confidence	79.7	81.3	81.7	79.2	80.9	81.6
interval	80.7	82.5	83.21	79.7	81.7	82.3
# illformed	0.2	0.6	0.2	0.6	1.4	0.2
# perfect (avg)	80.2	89.6	101.8	61.8	75.8	82.4
# perfect (all 5)	54	62	72	35	47	58
# zero (avg)	5.2	4.4	4.2	1.0	3.0	1.4
# zero (all 5)	4	3	4	0	1	0
# same (all 5)	176	175	189	133	153	169

Table B.6: Detailed Counter scores for our models on the **German** dev and test sets of release 3.0.0. All scores are averages of 5 runs.

	PMB release 3.0.0 - Italian					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	79.7	81.9	82.4	80.7	82.4	82.4
Rec	75.3	76.7	77.8	77.2	78.7	78.7
F1	77.4	79.2	80.0	78.9	80.5	80.5
Operators	88.6	91.2	91.9	89.9	92.6	92.5
Roles	80.6	82.0	82.6	81.3	82.8	82.8
Concepts	72.9	75.5	76.0	73.9	75.2	75.2
Nouns	82.5	83.8	83.8	83.0	82.9	82.9
Verbs	44.4	51.0	54.7	43.2	49.5	50.9
Adjectives	54.1	53.5	53.5	63.2	59.7	59.5
Adverbs	66.7	66.7	66.0	82.7	79.9	83.1
Events	46.7	51.6	54.4	48.4	52.3	53.2
Perfect sense	79.1	80.8	81.7	80.5	82.1	82.3
F1 std dev	0.4	0.28	0.29	0.37	0.4	0.33
F1 confidence	76.9	78.8	79.6	78.4	79.4	80.0
interval	78.0	79.6	80.5	79.4	81.1	81.0
# illformed	2.0	2.8	2.4	0.6	1.6	1.2
# perfect (avg)	74.6	91.6	95.8	97.6	106.6	106.8
# perfect (all 5)	45	58	66	60	72	75
# zero (avg)	5.8	7.0	5.6	5.6	7.2	6.8
# zero (all 5)	3	3	3	4	5	5
# same (all 5)	165	193	212	203	225	250

Table B.7: Detailed Counter scores for our models on the **Italian** dev and test sets of release 3.0.0. All scores are averages of 5 runs.

	PMB release 3.0.0 - Dutch					
	Development set			Test set		
	bert	+ch (1e)	+ch (2e)	bert	+ch (1e)	+ch (2e)
Prec	75.1	75.3	74.8	74.7	75.5	74.5
Rec	68.1	68.5	68.6	68.1	68.5	68.2
F1	71.4	71.8	71.6	71.2	71.8	71.2
Operators	83.7	84.4	84.0	84.1	84.2	83.9
Roles	74.2	74.4	74.1	73.0	73.8	72.7
Concepts	69.9	69.7	69.1	69.9	70.9	69.7
Nouns	78.3	78.3	77.8	78.4	79.2	78.2
Verbs	41.1	40.2	40.2	42.3	42.7	40.4
Adjectives	55.1	52.1	52.2	51.1	53.5	54.1
Adverbs	81.1	78.8	77.3	74.2	74.1	86.4
Events	44.4	43.1	43.0	44.5	45.4	43.7
Perfect sense	73.3	73.7	73.8	73.3	73.8	73.2
F1 std dev	0.52	0.27	0.26	0.39	0.3	0.18
F1 confidence	70.7	71.4	71.2	70.7	71.4	71.0
interval	72.1	72.1	72.0	71.8	72.2	71.4
# illformed	5.4	4.2	6.2	2.4	2.6	4.0
# perfect (avg)	55.0	52.4	58.4	47.4	49.8	47.6
# perfect (all 5)	38	33	39	29	35	35
# zero (avg)	11.8	13.2	15.0	9.4	6.0	9.6
# zero (all 5)	4	7	5	3	2	3
# same (all 5)	135	154	153	145	131	158

Table B.8: Detailed Counter scores for our models on the **Dutch** dev and test sets of release 3.0.0. All scores are averages of 5 runs.

	PMB release 3.0.0 - English									
	Development set					Test set				
	#	bert	+ch (1e)	+ch (2e)	+ch +sem	#	bert	+ch (1e)	+ch (2e)	+ch +sem
PRESUPPOSITION	977	96.1	96.6	96.0	96.8	1,074	96.2	96.5	96.0	97.0
EQU	460	93.3	93.4	93.8	93.7	454	92.7	93.5	93.6	93.7
TPR	414	97.7	97.9	97.5	98.0	468	97.1	97.1	97.4	97.4
NEGATION	73	95.2	95.6	94.7	96.3	80	99.0	98.0	97.4	97.7
CONSEQUENCE	33	88.3	92.1	89.9	87.6	33	81.5	86.1	85.8	84.6
CONDITION	33	83.8	84.9	86.4	86.5	33	80.0	86.2	83.6	88.0
ATTRIBUTION	27	93.8	95.6	95.6	94.4	18	89.7	89.4	91.2	92.3
POSSIBILITY	24	79.0	83.2	83.2	85.7	18	95.5	95.5	95.5	96.6
TAB	17	95.1	97.0	97.6	98.2	16	100	92.5	98.8	98.8
TIN	16	92.1	94.4	92.1	92.6	4	72.7	100	97.5	92.7
CONTINUATION	14	89.7	91.1	92.4	90.1	15	95.9	93.9	94.5	95.8
NECESSITY	9	86.4	87.6	90.9	78.6	4	82.0	84.2	94.7	92.3
APX	5	100	86.8	90.2	90.6	9	94.1	94.1	94.1	94.1
LES	3	100	100	93.7	100	5	78.4	73.1	76.0	80.0
CONTRAST	2	80.0	86.9	86.9	80.0	5	87.3	92.3	94.1	94.1

Table B.9: F-scores per individual **operator** on the English development and test sets of release 3.0.0. All scores are averages of 5 runs. Operators are included if they occur at least 5 times in either the development or test set.

	PMB release 3.0.0 - English									
	Development set					Test set				
	#	bert	+ch (1e)	+ch (2e)	+ch +sem	#	bert	+ch (1e)	+ch (2e)	+ch +sem
Time	870	96.8	97.3	96.9	96.7	916	96.8	97.0	96.9	97.2
Theme	463	83.0	84.2	85.1	85.2	539	86.2	87.1	86.9	86.6
Agent	373	92.2	92.7	91.6	92.4	399	95.1	95.1	94.8	95.4
Name	365	88.3	88.2	88.4	89.0	393	83.3	84.1	84.0	84.0
Attribute	193	86.5	88.0	86.6	87.4	134	86.2	87.0	86.7	84.6
Experiencer	106	83.4	85.7	84.5	83.8	128	92.3	94.4	93.7	93.8
Role	99	92.9	92.5	93.0	94.9	91	92.7	94.3	92.9	94.3
Patient	97	79.4	82.4	85.0	83.8	121	81.2	84.1	84.1	82.6
Stimulus	87	86.1	86.6	85.6	88.2	99	89.9	92.2	90.7	90.5
Location	83	81.5	80.8	78.0	79.3	109	81.6	79.3	79.1	80.4
Quantity	81	92.8	92.0	92.7	92.8	113	85.0	84.5	86.8	86.9
Co-Theme	77	82.1	85.2	82.4	81.7	89	87.4	88.7	87.6	84.8
Of	44	83.3	85.1	84.8	88.0	36	79.2	81.8	81.2	81.4
Pivot	43	91.2	94.2	90.8	95.2	33	80.1	77.9	79.6	78.5
Source	42	78.7	77.7	80.0	76.7	55	79.5	81.5	79.7	80.8
Destination	42	77.8	76.7	81.4	80.0	51	74.4	74.7	75.5	77.1
User	37	76.5	74.1	75.5	79.1	53	78.2	78.1	80.0	79.2
Part	36	79.7	79.0	80.6	78.9	51	85.0	83.5	85.4	86.9
Value	35	72.6	75.2	70.4	79.0	48	79.5	80.3	78.8	83.3
Sub	31	93.8	91.1	92.6	94.6	18	80.4	79.1	70.0	71.7
Degree	30	88.0	89.7	84.8	87.3	20	94.5	96.4	93.6	95.4
Unit	27	86.5	89.8	88.7	86.4	48	90.1	91.1	91.0	91.9
Result	22	72.0	79.7	75.0	78.7	22	69.0	75.4	75.1	71.9
Recipient	22	89.8	86.5	87.2	89.1	16	72.1	74.1	76.7	75.3
Topic	18	65.9	61.7	65.5	67.4	13	57.6	63.6	63.0	64.3
Creator	17	77.5	73.2	62.9	75.0	14	68.8	65.2	59.9	71.7
Beneficiary	15	65.6	62.6	55.8	61.7	19	66.7	66.0	68.0	66.3
Manner	15	64.1	71.3	63.6	76.7	14	54.9	66.7	76.3	81.8
Owner	14	63.2	60.5	58.8	71.2	18	62.9	61.8	63.5	61.1
ClockTime	13	77.4	59.2	81.3	78.4	22	53.4	61.0	57.1	64.2
YearOfCentury	12	62.1	76.4	78.5	74.8	18	56.6	76.2	75.0	77.8

Table B.10: F-scores per individual **role** on the English development and test sets of release 3.0.0. All scores are averages of 5 runs. Roles are included if they occur at least 15 times in either the development or test set.

	PMB release 3.0.0 - English									
	Development set					Test set				
	#	bert	+ch (1e)	+ch (2e)	+ch +sem	#	bert	+ch (1e)	+ch (2e)	+ch +sem
time.n.08	846	97.3	97.7	97.3	97.4	884	97.4	97.6	97.5	97.8
male.n.02	303	96.3	96.8	96.5	97.3	294	97.1	97.7	97.1	97.0
person.n.01	159	93.8	92.9	93.4	94.2	150	92.5	93.3	93.1	96.1
female.n.02	112	95.9	96.6	95.8	96.8	145	94.7	95.4	93.9	93.6
entity.n.01	98	92.8	91.1	93.0	91.8	71	85.5	85.4	86.7	85.9
measure.n.02	42	93.7	96.8	95.5	94.4	70	96.8	97.3	98.2	98.7
be.v.03	32	86.3	84.5	86.0	87.5	39	93.9	95.9	91.8	93.2
country.n.02	32	83.1	82.0	80.1	83.4	38	88.3	86.7	85.0	89.0
be.v.02	30	81.7	84.1	83.9	83.4	27	86.2	86.8	86.7	87.7
proposition.n.01	22	93.4	94.7	95.5	94.9	16	88.3	88.0	92.6	92.6
be.v.01	20	63.9	65.9	59.5	66.7	26	69.8	71.4	69.6	67.4
very.r.01	16	90.4	93.9	90.4	91.6	14	97.1	97.8	97.1	96.4
city.n.01	15	66.3	68.7	78.4	73.2	30	83.7	88.9	88.6	87.6
location.n.01	15	91.0	82.3	85.9	86.8	17	84.0	80.9	82.4	74.9
dog.n.01	13	100	100	93.8	99.2	15	90.5	96.0	92.6	89.9
money.n.01	11	99.1	97.2	99.1	100	9	93.6	90.5	92.6	92.6
desire.v.01	10	78.8	90.9	89.9	88.9	6	96.5	100	100	100
buy.v.01	9	100	100	100	100	13	100	100	100	99.2
company.n.01	9	71.0	61.7	68.6	77.8	8	51.3	84.5	72.7	79.5
travel.v.01	8	88.9	86.7	88.9	92.0	9	90.9	94.1	86.4	93.2
have.v.02	8	66.7	60.3	55.7	61.8	8	52.5	32.0	44.1	49.2
die.v.01	7	91.4	100	94.3	95.7	8	94.1	92.8	94.1	93.0
name.n.01	6	66.7	57.2	55.3	68.0	18	83.5	82.0	82.7	80.2
populate.v.01	6	85.7	90.6	92.3	92.3	12	100	99.1	100	100
man.n.01	6	85.7	84.5	85.7	85.7	9	100	100	100	100
put.v.01	6	91.5	90.3	96.8	89.7	6	93.7	92.3	88.2	100

Table B.11: F-scores per individual **concept** on the English development and test sets of release 3.0.0. All scores are averages of 5 runs. Concepts are included if they occur at least 6 times in both the development and test set.

C Sentence Analysis

Table C.1 shows the sentences for which our best model (on 3.0.0 English dev) produced the lowest quality DRs, with a possible explanation. In Table C.2, we show the sentences for which our best model has the best performance (relative to the `Bert-only` baseline model). It is harder to give an explanation in this case, though we indicate which clauses were (in)correctly predicted by the models.

Document	F1	Comment
Look out!	0.00	Imperative
The dove symbolizes peace.	0.13	Condition + consequence
HBV Union Criticizes Deutsche Bank	0.25	Two multi-word expressions
You can buy stamps at any post office.	0.32	Possibility and quantifier
Fire burns.	0.33	Generic, short
How's Lanzarote?	0.36	How-question
I'd better drive you home.	0.37	Necessity, infreq. sense of drive
What a lot of books!	0.38	Multi-sentence
Do they belong to the university library?		
Maybe he is Italian or Spanish.	0.40	Possibility and conjunction
I always get up at 6 o'clock in the morning.	0.40	Necessity + clocktime

Table C.1: Sentences of the English 3.0.0 dev set for which our best model (+char +sem) produced the **worst** DRs.

Document	Diff	Comment
Fish surface for air.	0.554	Correctly produced <code>Goal</code>
Oil this bicycle.	0.482	Correctly produced <code>oil</code> as a verb
I'm fed up with this winter, I want spring right now!	0.404	Correctly produced <code>CONTINUATION</code> and <code>Pivot</code>
He's Argentinian.	0.386	<code>Bert-only</code> failed to produce <code>country</code> and <code>Name</code>
Alas!	0.364	Odd sentence, but correctly produced <code>state.v.01</code>
Fire burns.	0.300	Bad performance for both, <code>Bert-only</code> got 0.0
All journeys begin with a first step.	0.300	<code>Bert-only</code> produced many non-matching clauses
How heavy you are!	0.299	<code>Bert-only</code> produced many non-matching clauses
One plus two is equal to three.	0.252	Correctly produced <code>summation.n.04</code>
He's not like us.	0.246	Correctly produced <code>Theme</code> and <code>Co-Theme</code>

Table C.2: Sentences of the English 3.0.0 dev set for which our best model (+char +sem) produced the **best** DRs, relative to the `Bert-only` baseline.

Bibliographical Abbreviations

- ACL → Annual Meeting of the Association for Computational Linguistics
- COLING → International Conference on Computational Linguistics
- CoNLL → Conference on Computational Natural Language Learning
- EACL → Conference of the European Chapter of the Association for Computational Linguistics
- EMNLP → Conference on Empirical Methods in Natural Language Processing
- ICLR → International Conference on Learning Representations
- ICML → International Conference on Machine learning
- IJCNLP → International Joint Conference on Natural Language Processing
- IWCS → International Conference on Computational Semantics
- LREC → Language Resources and Evaluation Conference
- NAACL → Conference of the North American Chapter of the Association for Computational Linguistics
- NIPS → Neural Information Processing Systems Conference
- SemEval → International Workshop on Semantic Evaluation
- TACL → Transactions of the Association for Computational Linguistics

Bibliography

- Abend, O. and Rappoport, A. (2013). Universal conceptual cognitive annotation (UCCA). In *ACL (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria.
- Abzianidze, L., Bjerva, J., Evang, K., Haagsma, H., van Noord, R., Ludmann, P., Nguyen, D.-D., and Bos, J. (2017). The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *EACL: Volume 2, Short Papers*, pages 242–247, Valencia, Spain.
- Abzianidze, L. and Bos, J. (2017). Towards universal semantic tagging. In *IWCS - Short Papers*, pages 307–313, Montpellier, France.
- Abzianidze, L., van Noord, R., Haagsma, H., and Bos, J. (2019). The first shared task on discourse representation structure parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden.
- Aharoni, R. and Goldberg, Y. (2017). Towards string-to-tree neural machine translation. In *ACL (Volume 2: Short Papers)*, pages 132–140.
- Allen, J., Dzikovska, M., Manshadi, M., and Swift, M. (2007). Deep linguistic processing for spoken dialogue systems. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 49–56, Prague, Czech Republic.
- Allen, J. F., Swift, M., and de Beaumont, W. (2008). Deep Semantic Analysis of Text. In Bos, J. and Delmonte, R., editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.
- Alvarez-Melis, D. and Jaakkola, T. S. (2017). Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France.
- Anchiêta, R. and Pardo, T. (2018). Towards AMR-BR: A SemBank for Brazilian Portuguese language. In *LREC*, Miyazaki, Japan. ELRA.

- Andreas, J., Vlachos, A., and Clark, S. (2013). Semantic parsing as machine translation. In *ACL (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria.
- Artzi, Y., Lee, K., and Zettlemoyer, L. (2015). Broad-coverage CCG semantic parsing with AMR. In *EMNLP*, pages 1699–1710, Lisbon, Portugal.
- Asher, N. (1993). *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.
- Asher, N. and Lascarides, A. (2003). *Logics of Conversation*. Studies in natural language processing. Cambridge University Press.
- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.
- Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *ACL-COLING, Volume 1*, pages 86–90, Montreal, Quebec, Canada.
- Ballesteros, M. and Al-Onaizan, Y. (2017). AMR parsing using stack-LSTMs. In *EMNLP*, pages 1269–1275, Copenhagen, Denmark.
- Ballesteros, M., Dyer, C., and Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with LSTMs. In *EMNLP*, pages 349–359, Lisbon, Portugal.
- Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.
- Barzdins, G. and Gosko, D. (2016). RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *SemEval*, pages 1143–1147, San Diego, California.
- Basile, V., Bos, J., Evang, K., and Venhuizen, N. (2012a). Developing a large semantically annotated corpus. In *LREC*, pages 3196–3200, Istanbul, Turkey.
- Basile, V., Bos, J., Evang, K., and Venhuizen, N. (2012b). A platform for collaborative semantic annotation. In *EACL (Demonstrations)*, pages 92–96, Avignon, France.
- Bateman, J. A. (1990). Upper modeling: organizing knowledge for natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Generation*.

- Beaver, D. I. (2002). Presupposition projection in DRT: A critical assesment. In Beaver, D., Casillas, L., Clark, B., and Kaufmann, S., editors, *The Construction of Meaning*, pages 23–43. Stanford University.
- Bender, E. M. and Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. In *ACL*, pages 5185–5198, Online.
- Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.
- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.
- Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *ACL*, volume 1, pages 1415–1425, Baltimore, Maryland, USA.
- Bjerva, J., Bos, J., and Haagsma, H. (2016a). The Meaning Factory at SemEval-2016 task 8: Producing AMRs with Boxer. In *SemEval*, pages 1179–1184, San Diego, California.
- Bjerva, J., Plank, B., and Bos, J. (2016b). Semantic tagging with deep residual networks. In *COLING*, pages 3531–3541, Osaka, Japan.
- Blackburn, P. and Bos, J. (2005). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI.
- Blackburn, P., Bos, J., Kohlhase, M., and De Nivelle, H. (1998). Inference and computational semantics. In *IWCS*. Citeseer.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *TACL*, 5:135–146.
- Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark.
- Bonial, C., Corvey, W. J., Palmer, M., Petukhova, V., and Bunt, H. (2011). A hierarchical unification of LIRICS and VerbNet semantic roles. In *Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC 2011)*, pages 483–489, Palo Alto, CA, USA.

- Bos, J. (2001). Doris 2001: Underspecification, resolution and inference for discourse representation structures. In Blackburn, P. and Kohlhase, M., editors, *ICoS-3 Inference in Computational Semantics. Workshop Proceedings*, pages 117–124.
- Bos, J. (2005). Towards wide-coverage semantic interpretation. In *IWCS*, volume 6, pages 42–53.
- Bos, J. (2008a). Let’s not argue about semantics. In *LREC*, pages 2835–2840, Marrakech, Morocco.
- Bos, J. (2008b). Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications, Venice, Italy.
- Bos, J. (2015). Open-domain semantic parsing with Boxer. In Megyesi, B., editor, *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 301–304, Vilnius, Lithuania.
- Bos, J. (2016). Expressive power of Abstract Meaning Representations. *Computational Linguistics*, 42(3):527–535.
- Bos, J., Basile, V., Evang, K., Venhuizen, N., and Bjerva, J. (2017). The Groningen Meaning Bank. In Ide, N. and Pustejovsky, J., editors, *Handbook of Linguistic Annotation*. Springer Netherlands.
- Bos, J., Clark, S., Steedman, M., Curran, J. R., and Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *COLING*, pages 1240–1246, Geneva, Switzerland.
- Bos, J., Evang, K., and Nissim, M. (2012). Annotating semantic roles in a lexicalised grammar environment. In *Proceedings of the Eighth Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-8)*, pages 9–12, Pisa, Italy.
- Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2013). Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer.
- Brants, T. (2000). Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, ANLC’00*, pages 224–231, Stroudsburg, PA, USA.
- Britz, D., Goldie, A., Luong, M.-T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. In *EMNLP*, pages 1442–1451, Copenhagen, Denmark.

- Butler, A. (2010). *The Semantics of Grammatical Dependencies*, volume 23. Emerald Group Publishing Limited.
- Buys, J. and Blunsom, P. (2017). Robust incremental neural semantic graph parsing. In *ACL*, volume 1, pages 1215–1226, Vancouver, Canada.
- Cai, D. and Lam, W. (2019). Core semantic first: A top-down approach for AMR parsing. In *EMNLP-IJCNLP*, pages 3799–3809, Hong Kong, China.
- Cai, D. and Lam, W. (2020). AMR parsing via graph-sequence iterative inference. In *ACL*, pages 1290–1301, Online.
- Cai, S. and Knight, K. (2013). Smatch: An evaluation metric for semantic feature structures. In *ACL (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria.
- Calixto, I., Liu, Q., and Campbell, N. (2017). Doubly-attentive decoder for multi-modal neural machine translation. In *ACL (Volume 1: Long Papers)*, pages 1913–1924, Vancouver, Canada.
- Cavnar, W. (1995). Using an n-gram-based document representation with a vector processing retrieval model. *NIST special publication*, pages 269–277.
- Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (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.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, Doha, Qatar.
- Choe, H., Han, J., Park, H., and Kim, H. (2019). Copula and case-stacking annotations for Korean AMR. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 128–135, Florence, Italy.
- Chollet, F. et al. (2015). Keras. <https://keras.io>.
- Chung, J., Cho, K., and Bengio, Y. (2016a). A character-level decoder without explicit segmentation for neural machine translation. In *ACL (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany.
- Chung, J., Cho, K., and Bengio, Y. (2016b). NYU-MILA neural machine translation systems for WMT16. In *Proceedings of the First Conference on Machine Translation*, pages 268–271, Berlin, Germany.

- Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.
- Church, K. W. (1993). Char_align: a program for aligning parallel texts at the character level. In *ACL*, pages 1–8.
- Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, April 26-30, 2020*.
- Clark, S., Curran, J. R., and Osborne, M. (2003). Bootstrapping POS taggers using unlabelled data. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 49–55.
- Clarke, J., Goldwasser, D., Chang, M.-W., and Roth, D. (2010). Driving semantic parsing from the world’s response. In *CoNLL*, pages 18–27, Uppsala, Sweden.
- Collins, M., Koehn, P., and Kučerová, I. (2005). Clause restructuring for statistical machine translation. In *ACL*, pages 531–540.
- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451, Online.
- Copestake, A. (2007). Semantic composition with (robust) minimal recursion semantics. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 73–80, Prague, Czech Republic.
- Copestake, A., Flickinger, D., Sag, I., and Pollard, C. (2005). Minimal Recursion Semantics: An introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332.
- Costa-jussà, M. R. and Fonollosa, J. A. R. (2016). Character-based neural machine translation. In *ACL (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany.
- Cucerzan, S. (2007). Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP-CoNLL*, pages 708–716, Prague, Czech Republic.
- Cucerzan, S. and Yarowsky, D. (1999). Language independent named entity recognition combining morphological and contextual evidence. In *1999 Joint SIG-DAT Conference on Empirical Methods on Natural Language Processing and Very Large Corpora*.

- Curran, J., Clark, S., and Bos, J. (2007). Linguistically motivated large-scale NLP with C&C and Boxer. In *ACL Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic.
- Currey, A. and Heafield, K. (2018). Multi-source syntactic neural machine translation. In *EMNLP*, pages 2961–2966.
- Dagan, I., Glickman, O., and Magnini, B. (2005). The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.
- Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.
- Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.
- Damashek, M. (1995). Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848.
- Damerau, F. J. (1981). Operating statistics for the transformational question answering system. *American Journal of Computational Linguistics*, 7(1):30–42.
- Damonte, M., Cohen, S. B., and Satta, G. (2017). An incremental parser for abstract meaning representation. In *EACL: Volume 1, Long Papers*, pages 536–546, Valencia, Spain.
- de Bruijn, N. G. (1972). Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. In *Indagationes Mathematicae (Proceedings)*, volume 75, pages 381–392. Elsevier.
- de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., and Nissim, M. (2019). BERTje: A Dutch BERT model. *arXiv:1912.09582 [cs]*.
- Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.
- Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

- Denkowski, M. and Neubig, G. (2017). Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27, Vancouver.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.
- Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *ACL (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany.
- Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *ACL (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia.
- Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for on-line learning and stochastic optimization. *Journal of machine learning research*, 12(7).
- Dunning, T. (1994). *Statistical identification of language*. Computing Research Laboratory, New Mexico State University Las Cruces, NM, USA.
- Duong, L., Afshar, H., Estival, D., Pink, G., Cohen, P., and Johnson, M. (2017). Multilingual semantic parsing and code-switching. In *CoNLL*, pages 379–389, Vancouver, Canada.
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179 – 211.
- Ettinger, A. (2020). What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *TACL*, 8:34–48.
- Evang, K. (2016). *Cross-lingual Semantic Parsing with Categorical Grammars*. PhD thesis, University of Groningen.
- Evang, K. (2019). Transition-based DRS parsing using stack-LSTMs. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden.
- Evang, K., Basile, V., Chrupała, G., and Bos, J. (2013). Elephant: Sequence labeling for word and sentence segmentation. In *EMNLP*, pages 1422–1426, Seattle, Washington, USA.
- Evang, K. and Bos, J. (2016). Cross-lingual learning of an open-domain semantic parser. In *COLING*, pages 579–588, Osaka, Japan.
- Fancellu, F., Gilroy, S., Lopez, A., and Lapata, M. (2019). Semantic graph parsing with recurrent neural network DAG grammars. In *EMNLP-IJCNLP*, pages 2769–2778, Hong Kong, China.

- Fellbaum, C., editor (1998). *WordNet. An Electronic Lexical Database*. The MIT Press, Cambridge, Ma., USA.
- Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D. (2018). Improving text-to-SQL evaluation methodology. In *ACL (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia.
- Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*, pages 866–875, San Diego, California.
- Flanigan, J., Dyer, C., Smith, N. A., and Carbonell, J. (2016). CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. *Proceedings of SemEval*, pages 1202–1206.
- Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., and Smith, N. A. (2014). A discriminative graph-based parser for the abstract meaning representation. In *ACL (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland.
- Foland, W. and Martin, J. H. (2017). Abstract meaning representation parsing using LSTM recurrent neural networks. In *ACL (Volume 1: Long Papers)*, pages 463–472, Vancouver, Canada.
- Fu, Q., Zhang, Y., Liu, J., and Zhang, M. (2020). DRTS parsing with structure-aware encoding and decoding. In *ACL*, pages 6818–6828, Online.
- Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, 12(2):23–38.
- Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.
- Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. S. (2017). AllenNLP: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.
- Ge, R. and Mooney, R. (2005). A statistical semantic parser that integrates syntax and semantics. In *CoNLL*, pages 9–16, Ann Arbor, Michigan.
- Ge, R. and Mooney, R. (2009). Learning a compositional semantic parser using an existing syntactic parser. In *ACL-AFNLP*, pages 611–619, Suntec, Singapore.
- Geach, P. T. (1962). *Reference and Generality: An Examination of Some Medieval and Modern Theories*. Cornell University Press.

- Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE.
- Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2.
- Geurts, B. (1999). *Presuppositions and Pronouns*, volume 3 of *Current Research in the Semantics/Pragmatics interface*. Elsevier.
- Gildea, D. and Jurafsky, D. (2000). Automatic labeling of semantic roles. In *ACL*, pages 512–520, Hong Kong.
- Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.
- Goldwasser, D., Reichart, R., Clarke, J., and Roth, D. (2011). Confidence driven unsupervised semantic parsing. In *ACL: Human Language Technologies*, pages 1486–1495, Portland, Oregon, USA.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Goodman, J., Vlachos, A., and Naradowsky, J. (2016). Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *ACL (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany.
- Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *LREC*.
- Graves, A. (2008). Supervised sequence labelling with recurrent neural networks [ph. d. dissertation]. *Technical University of Munich, Germany*.
- Graves, A. (2012). Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.
- Groschwitz, J., Fowlie, M., Johnson, M., and Koller, A. (2017). A constrained graph algebra for semantic parsing with AMRs. In *IWCS - Long papers*.

- Groschwitz, J., Lindemann, M., Fowlie, M., Johnson, M., and Koller, A. (2018). AMR dependency parsing with a typed semantic algebra. In *ACL (Volume 1: Long Papers)*, pages 1831–1841, Melbourne, Australia.
- Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.
- Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., and Zhang, Z. (2019). Star-transformer. In *NAACL, Volume 1 (Long and Short Papers)*, pages 1315–1325, Minneapolis, Minnesota.
- Guo, Z. and Lu, W. (2018). Better transition-based AMR parsing with a refined search space. In *EMNLP*, pages 1712–1722, Brussels, Belgium.
- Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Bojar, O., Cinková, S., Fučíková, E., Mikulová, M., Pajas, P., Popelka, J., Semecký, J., Šindlerová, J., Štěpánek, J., Toman, J., Urešová, Z., and Žabokrtský, Z. (2012). Announcing Prague Czech-English dependency treebank 2.0. In *LREC*, pages 3153–3160, Istanbul, Turkey. ELRA.
- Hamp, B. and Feldweg, H. (1997). GermaNet - a lexical-semantic net for German. In *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, Madrid, Spain.
- Hardy, H. and Vlachos, A. (2018). Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *EMNLP*, pages 768–773, Brussels, Belgium.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.
- He, X. and Golub, D. (2016). Character-level question answering with attention. In *EMNLP*, pages 1598–1607.
- Heinzerling, B. and Strube, M. (2018). BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *LREC*, Paris, France. ELRA.
- Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.
- Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., and Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, 3(2):105–147.

- Hershcovich, D., Abend, O., and Rappoport, A. (2017). A transition-based directed acyclic graph parser for UCCA. In *ACL (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada.
- Hershcovich, D., Abend, O., and Rappoport, A. (2018). Multitask parsing across semantic representations. In *ACL (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.
- Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). OntoNotes: The 90% solution. In *NAACL: Short Papers*, pages 57–60, New York City, USA.
- Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *ACL (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia.
- Issa, F., Damonte, M., Cohen, S. B., Yan, X., and Chang, Y. (2018). Abstract Meaning Representation for paraphrase detection. In *NAACL, Volume 1 (Long Papers)*, pages 442–452, New Orleans, Louisiana.
- Ivanova, A., Oepen, S., Øvrelid, L., and Flickinger, D. (2012). Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea.
- Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *ACL (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada.
- Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. In *ACL (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany.
- Johnson, M. and Klein, E. (1986). Discourse, anaphora and parsing. In *COLING*, pages 669–675, Bonn, Germany.
- Jones, B., Johnson, M., and Goldwater, S. (2012). Semantic parsing with Bayesian tree transducers. In *ACL (Volume 1: Long Papers)*, pages 488–496, Jeju Island, Korea.

- Jordan, M. I. (1986). Serial order: A parallel distributed processing approach, ics report 8604. *Institute for Cognitive Science, UCSD, La Jolla*.
- Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350.
- Junczys-Dowmunt, M., Dwojak, T., and Sennrich, R. (2016). The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany.
- Junczys-Dowmunt, M. and Grundkiewicz, R. (2017). An exploration of neural sequence-to-sequence architectures for automatic post-editing. In *IJCNLP (Volume 1: Long Papers)*, pages 120–129, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Hermann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *ACL (System Demonstrations)*, pages 116–121, Melbourne, Australia.
- Kadmon, N. (2001). *Formal Pragmatics*. Blackwell.
- Kamp, H. (1984). A theory of truth and semantic representation. In Groenendijk, J., Janssen, T. M., and Stokhof, M., editors, *Truth, Interpretation and Information*, pages 1–41. FORIS, Dordrecht – Holland/Cinnaminson – U.S.A.
- Kamp, H. and Reyle, U. (1993). *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.
- Kanerva, J., Luotolahti, J., and Ginter, F. (2015). Turku: Semantic dependency parsing as a sequence classification. In *SemEval*, pages 965–969, Denver, Colorado.
- Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Comput. Linguist.*, 20(3):331–378.
- Kasper, R. T. (1989). A flexible interface for linking applications to Penman’s sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.
- Kate, R. J. and Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *ACL*, pages 913–920, Sydney, Australia.

- Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. (2003). Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 180–183.
- Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *ACL (System Demonstrations)*, pages 67–72, Vancouver, Canada.
- Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *ACL Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.
- Konstas, I., Iyer, S., Yatskar, M., Choi, Y., and Zettlemoyer, L. (2017). Neural AMR: Sequence-to-sequence models for parsing and generation. In *ACL (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada.
- Krishnamurthy, J., Dasigi, P., and Gardner, M. (2017). Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*, pages 1516–1526, Copenhagen, Denmark.
- Kuijper, M., van Lenthe, M., and van Noord, R. (2018). UG18 at SemEval-2018 task 1: Generating additional training data for predicting emotion intensity in Spanish. In *SemEval*, pages 279–285, New Orleans, Louisiana.
- Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML ’01*, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

- Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *ACL-COLING, Volume 1*, pages 704–710, Montreal, Quebec, Canada.
- Le, P. and Zuidema, W. (2012). Learning compositional semantics for open domain semantic parsing. In *COLING*, pages 1535–1552, Mumbai, India.
- LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404.
- Lewis, M. and Steedman, M. (2013). Combined distributional and logical semantics. *TACL – Volume 1*, pages 179–192.
- Lewis, M. and Steedman, M. (2014). A* CCG parsing with a supertag-factored model. In *EMNLP*, pages 990–1000, Doha, Qatar.
- Li, B., Wen, Y., Qu, W., Bu, L., and Xue, N. (2016). Annotating the little prince with Chinese AMRs. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 7–15, Berlin, Germany.
- Liao, K., Lebanoff, L., and Liu, F. (2018). Abstract Meaning Representation for multi-document summarization. In *COLING*, pages 1178–1190, Santa Fe, New Mexico, USA.
- Libovický, J. and Helcl, J. (2017). Attention strategies for multi-source sequence-to-sequence learning. In *ACL (Volume 2: Short Papers)*, pages 196–202, Vancouver, Canada.
- Lindemann, M., Groschwitz, J., and Koller, A. (2019). Compositional semantic parsing across graphbanks. In *ACL*, pages 4576–4585, Florence, Italy.
- Ling, W., Blunsom, P., Grefenstette, E., Hermann, K. M., Kočiský, T., Wang, F., and Senior, A. (2016). Latent predictor networks for code generation. In *ACL (Volume 1: Long Papers)*, volume 1, pages 599–609, Berlin, Germany.
- Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fernandez, R., Amir, S., Marujo, L., and Luís, T. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, pages 1520–1530, Lisbon, Portugal.
- Linh, H. and Nguyen, H. (2019). A case study on meaning representation for Vietnamese. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 148–153, Florence, Italy.

- Liu, J., Cohen, S. B., and Lapata, M. (2018a). Discourse representation structure parsing. In *ACL (Volume 1: Long Papers)*, pages 429–439, Melbourne, Australia.
- Liu, J., Cohen, S. B., and Lapata, M. (2019a). Discourse representation parsing for sentences and documents. In *ACL*, pages 6248–6262, Florence, Italy.
- Liu, J., Cohen, S. B., and Lapata, M. (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.
- Liu, J., Cohen, S. B., and Lapata, M. (2020). Dscorer: A fast evaluation metric for discourse representation structure parsing. In *ACL*, pages 4547–4554, Online.
- Liu, Y., Che, W., Zheng, B., Qin, B., and Liu, T. (2018b). An AMR aligner tuned by transition-based parser. In *EMNLP*, pages 2422–2430, Brussels, Belgium.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019c). Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.
- Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany.
- Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421, Lisbon, Portugal.
- Lyu, C. and Titov, I. (2018). AMR parsing as graph prediction with latent alignment. In *ACL (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia.
- Ma, J. and Yarats, D. (2019). Quasi-hyperbolic momentum and Adam for deep learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.
- Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd ACL: System Demonstrations*, pages 55–60, Baltimore, Maryland.
- Martins, A. F. T. and Almeida, M. S. C. (2014). Priberam: A turbo semantic parser with second order features. In *SemEval*, pages 471–476, Dublin, Ireland.

- Masterman, M. (1961). Semantic message detection for machine translation, using an interlingua. In *Proc. 1961 International Conf. on Machine Translation*, pages 438–475.
- May, J. (2016). SemEval-2016 task 8: Meaning representation parsing. In *SemEval*, pages 1063–1073, San Diego, California.
- McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, pages 3428–3448, Florence, Italy.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and EMNLP*, pages 523–530, Vancouver, British Columbia, Canada.
- Migueles-Abraira, N., Agerri, R., and Diaz de Ilarraza, A. (2018). Annotating Abstract Meaning Representations for Spanish. In *LREC*, Miyazaki, Japan. ELRA.
- Mihaylov, T. and Frank, A. (2019). Discourse-aware semantic self-attention for narrative reading comprehension. In *EMNLP-IJCNLP*, pages 2541–2552, Hong Kong, China.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Miller, S., Bobrow, R., Ingria, R., and Schwartz, R. (1994). Hidden understanding models of natural language. In *ACL*, pages 25–32, Las Cruces, New Mexico, USA.
- Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.
- Misra, D. K. and Artzi, Y. (2016). Neural shift-reduce CCG semantic parsing. In *EMNLP*, pages 1775–1786, Austin, Texas.
- Muskens, R. (1996). Combining montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186.
- Nakov, P. and Tiedemann, J. (2012). Combining word-level and character-level models for machine translation between closely-related languages. In *ACL: Short Papers-Volume 2*, pages 301–305.

- Naseem, T., Shah, A., Wan, H., Florian, R., Roukos, S., and Ballesteros, M. (2019). Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *ACL*, pages 4586–4592, Florence, Italy.
- Nissim, M., van Noord, R., and van der Goot, R. (2020). Fair is better than sensational: Man is to doctor as woman is to doctor. *Computational Linguistics*, 46(2):487–497.
- Niven, T. and Kao, H.-Y. (2019). Probing neural network comprehension of natural language arguments. In *ACL*, pages 4658–4664, Florence, Italy.
- Nivre, J. (2004). Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.
- Noreen, E. W. (1989). *Computer-intensive Methods for Testing Hypotheses*. Wiley New York.
- Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., Hajič, J., and Urešová, Z. (2015). SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *SemEval*, pages 915–926, Denver, Colorado.
- Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Flickinger, D., Hajič, J., Ivanova, A., and Zhang, Y. (2014). SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *SemEval*, pages 63–72, Dublin, Ireland.
- Opitz, J., Parcalabescu, L., and Frank, A. (2020). AMR similarity metrics from principles. *TACL*, 8:522–538.
- Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.
- Parsons, T. (1990). Events in the semantics of English. A study in subatomic semantics.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

- Peng, F., Schuurmans, D., Keselj, V., and Wang, S. (2003). Language independent authorship attribution with character level n-grams. In *EACL*, Budapest, Hungary.
- Peng, H., Thomson, S., and Smith, N. A. (2017a). Deep multitask learning for semantic dependency parsing. In *ACL (Volume 1: Long Papers)*, pages 2037–2048, Vancouver, Canada.
- Peng, H., Thomson, S., Swayamdipta, S., and Smith, N. A. (2018). Learning joint semantic parsers from disjoint data. In *NAACL, Volume 1 (Long Papers)*, pages 1492–1502, New Orleans, Louisiana.
- Peng, X., Song, L., and Gildea, D. (2015). A synchronous hyperedge replacement grammar based approach for AMR parsing. In *CoNLL*, pages 32–41, Beijing, China.
- Peng, X., Wang, C., Gildea, D., and Xue, N. (2017b). Addressing the data sparsity issue in neural AMR parsing. In *EACL: Volume 1, Long Papers*, pages 366–375, Valencia, Spain.
- Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar.
- Pereira, F. and Shieber, S. (1987). *Prolog and Natural Language Analysis*. CSLI Lecture Notes 10. Chicago University Press, Stanford.
- Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana.
- Pieraccini, R., Tzoukermann, E., Gorelov, Z., Gauvain, J.-L., Levin, E., Lee, C.-H., and Wilpon, J. G. (1992). A speech understanding system based on statistical representation of semantics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 193–196. IEEE Computer Society.
- Plank, B., Søgaard, A., and Goldberg, Y. (2016). Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany.
- Poon, H. (2013). Grounded unsupervised semantic parsing. In *ACL (Volume 1: Long Papers)*, pages 933–943, Sofia, Bulgaria.
- Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*.

- Postma, M., van Miltenburg, E., Segers, R., Schoen, A., and Vossen, P. (2016). Open Dutch WordNet. In *Proceedings of the Eight Global Wordnet Conference*, Bucharest, Romania.
- Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B. (2008). The Penn discourse TreeBank 2.0. In *LREC*, Marrakech, Morocco. ELRA.
- Press, O., Smith, N. A., and Levy, O. (2020). Improving transformer models by reordering their sublayers. In *ACL*, pages 2996–3005, Online.
- Pust, M., Hermjakob, U., Knight, K., Marcu, D., and May, J. (2015). Parsing English into Abstract Meaning Representation using syntax-based machine translation. In *EMNLP*, pages 1143–1154, Lisbon, Portugal.
- Pustejovsky, J., Lai, K., and Xue, N. (2019). Modeling quantification and scope in Abstract Meaning Representations. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 28–33, Florence, Italy.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.
- Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
- Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *TACL*, 8:842–866.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA.
- Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389, Lisbon, Portugal.
- Santos, C. D. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pages 1818–1826.

- Schank, R. C. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3(4):552–631.
- Schank, R. C. (1975). *Conceptual Information Processing*. Elsevier Science Inc., USA.
- Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.
- Sejnowski, T. J. and Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex systems*, 1(1):145–168.
- Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., HITSCHLER, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nädejde, M. (2017). Nematus: A toolkit for neural machine translation. In *EACL, Software Demonstrations*, pages 65–68, Valencia, Spain.
- Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, volume 1, pages 83–91.
- Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, volume 2, pages 371–376, Berlin, Germany.
- Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural machine translation of rare words with subword units. In *ACL (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.
- Sobrevilla Cabezudo, M. A. and Pardo, T. (2019). Towards a general Abstract Meaning Representation corpus for Brazilian Portuguese. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 236–244, Florence, Italy.
- Song, L. and Gildea, D. (2019). SemBleu: A robust metric for AMR parsing evaluation. In *ACL*, pages 4547–4552, Florence, Italy.
- Song, L., Gildea, D., Zhang, Y., Wang, Z., and Su, J. (2019). Semantic neural machine translation using AMR. *TACL*, 7:19–31.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- Stanovsky, G. and Dagan, I. (2018). Semantics as a foreign language. In *EMNLP*, pages 2412–2421, Brussels, Belgium.

- Susanto, R. H. and Lu, W. (2017). Neural architectures for multilingual semantic parsing. In *ACL (Volume 2: Short Papers)*, pages 38–44, Vancouver, Canada.
- Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Templeton, M. and Burger, J. (1983). Problems in natural-language interface to DBMS with examples from EUFID. In *First Conference on Applied Natural Language Processing*, pages 3–16, Santa Monica, California, USA.
- Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*.
- Toral, A. and Sánchez-Cartagena, V. M. (2017). A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *EACL: Volume 1, Long Papers*, pages 1063–1073, Valencia, Spain.
- van der Goot, R., Üstün, A., Ramponi, A., and Plank, B. (2020). Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. *arXiv preprint arXiv:2005.14672*.
- van der Goot, R., van Noord, R., and van Noord, G. (2018). A taxonomy for in-depth evaluation of normalization for user generated content. In *LREC*, Miyazaki, Japan. ELRA.
- Van der Sandt, R. A. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics*, 9(4):333–377.
- van Eijck, J. and Kamp, H. (1997). Representing discourse in context. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Language*, pages 179–240. Elsevier, MIT.
- van Noord, R. (2019). Neural Boxer at the IWCS shared task on DRS parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden.
- van Noord, R., Abzianidze, L., Haagsma, H., and Bos, J. (2018a). Evaluating scoped meaning representations. In *LREC*, pages 1685–1693, Paris, France. ELRA.
- van Noord, R., Abzianidze, L., Toral, A., and Bos, J. (2018b). Exploring neural methods for parsing discourse representation structures. *TACL*, 6:619–633.

- van Noord, R. and Bos, J. (2017a). Dealing with co-reference in neural semantic parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*, pages 41–49, Montpellier, France.
- van Noord, R. and Bos, J. (2017b). The Meaning Factory at SemEval-2017 task 9: Producing AMRs with neural semantic parsing. In *SemEval*, pages 929–933, Vancouver, Canada.
- van Noord, R. and Bos, J. (2017c). Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.
- van Noord, R., Toral, A., and Bos, J. (2019). Linguistic information in neural semantic parsing with multiple encoders. In *IWCS - Short Papers*, pages 24–31, Gothenburg, Sweden.
- van Noord, R., Toral, A., and Bos, J. (2020). Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *EMNLP*, pages 4587–4603, Online. ACL.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, California, USA.
- Veenhoven, R., Snijders, S., van der Hall, D., and van Noord, R. (2018). Using translated data to improve deep learning author profiling models. In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*, volume 2125.
- Venhuizen, N. J. (2015). *Projection in Discourse: A data-driven formal semantic analysis*. PhD thesis, University of Groningen.
- Venhuizen, N. J., Bos, J., Hendriks, P., and Brouwer, H. (2018). Discourse semantics with information structure. *Journal of Semantics*, 35(1):127–169.
- Vilar, D., Peter, J.-T., and Ney, H. (2007). Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.
- Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781.
- Wada, H. and Asher, N. (1986). BUILDERS: An implementation of DR theory and LFG. In *COLING*, pages 540–545, Bonn, Germany.

- Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. (2019). Do NLP models know numbers? Probing numeracy in embeddings. In *EMNLP-IJCNLP*, pages 5307–5315, Hong Kong, China.
- Wang, C., Xue, N., and Pradhan, S. (2015a). Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *ACL-IJCNLP (Volume 2: Short Papers)*, pages 857–862, Beijing, China.
- Wang, C., Xue, N., and Pradhan, S. (2015b). A transition-based algorithm for AMR parsing. In *NAACL*, pages 366–375, Denver, Colorado.
- Weaver, W. (1955). Translation. *Machine translation of languages*, 14(15-23):10.
- Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.
- Werling, K., Angeli, G., and Manning, C. D. (2015). Robust subgraph generation improves Abstract Meaning Representation parsing. In *ACL (Volume 1: Long Papers)*, pages 982–991, Beijing, China.
- White, A. S., Reisinger, D., Sakaguchi, K., Vieira, T., Zhang, S., Rudinger, R., Rawlins, K., and Van Durme, B. (2016). Universal compositional semantics on universal dependencies. In *EMNLP*, pages 1713–1723, Austin, Texas.
- Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York. IRE.
- Wilks, Y. (1972). An artificial intelligence approach to machine translation. Technical report, Stanford University CA Dept. of Computer Science.
- Wilks, Y. (1975). A preferential, pattern-seeking, semantics for natural language inference. *Artificial intelligence*, 6(1):53–74.
- Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.
- Winograd, T. (1972). Understanding natural language. *Cognitive psychology*, 3(1):1–191.
- Wong, Y. W. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *NAACL, Main Conference*, pages 439–446, New York City, USA.
- Wong, Y. W. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, pages 960–967, Prague, Czech Republic.

- Woods, W., Kaplan, R., Kaplan, R., and Nash-Webber, B. (1972). *The Lunar Sciences Natural Language Information System: Final Report*. BBN report 2378. Bolt Beranek and Newman.
- Xiao, C., Dymetman, M., and Gardent, C. (2016). Sequence-based structured prediction for semantic parsing. In *ACL (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany.
- Xue, N., Bojar, O., Hajič, J., Palmer, M., Urešová, Z., and Zhang, X. (2014). Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In *LREC*, pages 1765–1772, Reykjavik, Iceland. ELRA.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.
- Yin, P. and Neubig, G. (2017). A syntactic neural model for general-purpose code generation. In *ACL (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada.
- Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.
- Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, Portland, Oregon, USA.
- Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666.
- Zhang, S., Ma, X., Duh, K., and Van Durme, B. (2019a). AMR parsing as sequence-to-graph transduction. In *ACL*, pages 80–94, Florence, Italy.
- Zhang, S., Ma, X., Duh, K., and Van Durme, B. (2019b). Broad-coverage semantic parsing as transduction. In *EMNLP-IJCNLP*, pages 3786–3798, Hong Kong, China.
- Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

- Zhou, J., Xu, F., Uszkoreit, H., Qu, W., Li, R., and Gu, Y. (2016). AMR parsing with an incremental joint model. In *EMNLP*, pages 680–689, Austin, Texas.
- Zhou, Q., Zhang, Y., Ji, D., and Tang, H. (2020). AMR parsing with latent structural information. In *ACL*, pages 4306–4319, Online.
- Zoph, B. and Knight, K. (2016). Multi-source neural translation. In *NAACL*, pages 30–34, San Diego, California.

Samenvatting

Mensen en computers spreken niet dezelfde taal. Het zou echter heel handig zijn als we wél in natuurlijke taal met computers zouden kunnen communiceren. In dat geval moet de computer taal niet zien als slechts een verzameling woorden, maar ook de diepere betekenis zien te begrijpen. Een manier om dit voor elkaar te krijgen is *semantisch ontleden*, of in andere woorden, het automatisch toekennen van een gestructureerde representatie van de betekenis aan iedere tekst. Dit soort representaties zijn gemakkelijker te begrijpen voor een computer.

Eerder onderzoek probeerde dit in eerste instantie op te lossen aan de hand van de syntactische ontleding van de tekst. Hiermee werd succes geboekt, maar al snel werd duidelijk dat de betekenis meer is dan alleen de syntactische ontleding. Zie bijvoorbeeld de volgende twee voorbeelden:

(1) Marie lag dubbel om een paal.

(2) Marie lag dubbel om een grap.

Deze twee zinnen hebben precies dezelfde syntactische ontleding, maar hebben overduidelijk een andere betekenis. Een accurate semantische ontleder zal rekening moeten houden met zowel de betekenis van individuele woorden, als hoe ze gecombineerd worden in een zin. Er zijn een aantal onderzoeken die proberen een betekenis component toe te voegen aan de syntactisch ontleding. Dit werkte tot op zeker hoogte, maar het heeft ook een nadeel: deze betekenis componenten moesten met de hand toegevoegd worden en werkten daarom slecht buiten het specifieke gebied waarvoor ze gecreëerd waren. We hebben liever een ontleder die automatisch kan *leren* om de betekenis representaties te produceren.

Er zijn inderdaad een aantal studies die deze methode hanteren, met soms indrukwekkende resultaten. Echter ze zijn nog steeds afhankelijk van taalkundige hulpmiddelen, zoals syntactische analyses of vooraf gedefinieerde woordenlijsten. We zouden het liefst een ontleder hebben die betekenis representaties kan produceren zonder afhankelijk te zijn van zulke middelen. In deze dissertatie werken we daarom met een neuraal netwerk dat een

sequence-to-sequence model genoemd wordt. Dit model kan automatisch leren om een sequentie van inputs (letters of woorden) te transformeren tot een sequentie van outputs (delen van de betekenis representatie), zonder andere taalkundige hulpmiddelen te gebruiken. Dit model wordt in detail beschreven in **Deel I**, samen met een overzicht van andere semantische ontleders tot dusver.

We passen dit sequence-to-sequence model toe op twee bekende formalismen van betekenis representaties: *Abstract Meaning Representation* (AMR, **Deel II**) en *Discourse Representation Structures* (DRS, **Deel III**). Een belangrijk verschil tussen de formalismen is dat DRS meer taalkundige fenomenen accuraat probeert te modelleren, waar het ook meer variabelen voor nodig heeft. Deze variabelen zijn een uitdaging voor ons model, aangezien de namen van de variabelen willekeurig gekozen zijn en geen betekenis hebben. Een andere belangrijke vraag is wat voor ons model de beste representatie van de input tekst is. Intuïtief gezien is dit een sequentie van woorden, maar het is ook mogelijk om een sequentie van letters te gebruiken. Het gebruiken van letters heeft bijvoorbeeld als voordeel dat het model beter kan omgaan met onbekende woorden en spelfouten. Een laatste uitdaging is het feit dat sequence-to-sequence modellen erom bekend te staan veel data nodig te hebben om te kunnen leren, terwijl er voor de AMR en DRS formalismen maar relatief weinig data beschikbaar is.

We vinden uiteindelijk dat ons sequence-to-sequence model accurate betekenis representaties kan genereren, voor zowel AMR als DRS. We krijgen de beste resultaten door inderdaad letters als input sequentie te gebruiken en de variabelen te herschrijven gebaseerd op de volgorde waarin ze geïntroduceerd werden. Het data-probleem lossen we op door ons model ook te laten leren van automatisch gegenereerde data van andere semantische ontleders, wat de resultaten flink verbeterde voor zowel AMR als DRS. Onze modellen gebruikten tot dusver geen enkel taalkundig hulpmiddel, maar voor DRS hebben we onderzocht of we de resultaten konden verbeteren als we zulke middelen wel zouden gebruiken. Uiteindelijk vonden we dat dit inderdaad mogelijk is, maar slechts met een kleine marge en niet voor alle verschillende middelen.

In **Deel IV** houden we ons bezig met een nieuwe ontwikkeling die zich tijdens het schrijven van deze dissertatie heeft voorgedaan. Kort gezegd werd ontdekt dat het niet verstandig is om voor elke (taal-gebaseerde) taak een nieuw model te gebruiken, maar modellen die oorspronkelijk bedoeld waren voor andere taken te hergebruiken. Het idee is dat elk model een bepaalde kennis van taal in het algemeen heeft opgedaan, die nuttig kan zijn voor iedere andere taal-gebaseerde taak, zoals in ons geval semantisch ontleden. Het grootste en belangrijkste model, dat BERT genoemd wordt, wordt gebruikt door vrijwel iedereen in het vakgebied

van de computationele taalkunde. Een belangrijk aspect van dit model is dat het geen letters of woorden als input sequentie gebruikt, maar een automatisch gegenereerde representatie die tussen woorden en letters in zit.

De vraag die we beantwoorden is de volgende: gegeven dit nieuwe BERT model, zijn letter-representaties nog steeds nuttig voor semantisch ontleden? We ontwikkelen twee methodes om letters en BERT te kunnen combineren en testen deze methodes op zowel AMR als DRS. Voor DRS kijken we nu niet alleen naar het Engels, maar ook naar het Duits, Italiaans en Nederlands. We vinden uiteindelijk dat de letter-representaties nog steeds nuttig zijn, met verbeterde scores voor beide methodes. Deze verbeteringen zijn klein, maar wel significant, en we vinden ze voor beide formalismen en alle vier talen. De toegepaste methodes zijn niet specifiek gericht op semantisch ontleden en kunnen wellicht ook toepasbaar zijn op andere taal-gebaseerde taken.

In **Deel V** reflecteren we op de belangrijkste bevindingen van deze dissertatie. Sequence-to-sequence modellen kunnen betekenis representaties van hoge kwaliteit produceren. De beste resultaten vinden we door letters als input sequentie te gebruiken. Het is ook belangrijk om op een slimme manier met de variabelen om te gaan en te zorgen dat het model genoeg data tot zijn beschikking heeft om van te leren. De letter-representaties kunnen zelfs nog nuttig zijn in combinatie met het nieuwe BERT model. Tenslotte geloven we dat toekomstig werk, gezien we nu een model hebben dat accurate betekenis representaties kan generen, moet onderzoeken of we deze representaties inderdaad kunnen gebruiken om mens-computer interactie te verbeteren.

1. Henriëtte de Swart (1991). *Adverbs of Quantification: A Generalized Quantifier Approach*.
2. Eric Hoekstra (1991). *Licensing Conditions on Phrase Structure*.
3. Dicky Gilbers (1992). *Phonological Networks. A Theory of Segment Representation*.
4. Helen de Hoop (1992). *Case Configuration and Noun Phrase Interpretation*.
5. Gosse Bouma (1993). *Nonmonotonicity and Categorical Unification Grammar*.
6. Peter I. Blok (1993). *The Interpretation of Focus*.
7. Roelien Bastiaanse (1993). *Studies in Aphasia*.
8. Bert Bos (1993). *Rapid User Interface Development with the Script Language Gist*.
9. Wim Kosmeijer (1993). *Barriers and Licensing*.
10. Jan-Wouter Zwart (1993). *Dutch Syntax: A Minimalist Approach*.
11. Mark Kas (1993). *Essays on Boolean Functions and Negative Polarity*.
12. Ton van der Wouden (1994). *Negative Contexts*.
13. Joop Houtman (1994). *Coordination and Constituency: A Study in Categorical Grammar*.
14. Petra Hendriks (1995). *Comparatives and Categorical Grammar*.
15. Maarten de Wind (1995). *Inversion in French*.
16. Jelly Julia de Jong (1996). *The Case of Bound Pronouns in Peripheral Romance*.
17. Sjoukje van der Wal (1996). *Negative Polarity Items and Negation: Tandem Acquisition*.
18. Anastasia Giannakidou (1997). *The Landscape of Polarity Items*.
19. Karen Lattewitz (1997). *Adjacency in Dutch and German*.
20. Edith Kaan (1997). *Processing Subject-Object Ambiguities in Dutch*.
21. Henny Klein (1997). *Adverbs of Degree in Dutch*.
22. Leonie Bosveld-de Smet (1998). *On Mass and Plural Quantification: The case of French 'des'/'du'-NPs*.
23. Rita Landeweerd (1998). *Discourse semantics of perspective and temporal structure*.
24. Mettina Veenstra (1998). *Formalizing the Minimalist Program*.
25. Roel Jonkers (1998). *Comprehension and Production of Verbs in aphasic Speakers*.
26. Erik F. Tjong Kim Sang (1998). *Machine Learning of Phonotactics*.
27. Paulien Rijkhoek (1998). *On Degree Phrases and Result Clauses*.
28. Jan de Jong (1999). *Specific Language Impairment in Dutch: Inflectional Morphology and Argument Structure*.
29. H. Wee (1999). *Definite Focus*.
30. Eun-Hee Lee (2000). *Dynamic and Stative Information in Temporal Reasoning: Korean tense and aspect in discourse*.

31. Ivilin P. Stoianov (2001). *Connectionist Lexical Processing*.
32. Klarien van der Linde (2001). *Sonority substitutions*.
33. Monique Lamers (2001). *Sentence processing: using syntactic, semantic, and thematic information*.
34. Shalom Zuckerman (2001). *The Acquisition of "Optional" Movement*.
35. Rob Koeling (2001). *Dialogue-Based Disambiguation: Using Dialogue Status to Improve Speech Understanding*.
36. Esther Ruigendijk (2002). *Case assignment in Agrammatism: a cross-linguistic study*.
37. Tony Mullen (2002). *An Investigation into Compositional Features and Feature Merging for Maximum Entropy-Based Parse Selection*.
38. Nanette Bienfait (2002). *Grammatica-onderwijs aan allochtone jongeren*.
39. Dirk-Bart den Ouden (2002). *Phonology in Aphasia: Syllables and segments in level-specific deficits*.
40. Rienk Withaar (2002). *The Role of the Phonological Loop in Sentence Comprehension*.
41. Kim Sauter (2002). *Transfer and Access to Universal Grammar in Adult Second Language Acquisition*.
42. Laura Sabourin (2003). *Grammatical Gender and Second Language Processing: An ERP Study*.
43. Hein van Schie (2003). *Visual Semantics*.
44. Lilia Schürcks-Grozeva (2003). *Binding and Bulgarian*.
45. Stasinos Konstantopoulos (2003). *Using ILP to Learn Local Linguistic Structures*.
46. Wilbert Heeringa (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance*.
47. Wouter Jansen (2004). *Laryngeal Contrast and Phonetic Voicing: A Laboratory Phonology*.
48. Judith Rispens (2004). *Syntactic and phonological processing in developmental dyslexia*.
49. Danielle Bougaïré (2004). *L'approche communicative des campagnes de sensibilisation en santé publique au Burkina Faso: Les cas de la planification familiale, du sida et de l'excision*.
50. Tanja Gaustad (2004). *Linguistic Knowledge and Word Sense Disambiguation*.
51. Susanne Schoof (2004). *An HPSG Account of Nonfinite Verbal Complements in Latin*.
52. M. Begoña Villada Moirón (2005). *Data-driven identification of fixed expressions and their modifiability*.
53. Robbert Prins (2005). *Finite-State Pre-Processing for Natural Language Analysis*.
54. Leonoor van der Beek (2005). *Topics in Corpus-Based Dutch Syntax*.
55. Keiko Yoshioka (2005). *Linguistic and gestural introduction and tracking of referents in L1 and L2 discourse*.

56. Sible Andringa (2005). *Form-focused instruction and the development of second language proficiency.*
57. Joanneke Prenger (2005). *Taal telt! Een onderzoek naar de rol van taalvaardigheid en tekstbegrip in het realistisch wiskundeonderwijs.*
58. Neslihan Kansu-Yetkiner (2006). *Blood, Shame and Fear: Self-Presentation Strategies of Turkish Women's Talk about their Health and Sexuality.*
59. Mónika Z. Zempléni (2006). *Functional imaging of the hemispheric contribution to language processing.*
60. Maartje Schreuder (2006). *Prosodic Processes in Language and Music.*
61. Hidetoshi Shiraishi (2006). *Topics in Nivkh Phonology.*
62. Tamás Biró (2006). *Finding the Right Words: Implementing Optimality Theory with Simulated Annealing.*
63. Dieuwke de Goede (2006). *Verbs in Spoken Sentence Processing: Unraveling the Activation Pattern of the Matrix Verb.*
64. Eleonora Rossi (2007). *Clitic production in Italian agrammatism.*
65. Holger Hopp (2007). *Ultimate Attainment at the Interfaces in Second Language Acquisition: Grammar and Processing.*
66. Gerlof Bouma (2008). *Starting a Sentence in Dutch: A corpus study of subject- and object-fronting.*
67. Julia Klitsch (2008). *Open your eyes and listen carefully. Auditory and audiovisual speech perception and the McGurk effect in Dutch speakers with and without aphasia.*
68. Janneke ter Beek (2008). *Restructuring and Infinitival Complements in Dutch.*
69. Jori Mur (2008). *Off-line Answer Extraction for Question Answering.*
70. Lonneke van der Plas (2008). *Automatic Lexico-Semantic Acquisition for Question Answering.*
71. Arjen Versloot (2008). *Mechanisms of Language Change: Vowel reduction in 15th century West Frisian.*
72. Ismail Fahmi (2009). *Automatic term and Relation Extraction for Medical Question Answering System.*
73. Tuba Yarbay Duman (2009). *Turkish Agrammatic Aphasia: Word Order, Time Reference and Case.*
74. Maria Trofimova (2009). *Case Assignment by Prepositions in Russian Aphasia.*
75. Rasmus Steinkrauss (2009). *Frequency and Function in WH Question Acquisition. A Usage-Based Case Study of German L1 Acquisition.*
76. Marjolein Deunk (2009). *Discourse Practices in Preschool. Young Children's Participation in Everyday Classroom Activities.*

77. Sake Jager (2009). *Towards ICT-Integrated Language Learning: Developing an Implementation Framework in terms of Pedagogy, Technology and Environment.*
78. Francisco Dellatorre Borges (2010). *Parse Selection with Support Vector Machines.*
79. Geoffrey Andogah (2010). *Geographically Constrained Information Retrieval.*
80. Jacqueline van Kruiningen (2010). *Onderwijsontwerp als conversatie. Probleemoplossing in interprofessioneel overleg.*
81. Robert G. Shackleton (2010). *Quantitative Assessment of English-American Speech Relationships.*
82. Tim Van de Cruys (2010). *Mining for Meaning: The Extraction of Lexico-semantic Knowledge from Text.*
83. Therese Leinonen (2010). *An Acoustic Analysis of Vowel Pronunciation in Swedish Dialects.*
84. Erik-Jan Smits (2010). *Acquiring Quantification. How Children Use Semantics and Pragmatics to Constrain Meaning.*
85. Tal Caspi (2010). *A Dynamic Perspective on Second Language Development.*
86. Teodora Mehotcheva (2010). *After the fiesta is over. Foreign language attrition of Spanish in Dutch and German Erasmus Student.*
87. Xiaoyan Xu (2010). *English language attrition and retention in Chinese and Dutch university students.*
88. Jelena Prokić (2010). *Families and Resemblances.*
89. Radek Šimik (2011). *Modal existential wh-constructions.*
90. Katrien Colman (2011). *Behavioral and neuroimaging studies on language processing in Dutch speakers with Parkinson's disease.*
91. Siti Mina Tamah (2011). *A Study on Student Interaction in the Implementation of the Jigsaw Technique in Language Teaching.*
92. Aletta Kwant (2011). *Geraakt door prentenboeken. Effecten van het gebruik van prentenboeken op de sociaal-emotionele ontwikkeling van kleuters.*
93. Marlies Kluck (2011). *Sentence amalgamation.*
94. Anja Schüppert (2011). *Origin of asymmetry: Mutual intelligibility of spoken Danish and Swedish.*
95. Peter Nabende (2011). *Applying Dynamic Bayesian Networks in Transliteration Detection and Generation.*
96. Barbara Plank (2011). *Domain Adaptation for Parsing.*
97. Cagri Coltekin (2011). *Catching Words in a Stream of Speech: Computational simulations of segmenting transcribed child-directed speech.*

98. Dörte Hessler (2011). *Audiovisual Processing in Aphasic and Non-Brain-Damaged Listeners: The Whole is More than the Sum of its Parts.*
99. Herman Heringa (2012). *Appositional constructions.*
100. Diana Dimitrova (2012). *Neural Correlates of Prosody and Information Structure.*
101. Harwintha Anjarningsih (2012). *Time Reference in Standard Indonesian Agrammatic Aphasia.*
102. Myrte Gosen (2012). *Tracing learning in interaction. An analysis of shared reading of picture books at kindergarten.*
103. Martijn Wieling (2012). *A Quantitative Approach to Social and Geographical Dialect Variation.*
104. Gisi Cannizzaro (2012). *Early word order and animacy.*
105. Kostadin Cholakov (2012). *Lexical Acquisition for Computational Grammars. A Unified Model.*
106. Karin Beijering (2012). *Expressions of epistemic modality in Mainland Scandinavian. A study into the lexicalization-grammaticalization-pragmaticalization interface.*
107. Veerle Baaijen (2012). *The development of understanding through writing.*
108. Jacolien van Rij (2012). *Pronoun processing: Computational, behavioral, and psychophysiological studies in children and adults.*
109. Ankelen Schippers (2012). *Variation and change in Germanic long-distance dependencies.*
110. Hanneke Loerts (2012). *Uncommon gender: Eyes and brains, native and second language learners, & grammatical gender.*
111. Marjoleine Sloos (2013). *Frequency and phonological grammar: An integrated approach. Evidence from German, Indonesian, and Japanese.*
112. Aysa Arylova. (2013) *Possession in the Russian clause. Towards dynamicity in syntax.*
113. Daniël de Kok (2013). *Reversible Stochastic Attribute-Value Grammars.*
114. Gideon Kotzé (2013). *Complementary approaches to tree alignment: Combining statistical and rule-based methods.*
115. Fridah Katshemererwe (2013). *Computational Morphology and Bantu Language Learning: an Implementation for Runyakitara.*
116. Ryan C. Taylor (2013). *Tracking Referents: Markedness, World Knowledge and Pronoun Resolution.*
117. Hana Smiskova-Gustafsson (2013). *Chunks in L2 Development: A Usage-based Perspective.*
118. Milada Walková (2013). *The aspectual function of particles in phrasal verbs.*
119. Tom O. Abuom (2013). *Verb and Word Order Deficits in Swahili-English bilingual agrammatic speakers.*

120. Gülsen Yılmaz (2013). *Bilingual Language Development among the First Generation Turkish Immigrants in the Netherlands*.
121. Trevor Benjamin (2013). *Signaling Trouble: On the linguistic design of other-initiation of repair in English conversation*.
122. Nguyen Hong Thi Phuong (2013). *A Dynamic Usage-based Approach to Second Language Teaching*.
123. Harm Brouwer (2014). *The Electrophysiology of Language Comprehension: A Neurocomputational Model*.
124. Kendall Decker (2014). *Orthography Development for Creole Languages*.
125. Laura S. Bos (2015). *The Brain, Verbs, and the Past: Neurolinguistic Studies on Time Reference*.
126. Rimke Groenewold (2015). *Direct and indirect speech in aphasia: Studies of spoken discourse production and comprehension*.
127. Huiping Chan (2015). *A Dynamic Approach to the Development of Lexicon and Syntax in a Second Language*.
128. James Griffiths (2015). *On appositives*.
129. Pavel Rudnev (2015). *Dependency and discourse-configurationality: A study of Avar*.
130. Kirsten Kolstrup (2015). *Opportunities to speak. A qualitative study of a second language in use*.
131. Güliz Güneş (2015). *Deriving Prosodic structures*.
132. Cornelia Lahmann (2015). *Beyond barriers. Complexity, accuracy, and fluency in long-term L2 speakers' speech*.
133. Sri Wachyunni (2015). *Scaffolding and Cooperative Learning: Effects on Reading Comprehension and Vocabulary Knowledge in English as a Foreign Language*.
134. Albert Walsweer (2015). *Ruimte voor leren. Een etnografisch onderzoek naar het verloop van een interventie gericht op versterking van het taalgebruik in een knowledge building environment op kleine Friese basisscholen*.
135. Aleyda Lizeth Linares Calix (2015). *Raising Metacognitive Genre Awareness in L2 Academic Readers and Writers*.
136. Fathima Mufeeda Irshad (2015). *Second Language Development through the Lens of a Dynamic Usage-Based Approach*.
137. Oscar Strik (2015). *Modelling analogical change. A history of Swedish and Frisian verb inflection*.
138. He Sun (2015). *Predictors and stages of very young child EFL learners' English development in China*.

139. Marieke Haan (2015). *Mode Matters. Effects of survey modes on participation and answering behavior.*
140. Nienke Houtzager (2015). *Bilingual advantages in middle-aged and elderly populations.*
141. Noortje Joost Venhuizen (2015). *Projection in Discourse: A data-driven formal semantic analysis.*
142. Valerio Basile (2015). *From Logic to Language: Natural Language Generation from Logical Forms.*
143. Jinxing Yue (2016). *Tone-word Recognition in Mandarin Chinese: Influences of lexical-level representations.*
144. Seçkin Arslan (2016). *Neurolinguistic and Psycholinguistic Investigations on Evidentiality in Turkish.*
145. Rui Qin (2016). *Neurophysiological Studies of Reading Fluency. Towards Visual and Auditory Markers of Developmental Dyslexia.*
146. Kashmiri Stec (2016). *Visible Quotation: The Multimodal Expression of Viewpoint.*
147. Yinxing Jin (2016). *Foreign language classroom anxiety: A study of Chinese university students of Japanese and English over time.*
148. Joost Hurkmans (2016). *The Treatment of Apraxia of Speech. Speech and Music Therapy, an Innovative Joint Effort.*
149. Franziska Köder (2016). *Between direct and indirect speech: The acquisition of pronouns in reported speech.*
150. Femke Swarte (2016). *Predicting the mutual intelligibility of Germanic languages from linguistic and extra-linguistic factors.*
151. Sanne Kuijper (2016). *Communication abilities of children with ASD and ADHD. Production, comprehension, and cognitive mechanisms.*
152. Jelena Golubović (2016). *Mutual intelligibility in the Slavic language area.*
153. Nynke van der Schaaf (2016). *"Kijk eens wat ik kan!" Sociale praktijken in de interactie tussen kinderen van 4 tot 8 jaar in de buitenschoolse opvang.*
154. Simon Šuster (2016). *Empirical studies on word representations.*
155. Kilian Evang (2016). *Cross-lingual Semantic Parsing with Categorical Grammars.*
156. Miren Arantzeta Pérez (2017). *Sentence comprehension in monolingual and bilingual aphasia: Evidence from behavioral and eye-tracking methods.*
157. Sana-e-Zehra Haidry (2017). *Assessment of Dyslexia in the Urdu Language.*
158. Srđan Popov (2017). *Auditory and Visual ERP Correlates of Gender Agreement Processing in Dutch and Italian.*

159. Molood Sadat Safavi (2017). *The Competition of Memory and Expectation in Resolving Long-Distance Dependencies: Psycholinguistic Evidence from Persian Complex Predicates.*
160. Christopher Bergmann (2017). *Facets of native-likeness: First-language attrition among German emigrants to Anglophone North America.*
161. Stefanie Keulen (2017). *Foreign Accent Syndrome: A Neurolinguistic Analysis.*
162. Franz Manni (2017). *Linguistic Probes into Human History.*
163. Margreet Vogelzang (2017). *Reference and cognition: Experimental and computational cognitive modeling studies on reference processing in Dutch and Italian.*
164. Johannes Bjerva (2017). *One Model to Rule them all. Multitask and Multilingual Modelling for Lexical Analysis: Multitask and Multilingual Modelling for Lexical Analysis.*
165. Dieke Oele (2018). *Automated translation with interlingual word representations.*
166. Lucas Seuren (2018). *The interactional accomplishment of action.*
167. Elisabeth Borleffs (2018). *Cracking the code - Towards understanding, diagnosing and remediating dyslexia in Standard Indonesian.*
168. Mirjam Günther-van der Meij (2018). *The impact of degree of bilingualism on L3 development English language development in early and later bilinguals in the Frisian context.*
169. Ruth Koops van 't Jagt (2018). *Show, don't just tell: Photo stories to support people with limited health literacy.*
170. Bernat Bardagil-Mas (2018). *Case and agreement in Panará.*
171. Jessica Overweg (2018). *Taking an alternative perspective on language in autism.*
172. Lennie Donné (2018). *Convincing through conversation: Unraveling the role of interpersonal health communication in health campaign effectiveness.*
173. Toivo Glatz (2018). *Serious games as a level playing field for early literacy: A behavioural and neurophysiological evaluation.*
174. Ellie van Setten (2019). *Neurolinguistic Profiles of Advanced Readers with Developmental Dyslexia.*
175. Anna Pot (2019). *Aging in multilingual Netherlands: Effects on cognition, wellbeing and health.*
176. Audrey Rouse-Malpat (2019). *Effectiveness of explicit vs. implicit L2 instruction: a longitudinal classroom study on oral and written skills.*

177. Rob van der Goot (2019). *Normalization and Parsing Algorithms for Uncertain Input*.
178. Azadeh Elmianvari (2019). *Multilingualism, Facebook and the Iranian diaspora*.
179. Joëlle Ooms (2019). *"Don't make my mistake": Narrative fear appeals in health communication*.
180. Annerose Willemsen (2019). *The floor is yours: A conversation analytic study of teachers' conduct facilitating whole-class discussions around texts*.
181. Frans Hiddink (2019). *Early childhood problem-solving interaction: Young children's discourse during small-group work in primary school*.
182. Hessel Haagsma (2020). *A Bigger Fish to Fry: Scaling up the Automatic Understanding of Idiomatic Expressions*.
183. Juliana Andrade Feiden (2020). *The Influence of Conceptual Number in Coreference Establishing: An ERP Study on Brazilian and European Portuguese*.
184. Sirkku Lesonen (2020). *Valuing variability: Dynamic usage-based principles in the L2 development of four Finnish language learners*.
185. Nathaniel Lartey (2020). *A neurolinguistic approach to the processing of resumption in Akan focus constructions*.
186. Bernard Amadeus Jaya Jap (2020). *Syntactic Frequency and Sentence Processing in Standard Indonesian*.
187. Ting Huang (2020). *Learning an L2 and L3 at the same time: help or hinder?*.
188. Anke Herder (2020). *Peer talk in collaborative writing of primary school students: A conversation analytic study of student interaction in the context of inquiry learning*.
189. Ellen Schep (2020). *Attachment in interaction: A conversation analytic study on dinner conversations with adolescents in family-style group care*.
190. Yulia Akinina (2020). *Individual behavioural patterns and neural underpinnings of verb processing in aphasia*.
191. Camila Martinez Rebolledo (2020). *Comprehending the development of reading difficulties in children with SLI*.
192. Jakolien den Hollander (2021). *Distinguishing a phonological encoding disorder from Apraxia of Speech in individuals with aphasia by using EEG*.
193. Rik van Noord (2021). *Character-based Neural Semantic Parsing*.

GRODIL

Center for Language and Cognition Groningen (CLCG)

P.O. Box 716

9700 AS Groningen

The Netherlands