Scalable Syntactic Inductive Biases for Neural Language Models

Adhiguna Surya Kuncoro
Balliol College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hillary 2022

To my family, for their unwavering support throughout this long journey, and without whom this DPhil would not mean as much.

Acknowledgements

The last five years have been an incredible journey — both in the personal and academic sense — which would not have been possible without the support of many amazing people that I have interacted with over the years.

First and foremost, I would like to thank my supervisor Phil Blunsom for his generous guidance, advice, support, patience, and encouragement throughout the course of my DPhil. Most of all, Phil taught me how to pick interesting research questions, conduct science in a rigorous way, think about the big picture and the *why*, and most importantly, how to focus on the things and ideas that will matter in the long run, above and beyond what is simply popular today.

I would also like to thank Chris Dyer, who has been my mentor and close collaborator for the past 8 years. I am grateful to Chris for taking me on as his student at CMU LTI back in 2015, despite my sparse resume and his initial plan on not taking any new students that year. I thank Chris for teaching me most of what I know about deep learning, syntax, and language, culminating in many fruitful collaborations that eventually comprised the bulk of this thesis.

I am indebted to Noah Smith, with whom I spent a great summer in 2016 interning at UW CSE. From collaborating with Noah, I learnt most of what I know about how to structure and write good research papers, and how to present the resulting ideas and findings in a clear and coherent way. I would also thank Miguel Ballesteros, who was my *de facto* second supervisor at CMU, and from whom I learnt a lot of what I know about syntax and parsing. I am also grateful to Graham Neubig, who kindly took me on during my last year at CMU.

I would like to thank Kevin Duh for hosting me during an internship at NAIST, where I started my first project on deep learning for NLP. This formative experience led me to the path where I am today, and I have been doing this line of work since. Kevin's kindness, patience, and guidance were much appreciated. Another special thank you goes to Bu Ayu Purwarianti, who introduced me to the wonderful world of statistical NLP research by supervising my undergraduate thesis in ITB on syntactic parsing for the low-resource Indonesian language.

I would like to thank my examiners Mark Steedman and Michael Bronstein for taking the time to review this thesis, and also their helpful advice, feedback, and discussion. Earlier iterations of this work also benefitted from valuable feedback from Yarin Gal, Thomas Lukasiewicz, and Matt Kusner. My DPhil study is funded by an EPSRC studentship, a Balliol Mark Sadler scholarship, and the Department of Computer Science at Oxford, for which I am very grateful.

In many ways, my DPhil journey is an unusual one, as a large chunk of it was spent at the DeepMind Language Team. In fact, I could not have asked for a better intellectual home: I have tremendously enjoyed the vibrant research community, the inspiring conversations, the diversity of the group's interests and expertise, and the collaborative and helpful nature of the group members and DeepMinders more broadly. To that end, I would like to thank the following colleagues at DeepMind and beyond for great collaborations, inspirations, and conversations (in alphabetical order): Adam Liska, Aida Nematzadeh, Aishwarya Agrawal, Alhussein Fawzi, Allison Tam, Andrei Rusu, Andrew Trask, Angeliki Lazaridou, Anna Potapenko, Arthur Douillard, Arthur Szlam, Bola Branco, Christos Kaplanis, Clara Meister, Cyprien de Masson d'Autume, Daniel Fried, Dani Yogatama, Devang Agrawal, Elena Gribovskaya, Ellen Gilsenan-McMahon, Felix Hill, Gabor Melis, Ivana Balazevic, Ivana Kajic, Jack Rae, Jiajun Shen, John Hale, John Hewitt, Jordan Hoffmann, Karl Moritz Hermann, Kazuya Kawakami, Kevin Villela, Kris Cao, Laura Rimell, Laurent Sartran, Layide Oshun, Lei Yu, Lingpeng Kong, Lisa Anne Hendricks, Marc'Aurelio Ranzato, Mikel Artetxe, Milos Stanojevic, Nathan Schuher, Nelly Papalampidi, Owen He, Ozlem Aslan, Paul Michel, Pavel Khruskov, Qixuan Feng, Rhys May, Samuel Barrett, Sebastian Farquhar, Sebastian Riedel, Sebastian Ruder, Sibon Li, Sophia Austin, Stephen Clark, Susannah Young, Tomas Kocisky, Tom Everitt, Wang Ling, Wojciech Stokowiec, Xiang Lorraine Li, Yishu Miao, Yoon Kim, Yujia Li, Yutian Chen, and many other brilliant colleagues and mentors.

I would like to thank Laura Rimell, who has been a fantastic mentor and manager at DeepMind; over the years, Laura has been very generous with her time, help, feedback, and advice. Laura also went above and beyond her call of DeepMind-managerial duty by providing valuable feedback to earlier drafts this thesis. I am also grateful to Stephen Clark for being a great manager during my early days at DeepMind, and helped me settle in well into the company.

During the course of my career thus far, I had the wonderful opportunity to co-mentor three amazing researchers during their internships at DeepMind: Xiang Lorraine Li, Clara Meister, and John Hewitt. Their enthusiasm and research quality have been an inspiration, and I learnt a lot from these collaborations. I look forward to seeing the great things they do in the years to come.

I would also like to thank my friends and collaborators in the Indonesian NLP researcher community (in alphabetical order): Alham Fikri Aji, Bryan Willie, Genta Indra Winata, Haryo Akbarianto Wibowo, Karissa Vincentio, Made Nindyatama, Radityo Eko Prasojo, and Samuel Cahyawijaya. Through the IndoNLG paper, I am glad to have the opportunity to contribute to the development of a pre-trained language model for the Indonesian language.

Co-organising SeaMLS 2019 in Jakarta was one of the highlights of the last few years. I would especially like to thank my fellow co-organisers Febria Roosita, Dani Yogatama, and Wittawat Jitkrittum for their amazing contributions and camaraderie for making the event happen; I am glad we had the opportunity to empower underrepresented machine learning communities in Southeast Asia. I would also like to thank the speakers (some of whom flew 10,000 miles just for the occasion), sponsors, local hosts and organisers, panelists, and of course the 200+ amazing participants for making SeaMLS 2019 a success.

On a more reflective note, I realise that the opportunity to complete this DPhil study — some 7,000 miles away from home — is a product of immense privilege, one that is built on top of the hard work and sacrifices of my wonderful parents, Bapak and Ibu. Thank you for providing me with the best education from my earliest years. Thank you for creating a loving and nurturing environment, one in which I have the freedom and support to do the things that I am passionate about. Thank you for believing in me, and for your unwavering support in pursuing my dreams. Thank you for flying back and forth between Jakarta and the UK to visit us many, many times — not least to help a lot with the newborn days, when I had no idea what I was doing — despite the 18-hour flight and the jetlag that each trip entails. I cannot ever repay your kindness, but I will do my best to pay it forward. A special thank you also goes to my sister Nia for her support, encouragement, camaraderie, and for her generous help with childcare each time we are in the same place. I would also like to thank my in-laws: Papa, Mama, Wayda, Risqa, Mahir, and Fawwaz, for their generous support and encouragement for me, Aulia, and Kanza during this long journey.

In truth, completing this DPhil — whilst working at DeepMind and having a child in the first year of the journey — would not have been possible without the best support system. For that, I would like to give a very special thank you to my wife Aulia, without whom none of this would have been possible. Thank you for always being there and supporting me through thick and thin. Thank you for always believing in me. Thank you for being my best friend, life partner, sounding board, and advisor-on-all-matters rolled into one. Thank you for all the fun, laughter, and the many adventures that we shared; my world is a better place since you came along. I look forward to the next phase of our journey.

And of course, to my dear little Kanza. They say that time flies when you are having fun. Indeed, the last five years flew by so fast, because I had so much fun being your father. Thank you for being a bright ray of sunshine at the end of each long day, and for all our wonderful adventures. You inspire me with your kindness and curiosity; may you nurture that all your life. I look forward to many more of our fun adventures, and to see the amazing person that you will become, but please don't grow up too quickly. And to baby Hamza, watching you grow has been the highlight of the previous year; thank you for bringing a lot of joy and fun into our lives. I really look forward to taking you on many fun adventures! I am also grateful to our wonderful nanny Helena, who takes excellent care of Kanza and Hamza, and took away all of our childcare worries. This DPhil would have been only half as productive without her help.

I would also like to thank our London friends: Dikra, Riesta, Nisa, Kiki, Yorga, Widya, Wiyan, Hanifi, Elmo, Mita, and Andra, among many others. Our game

night sessions have been a source of joy and fun amidst life's hecticness.

Last but not at all least, I would like to thank my wonderful grandparents: Kakek, Nenek, Eyang Bayat, and Eyang Sri. Thank you for being a part of my childhood, and for shaping me to become who I am today. I would also like to thank my late-aunt Bude Baby Noviani, whose care for me is much appreciated. A very special thank you goes to my late-grandmother: Nenek Tien Yuliandanti; you are dearly missed. Thank you for being my best friend in childhood; I had great fun. Thank you for taking me to and from my after-school lessons. Thank you for instilling in me the value of hard work and big dreams. Thank you for encouraging me to pursue the best and highest education that I can, wherever that path takes me. Thank you for always assuring me that I can do anything as long as I set my mind to it, work hard, and be disciplined. You asked me to pursue the best education that I possibly can. I hope I have done you proud.

Abstract

Natural language has a sequential surface form, although its underlying structure has been argued to be hierarchical and tree-structured in nature, whereby smaller linguistic units like words are recursively composed to form larger ones, such as phrases and sentences. This thesis aims to answer the following open research questions: To what extent—if at all—can more explicit notions of hierarchical syntactic structures *further improve* the performance of neural models within NLP, even within the context of successful models like BERT that learn from large amounts of data? And where exactly would stronger notions of syntactic structures be beneficial in different types of language understanding tasks?

To answer these questions, we explore two approaches for augmenting neural sequence models with an inductive bias that encourages a more explicit modelling of hierarchical syntactic structures. In the first approach, we use existing techniques that design **tree-structured** neural networks, where the ordering of the computational operations is determined by hierarchical syntax trees. We discover that this approach is indeed effective for designing better and more robust models at various challenging benchmarks of syntactic competence, although these benefits nevertheless come at the expense of **scalability**: In practice, such tree-structured models are much more challenging to scale to large datasets.

Hence, in the second approach, we devise a novel **knowledge distillation strategy** for combining the best of both syntactic inductive biases and data scale. Our proposed approach is effective across different neural sequence modelling architectures and objective functions: By applying our approach on top of a left-to-right LSTM, we design a distilled syntax-aware (DSA) LSTM that achieves a new state of the art (as of mid-2019) and human-level performance at targeted syntactic evaluations. By applying our approach on top of a Transformer-based BERT masked language model that works well at scale, we outperform a strong BERT baseline on six structured prediction tasks—including those that are not explicitly syntactic in nature—in addition to the corpus of linguistic acceptability. Notably, our approach yields a new state of the art (as of mid-2020)—among models pre-trained on the original BERT dataset—on four structured prediction tasks: In-domain and out-of-domain phrase-structure parsing, dependency parsing, and semantic role labelling.

Altogether, our findings and methods in this work: (i) provide an example of how existing linguistic theories (particularly concerning the syntax of language), annotations, and resources can be used both as diagnostic evaluation tools, and also as a source of prior knowledge for crafting inductive biases that can

improve the performance of computational models of language; (ii) showcase the *continued* relevance and benefits of more explicit syntactic inductive biases, even within the context of scalable neural models like BERT that can derive their knowledge from large amounts of data; (iii) contribute to a better understanding of where exactly syntactic biases are most helpful in different types of NLP tasks; and (iv) motivate the broader question of how we can design models that integrate stronger syntactic biases—and yet can be easily scalable at the same time—as a promising (if relatively underexplored) direction of NLP research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Hierarchical Syntax in Natural Language

Natural language is a key hallmark of human intelligence, and one of its fundamental property is that it enables humans to combine its basic linguistic units, such as morphemes and words, into larger and more complex units, such as phrases and sentences. This key property of language thus endows humans with a remarkable ability to express an unbounded number of complex thoughts and ideas from a much smaller set of words: Indeed, it is reasonable to expect someone who already knows the meaning of the colour "*blue*", and who has also been told the meaning of a novel noun "*wug*"—an imaginary cartoon character that looks rather like a bird (Berko, 1958)—to also know the meaning of a new noun phrase "*a blue wug*", even if she has never encountered this exact same phrase in the past. This essential property of language is concisely summarised by von Humboldt (1836), who remarked that "[Natural language] makes infinite use of finite means" (Chomsky, 1965).

To that end, this productive aspect of natural language can be modelled through recursive functions that can take their own outputs as inputs; for instance, the noun phrase "*a nice blue wug*" can be nested within another, larger noun phrase "*a nice blue wug playing with two children*", which in turn can be nested within an even larger noun phrase "*a nice blue wug playing with two children in the park*", and so on. Through these recursive functions, we are able to derive a finite (and therefore tractable) description of an infinite set of strings. Within the context of natural language, these recursive functions are often represented through a generative grammar in the form of phrase-structure rules, which provide a **hierarchical, tree-structured** account of part-whole relationships that exist within the sentence.

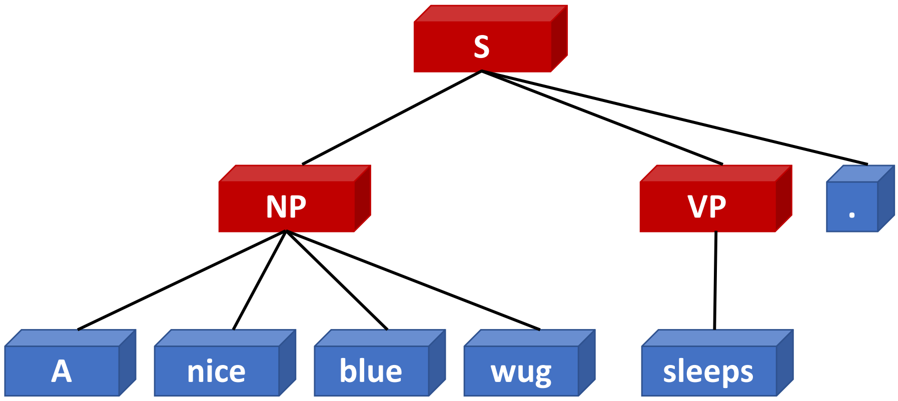


Figure 1.1: An example of a sentence "*A nice blue wug sleeps .*" (denoted in blue), and its corresponding phrase-structure tree (denoted in red). Here "NP", "VP", and "S" denote a noun phrase, a verb phrase, and a declarative clause, respectively. This phrase-structure tree represents how the words in the sentence recursively combine to form larger linguistic units—such as phrases, clauses, and eventually the whole sentence—in a hierarchical fashion.

Concretely, given an input string and a generative grammar that defines the phrase-structure rules for a particular language, we can determine whether the input string belongs to the language described by the grammar. For instance, our previous example "*a nice blue wug*" constitutes a well-formed noun phrase according to the grammar of the English language, whereas another example, "**wug blue a nice*", does not, even though both phrases are constructed from the exact same set of four underlying words. In the case of strings that *do* belong to the language, we can identify the hierarchical phrase-structure tree that represents how the words in that sentence *recursively* combine to form phrases, clauses, and eventually the whole sentence (Chomsky, 1957)—an example of which is shown in Fig. 1.1.

1.2 Motivation and Objective

Although natural language has an overt sequential surface form, its underlying latent structure is argued to be hierarchical and tree-structured (as opposed to sequential) in nature (§1.1). Nevertheless, recent rapid progress (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020, *inter alia*) within natural language processing (NLP) is *not* derived from augmenting computational models of natural language—in particular, deep neural networks—with our prior knowledge about these hierarchical structures that exist within language. On the contrary, recent progress mostly stems from the use of ever-larger deep neural architectures that lack explicit notions of hierarchical syntactic structures, but are nevertheless trained on ever-larger amounts of data (Yang et al., 2019; Liu et al., 2019b). In light of this recent success, our goal in this thesis is to answer the following open research question: To what extent—if at all—can more explicit modelling of hierarchical syntactic structures improve the performance of computational models of natural language, even in the case of powerful neural architectures that can be trained on vast amounts of data? We argue that this research question is an important one, both from the scientific (in particular, cognitive science, computational linguistics, and deep learning) and the language applications perspective.

The computational linguistics perspective. Through decades of linguistic research, prior work has made substantial progress in characterising the precise nature of this grammar, culminating in a wide variety of established grammar formalisms such as phrase-structure grammar, dependency grammar (Tesnière, 1959; Mel'čuk, 1988), combinatory categorial grammar (Steedman, 2000), and others. Furthermore, since the advent of statistical natural language processing (NLP) in the 1990s, substantial effort has been dedicated into curating a large number of sentences and their corresponding, manually-annotated hierarchical syntax trees, such as the Penn Treebank (Marcus et al., 1993) and the Universal Dependencies Treebank (Nivre et al., 2016) projects. Throughout this thesis, we aim to make progress towards answering the following open research questions in computational linguistics:

- To what extent can we adopt certain insights from existing syntactic theories, and further leverage existing syntactic annotations and resources as a valuable source of *prior knowledge* and *model inductive bias* for designing better computational models of language?
- Is a more explicit modelling of syntactic structures still relevant and beneficial, even at a time when most state of the art results in NLP are achieved by neural network models that lack explicit notions of hierarchical syntactic structures—but are nevertheless trained on very large amounts of data—such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020)? In other words, can we make these models *even better*, by augmenting them with our prior knowledge about the underlying hierarchical structures within language?
- Assuming that we find more explicit syntactic inductive biases to be beneficial, even for such neural language models that work well at scale (as we do in this work), what types of tasks would benefit the most from syntactic biases, and what types of tasks would not? Would such syntactic biases benefit all types of NLP tasks to the same extent? Or would

we instead observe different patterns of results depending on the nature of the tasks, and discover that stronger syntactic biases are not necessarily advantageous for some tasks?

The cognitive science perspective. Our overarching research question closely mirrors a long-standing open question in the cognitive science and language acquisition literature, namely: How exactly are young children able to master natural language—including the ability to generate novel sentences that conform to the underlying grammar of the language—even when they only have access to a limited amount of data (Hart and Risley, 1995), and are not provided with explicit instructions or supervision on how to do so?[1] Is an innate preference for hierarchical syntactic structures (Chomsky, 1980) a necessary prerequisite for learners that are able to infer the underlying grammar of natural language, or can such mastery still be achieved, even by learners that lack an explicit preference for hierarchical syntactic structures (*e.g.,* Perfors et al., 2011)? By answering our overarching question, we make an empirical contribution to this long-standing debate by way of computational modelling: Concretely, we find that neural language models are, in fact, largely able to perform well on multiple challenging benchmarks of syntactic competence (albeit with some important caveats, as detailed in Ch. 3). Remarkably, this success extends even to neural models that: (i) lack an explicit modelling of hierarchical syntactic structures, and (ii) do not benefit from any form of explicit syntactic supervision. To the extent that the assumptions in our computational experiments are applicable to the circumstances in which child language acquisition occurs, our findings provide evidence for the learnability of the grammar from the observed surface strings alone, and against the necessity of innate syntactic constraints for building language learners that can master the underlying grammar of the language.

Historical context. During the period of statistical NLP—a period spanning the 1990s and the 2000s where statistical (albeit mostly non-neural) techniques are used to infer patterns from textual and speech data (Jones, 1994; Abney, 1996; Manning and Schütze, 1999; Pereira, 2000)—the answer to "Are syntactic structures helpful for NLP?" was a resounding "Yes". Within this time period, hierarchical syntax trees—typically obtained from the output

[1]The difficulty of the child language acquisition process is compounded by the nature of the linguistic stimuli that children are exposed to, which: (i) consist mostly of simple utterances, where syntactically informative (*i.e.,* grammatically complex) examples are few and far in between Chomsky (1971, 1980); and (ii) consist mostly of positive examples that conform to the grammar of the language (Chomsky, 1965), with little in the way of negative examples and corrective feedback that can help disambiguate what is grammatical and what is not, although later work has disputed this assertion (Pullum, 1996; Pullum and Scholz, 2002).

of separetely-trained syntactic parsers—were shown to be useful for various NLP tasks, such as phrase-based machine translation (Blunsom and Osborne, 2008; Marton and Resnik, 2008; Chiang et al., 2008, 2009), sentiment analysis (Moilanen and Pulman, 2007; Choi and Cardie, 2008; Jia et al., 2009), semantic role labelling (Gildea and Jurafsky, 2002; Hacioglu, 2004), relation extraction (Hearst, 1992; Mintz et al., 2009), and many others.

The deep learning perspective. In contrast to the statistical NLP techniques in the 1990s and 2000s, **deep learning** provides an alternative computational approach that models natural language through end-to-end learning of expressive, multi-layer neural networks. Within NLP, commonly used deep learning models include the broader family of recurrent neural networks (Elman, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014, RNNs) and Transformer (Vaswani et al., 2017) models. In practice, both classes of models feature multiple layers of neural network operations (hence the "deep" epithet): Within RNNs, the sequential connections between adjacent inputs can be "unrolled" and interpreted as multi-layer neural network operations, while most Transformer models in practical use are comprised of multiple layers of self-attention operations interleaved with feedforward ones.

To date, deep learning techniques have achieved major successes within NLP, often outperforming prior approaches by large margins in various tasks, such as language modelling (Bengio et al., 2003; Dai et al., 2019; Radford et al., 2019; Brown et al., 2020), machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017; Chen et al., 2018b), text summarisation (See et al., 2017; Paulus et al., 2018; Narayan et al., 2018), and more recently, self-supervised representation learning that derives dense contextual representations that can improve performance on many downstream tasks (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019; Clark et al., 2020). Crucially, the end-to-end nature of neural network training departs from the traditional "pipeline" approach of statistical NLP (*e.g.,* parse the sentence first, and use the resulting syntax trees as part of a manually-crafted set of informative features for a downstream task) in one important way. Under the deep learning approach, any phenomena that are necessary for solving the task (including any relevant syntactic information) are: (i) derived *automatically* from the objective, hence eliminating the need for manually crafting a set of informative features, including potentially any syntactic ones, and (ii) captured *implicitly* within the hidden layers of the model.

Hence, from the deep learning perspective, this work contributes towards answering the following open research questions in the field:

- Within the context of natural language processing applications, how can deep learning research benefit from existing insights and knowledge from a scientific discipline that studies the domain of interest—in this case, linguistics as the study of human language? In this work, we provide an example of how targeted evaluation metrics that are inspired by human syntactic competence (Linzen et al., 2016; Marvin and Linzen, 2018; Warstadt et al., 2018) can be used to discover certain limitations in the generalisation ability of neural language models (Ch. 3), even when such limitations are not easily discernible from standard aggregate evaluation metrics, such as language modelling perplexity.
- Given the end-to-end nature of neural network training, how can we effectively inject domain knowledge and inductive biases as part of the training procedure for neural network models? To that end, we explore two approaches for injecting hierarchical syntactic inductive biases into neural language models. In the first approach, we use existing techniques that modify the neural architectures such that the *ordering* of neural network operations is determined by hierarchical syntax trees (Socher et al., 2011; Tai et al., 2015; Dyer et al., 2015, 2016; Andreas et al., 2016a,b), as opposed to the sequential positions of the input words. In the second approach, we introduce a novel knowledge distillation strategy that injects syntactic biases through the objective function (as opposed to modifying the model architecture directly), which scales better to larger datasets.

The language applications perspective. From the language applications perspective, we are interested in building the best-performing models for natural language processing that we possibly can. In light of the recent success of neural language models that are trained on large amounts of data, what, then, are the promising ways forward? Is building ever-larger neural language models, that are paired with increasingly larger amounts of training data, the only way forward, and that more explicit syntactic inductive biases are no longer helpful for these neural language models that work well at scale? Or are there *still* some benefits of more explicit modelling of hierarchical syntactic structures, where such benefits can be complementary and thus orthogonal to further advances in terms of model and data scale? Our key findings provide evidence for the latter: We find that more explicit syntactic inductive biases are, in fact, still beneficial, even for neural models like BERT that are trained on large amounts of data, and have *already* been shown to perform well at various tasks. We argue that these findings hold relevance for the broader field of NLP engineering and research: Our findings motivate the broader question of how we can design models that integrate stronger notions of syntactic inductive biases—and yet can be easily scalable at the same time—as a promising (if relatively underexplored) direction of future NLP research.

1.3 Why Syntactic Inductive Biases for Neural Models?

Our key hypothesis is that more explicit notions of hierarchical syntactic structures can still be beneficial for improving the performance of neural language models at various tasks, including for models like BERT that derive their knowledge from large amounts of textual data. This hypothesis is based on the following set of three reasons:

- Due to the recursive nature of the grammar, humans are able to produce and understand novel utterances from existing words and concepts—an ability often denoted as **systematic compositionality**. Indeed, we expect someone who knows the meaning of "*turn left*" and "*turn right*" to also know the meaning of "*turn left then turn right*", without having to observe that exact combination in the past (Lake and Baroni, 2018). Hence, we conjecture that models that have more explicit notions of syntax—where words recursively combine to form phrases, and smaller phrases combine to form larger phrases and eventually sentences—shall have better and more sample-efficient generalisation at various tasks.
- Prior work has shown that some of the known failure modes of neural models are, to some extent, related to syntactic structures; in other words, such mistakes arise due to a misidentification of the syntactic relations within the sentence. For instance, Ribeiro et al. (2020) demonstrated that, given a simple prompt "*Richard bothers Elizabeth*" and a corresponding question "*Who is bothered?*", a BERT model would incorrectly assign a higher probability to "**Richard*", even though the correct answer is clearly "*Elizabeth*" (*i.e.,* a mistake in distinguishing between the subject and the object of the sentence). These syntactic mistakes are likely to be consequential in nature, particularly because semantic processing (*i.e.,* the process that computes the meaning representation of a sentence) has been argued to work in conjunction with syntactic processing (Bach, 1976). Hence, we conjecture that neural models that are able to discover the *correct* syntactic interpretation of the sentence would be better-suited to compute its meaning representation. Throughout this thesis, we hypothesise that more explicit syntactic biases can enable models like BERT to improve on these types of syntactic failure modes, and thereby perform better.

- Despite the tremendous importance of data scale for achieving recent NLP progress (Liu et al., 2019b; Brown et al., 2020), recent findings that systematically compare the syntactic competence of models trained at varying data scales suggest that model inductive biases are, in fact, more important predictors of syntactic competence than data scale (Hu et al., 2020). Based on these findings, we conjecture that syntactic biases can still be relevant and beneficial, even within the context of models that learn from vast amounts of data.

Beyond syntax. While this work focuses on the importance of more explicit syntactic biases for neural language models, it is not our goal to claim that combining the benefits of data scale and more explicit syntactic biases would be all it takes to achieve human-level natural language understanding in a data-efficient fashion. On the contrary, we argue that syntactic biases are only one of many key ingredients (albeit still an important one, hence our focus on syntactic biases in this work) for building general-purpose language understanding systems. Beyond more explicit syntactic biases, we conjecture that further progress would also hinge on additional innovations that lead to a better modelling of semantics (*i.e.,* the study of meaning) and pragmatics (*i.e.,* the study of how context contributes to meaning, including in multi-agent situations), among others. We expect further advances in these aspects of language to be complementary to the benefits of syntactic biases that we explore in this work, although we leave it to future work to investigate whether this is the case.

1.4 Key Choices and Scope

In reality, the answer to "Can a more explicit modelling of hierarchical syntactic structures improve the performance of neural models within NLP?" is highly dependent on the specific choices and parameters (in a broad sense) of our empirical setup. For instance, a neural architecture that is trained on a dataset of 1 billion words would likely be able to learn a large degree of the syntactic rules directly from data (and hence benefit less from syntactic biases), while a more explicit modelling of syntactic structures is likely to be much more important in the limited data scenario, *e.g.,* when the exact same neural architecture only has access to a much smaller dataset of just 1,000 words. For this reason, it is important to state and justify the key choices and parameters of our empirical setup, as discussed below.

- **What NLP task do we train the models on?** Given the end-to-end nature of neural network training, the extent to which neural models are able to capture syntactic knowledge (and by extension, how much they would benefit from more explicit syntactic biases) would depend on the particular training task. In other words, we expect the model to capture as much (or as little) syntactic information that is necessary to solve the particular task at hand. In this work, we choose to train the neural model on the broader family of **language modelling** objectives, which broadly include both generative, left-to-right language modelling (with GPT-3 as a prominent example), and bidirectional, masked language modelling (with BERT as a prominent example). We motivate this choice through the following three reasons. First, language modelling is a *foundational* task that forms the basis of many downstream NLP applications through language model pre-training (Bommasani et al., 2021). Injecting syntactic inductive biases into the language model training stage thus enables us to evaluate the benefits of syntactic biases on a wide array of NLP tasks that rely on language model pre-training, for instance by simply fine-tuning the pre-trained, syntax-augmented language model on multiple downstream tasks of interest. Second, recent findings have highlighted the importance of data scale for achieving the current state of the art performance in NLP. Unlike the vast majority of NLP tasks, the language modelling task does not require the presence of labelled data, and can thus be conducted on very large corpora of unannotated text. Through our choice to work with the language modelling objective, we are able to assess whether more explicit syntactic biases are still relevant for these neural models that work well at scale, and derive their knowledge from large amounts of data. Third, prior work has conclusively demonstrated the ability of neural language models to acquire a large degree of syntactic information directly from data (Liu et al., 2019a; Tenney et al., 2019a,b; Hewitt and Manning, 2019; Goldberg, 2019; Manning et al., 2020). These findings stand in contrast to the mixed evidence for the emergence of syntactic features from other, non-language-modelling tasks (Williams et al., 2018). Hence, we are interested in understanding whether more explicit syntactic biases are still relevant and beneficial, even as the language modelling objective has by itself led to the emergence of latent syntactic features.

- **How exactly do we inject the syntactic inductive biases?** Furthermore, we expect the benefits of more explicit syntactic biases to depend on the particular way in which the syntactic biases are incorporated into neural language models. In this work, we experiment with two ways of doing so. In Ch. 3, we use existing techniques that design *hierarchical, tree-structured* neural network models, where the model architecture—and by extension, the specific ordering of the neural network operations—is determined by hierarchical syntax trees, as opposed to the sequential positions of the input words. Despite the effectiveness of this approach, we find that tree-structured neural network models are, in practice, very hard to scale to large datasets (Ch. 4). Hence, in the second part of this work (Chs. 4 and 5), we introduce a novel knowledge distillation strategy (Hinton et al., 2015) that introduces syntactic inductive biases in a way that preserves the *scalability* of computationally convenient, non-tree-structured neural architectures.

- **What specific neural architectures do we work with?** As different neural architectures exhibit different inductive biases (Hu et al., 2020), we expect that more explicit syntactic biases may not necessarily benefit different neural architectures to the same extent. In this work, we put a sole emphasis on two widely used neural architectures that have achieved state of the art results within NLP: LSTMs and Transformers. If we find that our proposed way of injecting syntactic inductive biases is beneficial for *both* neural architectures (each with its own, distinctive set of inductive biases), then such findings would provide evidence for the generality and robustness of our proposed approach.

- **How much data are the neural models trained on?** We investigate the benefits of more explicit syntactic inductive biases for neural language models that are trained on three data scales: small ($\sim$3 million words), medium ($\sim$90 million words), and large ($\sim$3 billion words). While the importance of more explicit syntactic biases is *a priori* more obvious for models that are trained on smaller datasets, it remains an open question whether—and to what extent—neural language models that are trained on much larger datasets can still benefit from more explicit syntactic biases, which we aim to answer in this work. Due to limitations on computational resources, we do not experiment with models that are trained on more than 3 billion words, such as RoBERTa (Liu et al., 2019b) and GPT-3.

- **What tasks do we evaluate on?** In Chs. 3 and 4, we evaluate the neural language models (including the proposed model that is augmented with more explicit syntactic biases, and the baseline model that is not) on two challenging benchmarks of syntactic competence: long-range number agreement with multiple attractors (Linzen et al., 2016), and targeted syntactic evaluations (Marvin and Linzen, 2018). Our aim in these two chapters is to answer: Are neural language models already able to learn all they need to know about the syntax of natural language (as measured by their performance on challenging benchmarks of syntactic competence) from data alone, or can these models still benefit from more explicit syntactic biases? In Ch. 5, we move beyond syntactic competence benchmarks, and ask: Can more explicit syntactic inductive biases also be beneficial for other natural language understanding (NLU) benchmarks, including for NLU benchmarks that are not explicitly syntactic in nature? To that end, in Ch. 5, we evaluate the neural language models on a diverse set of 6 structured prediction tasks, including tasks like semantic role labelling and coreference resolution that are not explicitly syntactic in nature, in addition to the widely used GLUE benchmark (Wang et al., 2019), which measures language understanding progress through a diverse set of text classification and regression tasks.

1.5 Methodology and Technical Contribution

Our key method for answering "Can a more explicit modelling of hierarchical syntactic structures lead to better computational models of natural language" is indeed a simple one: Given a baseline model that has *no* explicit syntactic inductive biases, and another model that *does* benefit from more explicit syntactic biases, we compare the performance of the two models on a particular benchmark of interest. In order to isolate the importance of syntactic biases, we carefully control for various confounding, non-syntactic factors that can affect model performance, such as the size of the training data, the number of model parameters, and the extent to which we tune and select the hyper-parameters of each model.

Beneath this simplicity lies an important technical challenge: In practice, it is infeasible to train hierarchical, tree-structured neural network models on large amounts of data—a limitation we attribute to the difficulty of manually batching operations within such models (Ch. 4), which renders them rather ill-suited for taking advantage of specialised deep learning hardwares that aim to accelerate training, such as GPUs and TPUs. Hence, our key *technical contribution* in this work is as follows: We devise a novel knowledge distillation strategy that enables us to combine the best of both worlds, and design neural language models that not only can benefit from stronger notions of syntactic inductive biases, but also can be trained on large amounts of data at the same time. Our approach has four key properties.

- **Effectiveness.** By combining the best of both data scale and syntactic biases, our structure-distilled neural language models largely outperform *both* (i) hierarchical, tree-structured neural language models that benefit from syntactic biases, albeit at the expense of data scale, and (ii) a comparable baseline neural language model that works well at scale, but lacks more explicit syntactic biases. By augmenting $\text{BERT}_{\text{LARGE}}$ with our proposed knowledge distillation approach, we achieve new state of the art single model results (as of mid-2020)—among models pre-trained on the original BERT training set—on 4 structured prediction tasks: in-domain phrase-structure parsing, out-of-domain phrase-structure parsing, dependency parsing, and semantic role labelling (Ch. 5).

- **Efficiency.** Unlike tree-structured neural models, our approach injects syntactic biases through a simple modification of the objective function (*i.e.,* the training loss), without tampering with the structure of neural architectures that work well at scale, such as LSTMs and Transformers. For this reason, our structure-distilled neural models can be trained on much larger datasets, and are at least five times faster to train than a comparable tree-structured model—all the while retaining a stronger notion of syntactic inductive biases, as evidenced by their strong performance on benchmarks of syntactic competence.
- **Compatibility.** In practice, our proposed knowledge distillation approach can be used to inject syntactic biases in a way that is compatible with *any* existing neural language modelling architectures (all through a simple addition to the training loss). Our approach thus maintains complete compatibility with architectures like LSTMs and Transformers that are *already* widely used in practical systems, without necessitating any architectural changes, hence lowering the barrier for widespread adoption of our technique.
- **Generality and robustness.** Our proposed knowledge distillation approach improves model performance under two different neural language modelling architectures (LSTMs and Transformers), and also under two different language modelling objectives (unidirectional, left-to-right language modelling and masked language modelling). These findings thus provide evidence for the generality and robustness of our proposed approach.

1.6 Thesis Outline

The rest of this thesis is structured as follows.

- In Chapter 2, we provide an overview of the language modelling task, with an emphasis on syntactic language modelling and the recurrent neural network grammars (Dyer et al., 2016, RNNGs) model. Concretely, RNNGs are a class of hierarchical, tree-structured neural syntactic language models, which underpin the key technical aspect of this work.

- In Chapter 3, we revisit the question asked by Linzen et al. (2016): To what extent can LSTM neural language models succeed at a challenging number agreement benchmark of syntactic competence, even when they are not provided with explicit syntactic supervision or inductive biases to do so? Contrary to prior findings, we find that, despite their lack of explicit syntactic biases, LSTMs are seemingly able to succeed at this task, although we demonstrate through adversarial examples that their success can be partially attributed to superficial sequential (as opposed to syntactic) cues. We show that RNNGs, with their explicit modelling of syntactic structures, can perform *even better* at this task, and remain far more robust to adversarial perturbations. We then explore the hypothesis that *how* the syntactic structure is built affects the RNNGs' syntactic generalisation, and find the top-down traversal to achieve the best performance at complex number agreement cases.
- In Chapter 4, we find that, despite their advantages in terms of perplexity and syntactic competence on small amounts of data ($\sim$3 million tokens), in practice RNNGs are much more difficult to scale than sequential models like LSTMs, which means that it is infeasible to train them on larger amounts of data. Hence, given the strong importance of data scale for recent language modelling success, we devise a novel knowledge distillation approach that combines the best of both worlds, and design a distilled syntax-aware (DSA) LSTM that not only can be trained on larger amounts of data ($\sim$90 million tokens), but also incorporate stronger notions of syntactic inductive biases at the same time. We then demonstrate that the resulting DSA-LSTM model performs well on a challenging and diverse benchmark of syntactic competence (Marvin and Linzen, 2018). These findings thus demonstrate the continued relevance of syntactic inductive biases, even for powerful LSTM language models that derive their knowledge from medium-sized corpora.
- In Chapter 5, we investigate whether a similar form of more explicit syntactic biases can still be beneficial for Transformer-based BERT masked language models, even as these models are able to learn from very large amounts of data, and have been shown to achieve excellent performance at various downstream tasks. To that end, we apply a similar knowledge distillation technique as we did in Ch. 4, albeit with one key difference: As BERT estimates the probability of masked words in bidirectional context, we distill the RNNG teacher's marginal distribution over words in context, for which we develop an efficient yet effective approximation. Overall, our approach reduces relative error by 2-21% on a diverse set of syntactic and non-syntactic structured prediction tasks, although

we obtain mixed results on GLUE. These findings reaffirm the benefits of syntactic biases, even for Transformer models that learn from large textual corpora ($\sim$3 billion words), and contribute to a better understanding of where syntactic biases are helpful in NLP.

- In Chapter 6, we briefly summarise our key findings, reflect on their implications, and remark on several promising directions for future work.

1.7 Related Publications

Parts of this thesis are based on the following published work.

- **Chapter 2**:
  - Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. "Recurrent Neural Network Grammars," In Proceedings of NAACL (Volume 1: Long Papers), 2016. Oral presentation.
  - Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. "What Do Recurrent Neural Network Grammars Learn About Syntax?" In Proceedings of EACL (Volume 1: Long Papers), 2017. Oral presentation, **Outstanding Long Paper award**.
- **Chapter 3**:
  - Adhiguna Kuncoro, Chris Dyer, John T. Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. "LSTMs can Learn Syntax-Sensitive Dependencies Well, but Modeling Structure Makes Them Better," In Proceedings of ACL (Volume 1: Long Papers), 2018. Oral presentation.
  - Adhiguna Kuncoro, Chris Dyer, John T. Hale, and Phil Blunsom. "The Perils of Natural Behavioral Tests for Unnatural Models: The Case of Number Agreement," Poster presented at the Learning Language in Humans and in Machines (L2HM) conference, 2018.
- **Chapter 4**:

- Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. "Scalable Syntax-Aware Language Models Using Knowledge Distillation," In Proceedings of ACL (Volume 1: Long Papers), 2019. Oral presentation, **Nomination for Best Long Paper award**.

- **Chapter 5**:

    - Adhiguna Kuncoro*, Lingpeng Kong*, Daniel Fried*, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. "Syntactic Structure Distillation Pre-training for Bidirectional Encoders," In Transactions of the Association for Computational Linguistics (TACL), 2020. Presented at EMNLP 2020, Oral presentation.

*: equal contribution.

Chapter 2

Background: Recurrent Neural Network Grammars

Chapter Abstract

In this chapter, we provide a summary of the relevant background material that will be used throughout the rest of this thesis. To that end, the crux of this chapter is devoted to describing recurrent neural network grammars (Dyer et al., 2016, RNNGs). RNNGs denote a class of neural syntactic language model that explicitly models hierarchical syntactic structures, where larger linguistic units are constructed from smaller ones through recursive compositions. Due to their strong syntactic biases, RNNGs underpin the key technical aspect of this work: We directly work with RNNGs in Ch. 3, and evaluate their syntactic competence in comparison to other neural language models that lack explicit modelling of syntactic structures. RNNGs also play a key role in Chs. 4 and 5, where we use them as a means of injecting syntactic biases into other, non-hierarchical neural language modelling architectures that can nevertheless work much better at scale, and are thus able to derive their knowledge from larger amounts of data.

2.1 Notation

We begin by outlining the mathematical notations that are used throughout this work. We use $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ to denote a contiguous sequence of T discrete tokens, where x_t

The material in this chapter is adapted from Dyer, Kuncoro et al. (2016) and Kuncoro et al. (2017).

denotes the t-th token of the sequence $\mathbf{x}$. Throughout this work, we mostly consider the case where $\mathbf{x}$ is a single natural language sentence. This choice is motivated by the fact that syntactic structures are typically defined at the sentence level, where syntax defines the ways in which words combine to form phrases, clauses, and sentences (Ch. 1). Nevertheless, within the context of the BERT model in Ch. 5, we consider the case where $\mathbf{x}$ denotes a longer sequence of 128 or 512 words, which in practice often spans multiple sentences.

Throughout this work, we consider both cases where each token x_t denotes a standalone word (*e.g.,* "*dog*", as we do in Chs. 3 and 4), and also where each token x_t is a subword unit (Sennrich et al., 2016), as we do in Ch. 5. In the latter case, each word can be split into one or more subword units—for instance, by splitting the word "*dog*" into two separate tokens "*d*" and "*##og*", where tokens that are prefixed by "*##*" are subword units.[1] We denote the finite set of all possible word types (or alternatively, the finite set of all possible subword symbols) as the vocabulary Σ, where each word x_t in the utterance $\mathbf{x}$ must be part of the vocabulary Σ. We represent each word or subword symbol interchangeably with its corresponding index in the vocabulary Σ: For instance, the word "*dog*" is assigned an index of 1, while the word "*cat*" is assigned an index of 2, and so on; by definition, the maximum word index corresponds to $|\Sigma|$. As each utterance $\mathbf{x}$ can consist of an arbitrary number of words, and thus may have an arbitrary length $|\mathbf{x}|$, $\Sigma^\star$ denotes the set of all possible utterances in natural language. Furthermore, let $D = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(K)}\right]$ be a dataset that contains a collection of $K = |D|$ utterances, where $\mathbf{x}^{(k)} \in \Sigma^\star$ denotes the k-th utterance in the dataset D. When there is no need to distinguish a particular utterance $\mathbf{x}^{(k)}$ from all the other utterances in the dataset, we may omit the superscript and simply refer to a generic utterance as $\mathbf{x}$.

We use $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ to denote the *joint* probability of generating a particular utterance $\mathbf{x}$ and a corresponding hierarchical syntax tree $\mathbf{y}$, under the set of model parameters $\boldsymbol{\theta}$.[2] The *marginal* probability of generating an utterance $\mathbf{x}$ is obtained by summing over all possible syntax trees that are compatible with $\mathbf{x}$, denoted as $Y(\mathbf{x})$; *i.e.,* $p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{y} \in Y(\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. As the number of all possible syntax trees is exponential in the length of the utterance $\mathbf{x}$, exact marginalisation is often intractable for models that do not make strong independence

[1] By working with subwords as opposed to word units, the model is able to generate new word types that it had not seen in the training set. For example, in the case where "*dogs*" had not been observed in the training set, this novel word can be generated by decomposing its prediction into "*d*", "*##og*", and "*##s*".

[2] Our notation here overloads the random variable and its particular realisation. Concretely, let X be a discrete random variable, where $\Sigma^\star$ (*i.e.,* the set of all possible utterances in natural language) is its sample space. $p_{\boldsymbol{\theta}}(X = \mathbf{x})$ denotes the probability of that random variable taking a particular realisation (here the utterance $\mathbf{x}$) under the set of model parameters $\boldsymbol{\theta}$. Throughout this work, we simplify this notation as $p_{\boldsymbol{\theta}}(\mathbf{x})$.

assumptions (as is the case in this work). Throughout this work, we focus on the case where the syntax trees are defined by phrase-structure grammar (*i.e.,* $\mathbf{y}$ denotes a **phrase-structure tree** for a sentence $\mathbf{x}$), and leave the extension to other grammar formalisms to future work.

We use bold letters to refer to vectors or sequences; hence $\mathbf{x}$ denotes an utterance, which consists of a sequence of T words, while x_t denotes a single word. Following standard convention, we use lowercase letters (*e.g.,* $\mathbf{v}$) to denote vectors, and uppercase letters (*e.g.,* $\mathbf{W}$) to denote matrices. We use semicolons to denote vector concatenations, *e.g.,* $[\mathbf{u}; \mathbf{v}]$.

2.2 Neural Networks

Having described the mathematical notations, we briefly introduce artificial neural networks, which lay the foundation of the machine learning models that we use throughout this work. In its simplest form, a neural network defines a mapping from an input vector $\mathbf{e} \in \mathbb{R}^d$ into a hidden layer representation $\mathbf{h} \in \mathbb{R}^n$, which is computed using the model parameters $\boldsymbol{\theta}$. In the case of **feedforward** neural networks—alternatively known as the multi-layer perceptrons (MLPs)—a single layer of feedforward neural network operation is defined as:

$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}), \tag{2.1}$$

where $f(\cdot)$ denotes an element-wise non-linearity function, such as $tanh$, $sigmoid$, or $ReLU$; in this case, the set of model parameters $\boldsymbol{\theta}$ encompasses the weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$ and the bias vector $\mathbf{b} \in \mathbb{R}^n$, both of which are learnt from data through backpropagation (*i.e.,* $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$). This feedforward network formulation can be easily extended to the case where the network has multiple layers of non-linear transformations (hence the "deep" epithet), where the hidden layer representation at layer $l \in \{1, \cdots, L\}$, denoted as $\mathbf{h}^{(l)}$, is computed based on the hidden layer representation at layer $l - 1$, denoted as $\mathbf{h}^{(l-1)}$:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\,\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}),$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ denote the layer-specific weight matrix and bias vector, and $\mathbf{h}^0 = \mathbf{e}$ (*i.e.,* the first hidden layer operates on the input vector $\mathbf{e}$); hence $\boldsymbol{\theta} = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \cdots, \mathbf{W}^{(L)}, \mathbf{b}^{(L)}\}$. In practice, this formulation is often augmented with other techniques that have been shown to stabilise deep neural network training and improve model performance, such as residual connections (He et al., 2016) and layer normalisation (Ba et al., 2016), among others.

Sequence modelling with neural networks. In practice, natural language textual data has a sequential overt form (although its latent structure is hierarchical rather than sequential, Ch. 1), where an utterance $\mathbf{x}$ can be decomposed into a sequence of T tokens $[x_1, x_2, \cdots, x_T]$. Hence, throughout this thesis, we work primarily with **neural sequence models**, which aim to map a sequence of input vectors $\mathbf{e}_{1:T} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_T]$ into a sequence of hidden layer vectors, denoted as $\mathbf{h}_{1:T} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T]$. An example of such model is the "vanilla" Elman recurrent neural networks (Elman, 1990, RNNs), which compute the mapping as:

$$\mathbf{h_t} = \sigma\left(\mathbf{W}\mathbf{e}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}\right), \tag{2.2}$$

where $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{U}, \mathbf{b}\}$ and $\mathbf{U} \in \mathbb{R}^{n \times n}$; here σ denotes the sigmoid activation function. In other words, under the Elman RNN, the current hidden layer $\mathbf{h_t}$ is computed based on both the input at the current time step $\mathbf{e}_t$, and the previous hidden layer $\mathbf{h}_{t-1}$. In practice, the Elman RNN is difficult to train due to the vanishing and exploding gradient problem associated with very deep neural networks, including RNN-based architectures where the processing of each additional input adds to the depth of the network (Hochreiter, 1998; Pascanu et al., 2013). In order to circumvent this problem and improve the modelling of long-range dependencies, prior work has proposed alternative RNN-based architectures that are augmented with certain gating mechanisms, such as the long short-term memory networks (Hochreiter and Schmidhuber, 1997, LSTMs) and gated recurrent units (Cho et al., 2014, GRUs). In addition to RNNs and their variants, another commonly used neural sequence modelling architecture is the Transformer model (Vaswani et al., 2017), which constructs its hidden layer representations through multiple layers of self-attention operations (Bahdanau et al., 2015) that are interleaved with feedforward ones.[3] Throughout this thesis, we work with both LSTM and Transformer neural sequence models: Concretely, we use LSTMs as the backbone of the RNNG model described in this chapter, and also in the language modelling experiments in Chs. 3 and 4, whereas we use the Transformer model within the context of the BERT model in Ch. 5. We refer interested readers to each respective work (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017) for more details on these models.

[3]RNN processing is necessarily sequential in nature: Concretely, the representation for the current hidden state $\mathbf{h}_t$ can only be computed once all its preceding hidden states $[\mathbf{h}_1, \cdots, \mathbf{h}_{t-1}]$ had been computed based on their respective input. In contrast, within a single Transformer layer, the computation of the hidden state $\mathbf{h}_t$ is done *independently* of all other hidden states, which lends itself well to greater parallelism at training time (Ch. 4), resulting in a substantial speed-up under specialised hardwares like graphics processing units (GPUs).

Input and output layers. As neural network models operate over vector representations of data, the first step in neural sequence modelling within NLP is to convert each *discrete* token $x_t \in \mathbf{x}$ into its corresponding vector representation $\mathbf{e}_t$. Formally, let $\mathbf{E} \in \mathbb{R}^{|\Sigma| \times d}$ be an embedding lookup matrix that converts each word type—or alternatively, each subword symbol, depending on the granularity of the input token representation—$w \in \Sigma$ into its corresponding vector representation $\mathbf{e}_t$.[4] The input vector $\mathbf{e}_t$ for the token x_t is defined as:

$$\mathbf{e}_t = \mathbf{E}\left[x_t\right], \tag{2.3}$$

where $\mathbf{E}\left[x_t\right]$ is the embedding vector corresponding to the x_t-th row of the embedding matrix $\mathbf{E}$;[5] *i.e.,* the input layer "selects" a row of $\mathbf{E}$ that corresponds to the word x_t.

Having described the input layer of neural sequence models within NLP, we briefly describe the output layer of such models. In most NLP use cases—including the language modelling task described in §2.3, and other language generation problems—the output layer maps the current hidden state $\mathbf{h}_t$ into a categorical distribution that defines the probability of generating each of the O potential output labels.[6] This process is done in two steps, where the first step is to map the current hidden state $\mathbf{h}_t$ into the *logits* output vector $\mathbf{o}_t$, as follows:

$$\mathbf{o}_t = \mathbf{R}\mathbf{h}_t + \mathbf{b}_o, \tag{2.4}$$

where $\mathbf{R} \in \mathbb{R}^{O \times n}$ and $\mathbf{b}_o \in \mathbb{R}^{O}$ denote the parameters of the output layer. A **softmax** activation function is then applied to the logits in order to derive a categorical distribution over all output labels, where the probability of predicting a particular output label $\hat{y}_t = k$ is:

$$p_{\boldsymbol{\theta}}(\hat{y}_t = k \mid \mathbf{h}_t) = \frac{\exp(o_t[k])}{\sum_{j=1}^{O} \exp(o_t[j])}, \tag{2.5}$$

where $\exp(\cdot)$ is the exponential function and $o_t[k]$ is the k-th value of the logits vector $\mathbf{o}_t$.

2.3 Language Modelling

The language modelling task is a density estimation problem, where a statistical model (in the case of this work, deep neural sequence models) is used to estimate the probability of an

[4]In practice, the embedding matrix $\mathbf{E}$ is typically a learnable parameter, although it can be initialised through pre-trained word embedding vectors that are learnt from large corpora (*e.g.,* Mikolov et al., 2013).

[5]Recall that the word token x_t is interchangeable with its corresponding index in the vocabulary Σ, §2.1.

[6]Most natural language generation problems can be decomposed into predicting the next word x_{t+1}, conditional on the words that had been generated thus far $[x_1, \cdots, x_t]$ and some conditioning context (*e.g.,* a document that needs to be summarised). In this case, the space of all possible labels for the next word is all the words in the vocabulary Σ; hence $O = |\Sigma|$, where O denotes the number of all possible output labels.

utterance $\mathbf{x}$, henceforth denoted as $p_{\boldsymbol{\theta}}(\mathbf{x})$, which can be decomposed as follows:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t \mid \mathbf{x}_{<t}), \qquad \text{(chain rule decomposition)} \tag{2.6}$$

where $\mathbf{x}_{<t}$ denotes the tokens in $\mathbf{x}$ that appear before x_t, *i.e.,* $[x_1, \cdots, x_{t-1}]$. In practice, the last token x_T is typically reserved for a special "end-of-sequence" token—the importance of which was emphasised by NLP researchers in the late 1990s (*e.g.,* Collins, 1999)—in order to form a valid probability distribution over all possible utterances in natural language Σ^*.

The language modelling objective in Eq. 2.6 decomposes the estimation of $p_{\boldsymbol{\theta}}(\mathbf{x})$ in a left-to-right fashion, where the language model (with model parameters $\boldsymbol{\theta}$) estimates the probability of each token x_t, conditional on its left context $\mathbf{x}_{<t}$. While the vast majority of prior language modelling work employed such left-to-right decomposition, we remark that other decomposition strategies, such as a right-to-left decomposition, constitute equally valid alternative formulations for the language modelling task. Nevertheless, the left-to-right decomposition, as outlined in Eq. 2.6, has an additional benefit from the cognitive science perspective, where such left-to-right language models constitute a more faithful computational model of human language processing, which occurs mostly in a left-to-right, **incremental** fashion (Marslen-Wilson, 1973; Ferreira and Clifton, 1986), with the exception of languages like Arabic and Hebrew that are read and written in a right-to-left fashion.

Neural language modelling. Having described the language modelling task, we provide an overview of how neural networks are used for language modelling; throughout this work, we refer to the broader family of such models as **neural language models**. To that end, existing approaches largely fall into two categories: (i) feedforward neural language models (Bengio et al., 2003; Mnih and Hinton, 2007), and (ii) recurrent neural language models (Mikolov et al., 2010; Sundermeyer et al., 2012); we now describe both models in turn.

As shown in Eq. 2.1, feedforward neural networks operate over a fixed-size input vector $\mathbf{e}$. Given the variable-length nature of each utterance $\mathbf{x} \in D$, a common technique in feedforward neural language modelling is to apply a Markov assumption, such that the prediction of the word x_t only depends on the preceding $n-1$ words—where n is fixed across all instances—as opposed to the whole left context $\mathbf{x}_{<t}$ that is variable in length

(Eq. 2.6).[7] Such Markov assumption is especially useful in the case where the utterance $\mathbf{x}$ refers to a long sequence of words that spans multiple sentences, such as a document or a book. In such cases, conditioning on the entire left context would be prohibitively expensive (*i.e.,* $|\mathbf{x}_{<t}| \gg n$). Formally, the Markov assumption is defined as follows:

$$p_{\boldsymbol{\theta}}(x_t \,|\, x_1, \cdots, x_{t-1}) \approx p_{\boldsymbol{\theta}}(x_t \,|\, x_{t-n+1}, \cdots, x_{t-1}). \qquad \text{(Markov assumption)} \qquad (2.7)$$

Applying the Markov assumption to the "vanilla" feedforward neural language models, the probability of predicting the next word $p_{\boldsymbol{\theta}}(x_t \,|\, \mathbf{x}_{<t})$ is defined as follows. Let $\mathbf{e}_{(t-n+1):(t-1)} = [\mathbf{e}_{t-n+1}, \cdots, \mathbf{e}_{t-1}]$ be the word embedding vectors for the input tokens $[x_{t-n+1}, \cdots, x_{t-1}]$, as obtained through Eq. 2.3. The hidden state $\mathbf{h}_{t-1}$ is defined as follows:

$$\mathbf{h}_{t-1} = f(\sum_{j=1}^{n-1} \mathbf{W}_j \mathbf{e}_{t-j} + \mathbf{b_j}),$$

where $\mathbf{W}_j \in \mathbb{R}^{n \times d}$ and $\mathbf{b}_j \in \mathbb{R}^n$ denote the weight matrix and bias term that are specific to the j-th word prior to x_t, and $f(\cdot)$ denotes an element-wise non-linearity. Following Eqs. 2.4 and 2.5, the probability of generating x_t as the next word is thus defined as follows:

$$\begin{aligned} \mathbf{o}_{t-1} &= \mathbf{R}\mathbf{h}_{t-1} + \mathbf{b}_o, && \text{(Eq. 2.4, reproduced)} \\ p_{\boldsymbol{\theta}}(x_t \,|\, x_1, \cdots, x_{t-1}) &\approx p_{\boldsymbol{\theta}}(x_t \,|\, x_{t-n+1}, \cdots, x_{t-1}) && \text{(Eq. 2.7, reproduced)} \\ &= \frac{\exp(o_{t-1}[x_t])}{\sum_{w \in \Sigma} \exp(o_{t-1}[w])}. && \text{(softmax, Eq. 2.5)} \end{aligned}$$

In some sense, other types of neural language models, such as those based on Transformers (Vaswani et al., 2017) and convolutional networks (Dauphin et al., 2017), can be understood as more sophisticated variants of feedforward neural language models. Concretely, these models make a similar Markov assumption as in Eq. 2.7 (albeit typically with large values of n), and predict the next word x_t conditional on a finite context window $[x_{t-n+1}, \cdots, x_{t-1}]$, which is summarised by the hidden state vector $\mathbf{h}_{t-1}$; a key difference is that these models leverage additional techniques, such as self-attention and convolution, to construct $\mathbf{h}_{t-1}$.

Unlike feedforward neural networks, recurrent neural networks have a theoretical ability to propagate information for an unbounded amount of time, which is achieved through

[7] An alternative feedforward neural language modelling approach—which circumvents the need to apply the Markov assumption—is to set n to be the longest sequence length in the training dataset. This means that, for every utterance $\mathbf{x} \in D$, the prediction of every token x_t would take into account all its preceding words. Under this approach, the length of most $\mathbf{x}_{<t}$ would be smaller than n; in such cases, special padding symbols are used to pad the conditioning context $\mathbf{x}_{<t}$ to length n. Nevertheless, this approach can be computationally expensive when $\mathbf{x}$ corresponds to a long sequence (such as documents), as opposed to a single sentence.

the recurrent connections (Eq. 2.2). This property of RNNs lends itself well to sequence modelling tasks like language modelling, and—at least in theory—enables the model to predict the next token x_t conditional on the *entire* left context $\mathbf{x}_{<t}$, without the need to rely on any Markov assumptions. Nevertheless, the ability of RNNs to capture long-range dependencies and propagate information for a large number of steps is more limited in practice, even for variants of RNNs that are augmented with certain gating mechanisms and memory cells (Le et al., 2015; Arjovsky et al., 2016). Using the same notation as in the case of the feedforward neural language models, the application of vanilla Elman RNNs for the language modelling task can be derived from Eq. 2.2 in a straightforward fashion:

$$\mathbf{h}_{t-1} = \sigma\left(\mathbf{W}\mathbf{e}_{t-1} + \mathbf{U}\mathbf{h}_{t-2} + \mathbf{b}\right), \tag{2.8}$$

where $\mathbf{e}_0$ (*i.e.,* the input used to construct $\mathbf{h}_0$, based on which the model predicts x_1) is set to a special "start-of-sequence" symbol, whereas the very first hidden state $\mathbf{h}_{-1}$—which is used to construct $\mathbf{h}_0$ alongside $\mathbf{e}_0$—can be initialised to a vector of zeros, or cast as learnt model parameters. The output distribution $p_{\boldsymbol{\theta}}(x_t \mid \mathbf{x}_{<t})$ is computed based on $\mathbf{h}_{t-1}$ in a similar fashion as in the feedforward neural language modelling case (albeit without requiring any approximation in the form of a Markov assumption), which we do not repeat for brevity.

Learning. Regardless of their specific model architecture and parameterisation—which specify how the model constructs the hidden states $\mathbf{h}_{0:T}$ based on the input $\mathbf{e}_{0:T}$—neural language models typically aim to minimise the negative log likelihood of the observed data. Formally, the aim of the training process is to find the set of model parameters $\hat{\boldsymbol{\theta}}$, such that:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} -\sum_{\mathbf{x}\in D} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \\
&= \arg\min_{\boldsymbol{\theta}} -\sum_{\mathbf{x}\in D} \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(x_t \mid \mathbf{x}_{<t}). \qquad \text{(chain rule decomposition)}
\end{aligned}$$

The objective function above is differentiable in an end-to-end fashion, and can thus be optimised through standard (typically mini-batch) gradient descent, optionally with the addition of certain optimisation algorithms that dynamically adapt the learning rate based on historical gradient values, such as Adam (Kingma and Ba, 2015). Throughout this work, we explicitly state the optimiser and learning rate at each relevant experimental section.

2.4 Recurrent Neural Network Grammars

In this section, we describe recurrent neural network grammars (Dyer et al., 2016, RNNGs), which underpin the key technical aspect of this work. Concretely, RNNGs combine two modelling traditions: (i) recurrent neural network language models (§2.3), which have the ability to automatically learn how to craft informative vector representations of an unbounded history of words (Mikolov et al., 2010; Zaremba et al., 2015); and (ii) syntactic language modelling, which aims to generate a sequence of words jointly with its corresponding syntax tree (Baker, 1979; Jelinek and Lafferty, 1991). As a type of neural syntactic language model, RNNGs estimate the *joint* probability of surface strings $\mathbf{x}$ and phrase-structure trees $\mathbf{y}$, which are computed based on the set of neural model parameters $\boldsymbol{\theta}$; we henceforth denote this probability as $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. To that end, RNNGs make use of recurrent neural network (specifically, LSTM) architectures to derive their predictions, although they differ from conventional recurrent neural network language models in two important ways:

- Whereas RNN-based language models only model the probability of surface strings $p_{\boldsymbol{\theta}}(\mathbf{x})$, RNNGs model the joint probability of surface strings and phrase-structure trees $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, which carries two notable benefits. The first benefit comes in the form of interpretability and transparency: Given an input sentence $\mathbf{x}$ whose underlying syntactic structure is unknown, we can use RNNGs to derive an estimate of $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, and obtain a list J most likely syntactic interpretations $\hat{\mathbf{y}}^{(1)}, \cdots, \hat{\mathbf{y}}^{(J)}$ and their probabilities. This property means that we are able to better inspect and understand the model's predictions (*e.g.,* given a sentence "*Bob saw the man with a telescope.*" that has at least two possible syntactic interpretations,[8] which interpretation is more likely according to the model? And what are some other highly probable syntactic interpretations of this sentence?), which in turn can shed more light on their success and failure modes. In the case where we are agnostic to the syntactic interpretation of $\mathbf{x}$, we can obtain the RNNGs' estimate of $p_{\boldsymbol{\theta}}(\mathbf{x})$ by marginalising over the set of all phrase-structure trees $Y(\mathbf{x})$ that are compatible with $\mathbf{x}$. A second benefit comes in the form of controllability: In theory, we can use RNNGs to generate sentences that conform to certain pre-defined syntactic properties.[9]

[8] The first syntactic interpretation of "*Bob saw the man with a telescope*" is that Bob used the telescope as a visual aid device to see the man; the second interpretation is that Bob saw a man who had a telescope with him.

[9] In practice, taking advantage of RNNGs' joint modelling of surface strings and syntactic tree structures—*e.g.,* disambiguating the parse tree $\mathbf{y}$ for a sentence $\mathbf{x}$, or sampling a sentence $\mathbf{x}$ that conforms to a certain pre-defined syntactic structure $\mathbf{y}$—necessitates efficient inference procedures, which are non-trivial to design

- Whereas RNN language models operate in a sequential fashion (*i.e.,* they process an input sentence $\mathbf{x}$ as a linear/flat sequence of symbols, without explicitly taking into account its latent hierarchical syntactic structures), RNNGs differ in that they operate in an explicitly *hierarchical* and *compositional* fashion. This is achieved through a recursive composition function, where smaller units like words are recursively combined to form larger ones, such as phrases, clauses, and eventually sentences. To that end, the ordering of these composition operations is determined by the hierarchical syntax tree (in the case of this work, the phrase-structure tree, §2.4.1), which is assumed to be available at training time; at test time, the model must generate the corresponding syntax tree jointly with the surface string. Indeed, the explicit modelling of hierarchical syntactic structures within RNNGs corresponds well to systematic compositionality: We expect someone who already knows the meaning of individual words, such as "*blue*" and "*table*", to be able to produce and understand utterances that involve novel combinations of these words, such as "*a blue table*" and "*a nice blue table*", even if she had never encountered these exact combinations in the past. Throughout this work, we conjecture that an inductive bias for modelling syntactic structures is beneficial for improving performance at various benchmarks of syntactic competence (Chs. 3 and 4), in addition to improving performance and data efficiency at other syntax-sensitive benchmarks of natural language understanding (Ch. 5), even in the case of expressive neural language models that work well at scale.

2.4.1 Tree Traversal

Having provided a high-level overview of RNNGs and how they differ from conventional RNN-based language models, we provide a brief description of how RNNGs decompose the joint prediction of surface strings and their corresponding phrase-structure trees $(\mathbf{x}, \mathbf{y})$.

Phrase-structure tree. As a syntactic language model, RNNGs model the joint probability of strings $\mathbf{x}$ and phrase-structure trees $\mathbf{y}$. In Fig. 2.1, we provide a running example that represents a five-word sentence "*The hungry cat meows .*" (denoted in blue) and its phrase-structure tree (denoted in red). As shown in Fig. 2.1, the phrase-structure tree consists of three phrases, where each phrase is represented by a non-terminal symbol (shown in red) that spans one or more *contiguous* children nodes. Due to the recursive nature of the

for neural models like RNNGs that make no conditional independence assumptions.

grammar, each phrase can be nested within another phrase; this property means that the children nodes of each phrase can consist of both terminal (*i.e.,* words, shown in blue) and other non-terminal nodes (*i.e.,* another phrase, shown in red), or any combination thereof. Concretely, the phrase-structure tree in Fig. 2.1 consists of the following three phrases:[10]

- "(NP *The hungry cat*)", where "NP" is a noun phrase non-terminal node, where the words (from left to right): (i) "*The*", (ii) "*hungry*", and (iii) "*cat*" are its three children nodes;
- "(VP *meows*)", where "VP" denotes a verb phrase non-terminal node, where the word "*meows*"—which happens to be the main verb of the sentence—is its only child node; and
- "(S (NP *The hungry cat*) (VP *meows*) .)", where "S" denotes a declarative clause non-terminal node. This phrase has three children nodes, which are (from left to right): (i) the phrase "(NP *The hungry cat*)", (ii) the phrase "(VP *meows*)", and lastly (iii) the word ".". This phrase is the topmost non-terminal node in the tree, which spans the whole sentence.

We henceforth use the same bracketing notation—in the format of "([non-terminal symbol] [child node 1], ..., [child node n])"—to denote either a full phrase-structure tree (*e.g.,* "(S (NP *The hungry cat*) (VP *meows*) .)"), or one of its phrases (*e.g.,* "(NP *The hungry cat*)").

Top-down traversal. As shown in Eq. 2.6, the language modelling task typically decomposes the production of the string $\mathbf{x}$ into a series of local decisions, each of which predicts x_t conditional on its left context $\mathbf{x}_{<t}$. In the case of syntactic language models like RNNGs, we need to define a decomposition that generates the surface string $\mathbf{x}$ jointly with its phrase-structure tree $\mathbf{y}$. To that end, RNNGs employ a **top-down, left-to-right** traversal of $(\mathbf{x}, \mathbf{y})$, in a process that is reminiscent of a depth-first search. In the case of our running example, we demonstrate the traversal order in Fig. 2.1, where the numbers near each node represent the order in which that particular node is visited during a top-down, left-to-right traversal of the tree. We remark that each non-terminal node is visited twice during the course of generation: Each non-terminal node is visited once when it is first constructed (*e.g.,* step 2, Fig. 2.1, which "opens" a new noun phrase); it is then visited once more when the phrase is complete (*e.g.,* step 6, Fig. 2.1, which "closes" the noun phrase after all its children had been generated). Such top-down generation order is the standard for RNNGs, although we explore two extensions to left-corner and bottom-up generation orders in Ch. 3.

[10]In practice, existing phrase-structure tree annotations include additional part-of-speech (POS) tag information. As RNNGs do not model such POS tags as part of their generation strategy (*i.e.,* they only model the word terminals and the phrasal non-terminal symbols), we omit all POS tag annotations for brevity.

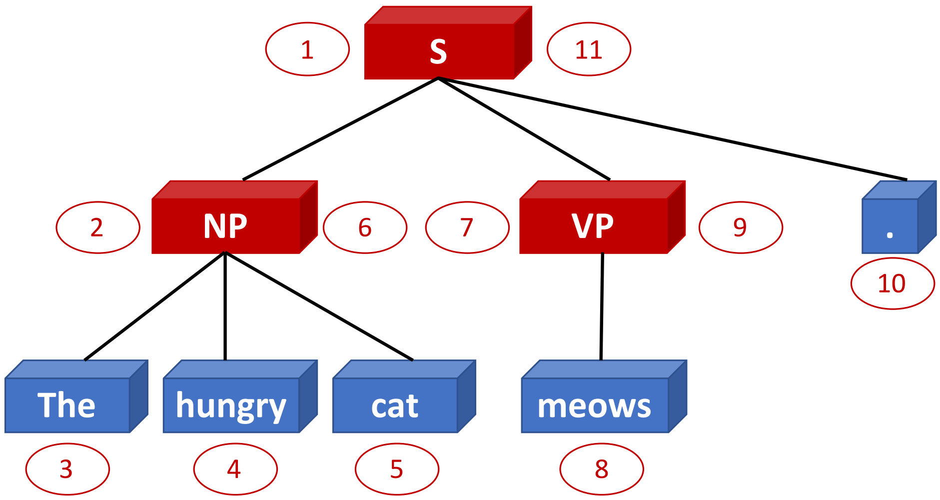


Figure 2.1: Example of a simple sentence "*The hungry cat meows .*" (denoted in blue) and its corresponding phrase-structure tree (denoted in red). Here "NP", "VP", and "S" denote the non-terminal symbols in the tree $\mathbf{y}$ that correspond to a noun phrase, a verb phrase, and a declarative clause, respectively. The numbers near each node represent the order in which that particular node is visited during a top-down, left-to-right traversal of $(\mathbf{x}, \mathbf{y})$.

Action sequences. During the course of the top-down, left-to-right generation of the phrase-structure tree, RNNGs employ the following three types of actions:

- $\text{NT}(n)$, which opens a new non-terminal symbol $n \in N$, where n denotes a non-terminal symbol and N represents the finite set of all non-terminal symbols (*e.g.,* in the case of our running example in Fig. 2.1, $N = \{\text{NP}, \text{VP}, \text{S}\}$). Here we assume that $N \cap \Sigma = \emptyset$; in other words, we assume that the set of all non-terminals N and the set of all words in the vocabulary Σ are disjoint. In the case of our running example in Fig. 2.1, step 2 would correspond to the action "$\text{NT}(\text{NP})$", while step 7 would correspond to "$\text{NT}(\text{VP})$".
- $\text{GEN}(w)$, which generates a word terminal symbol $w \in \Sigma$. In the case of Fig. 2.1, step 3 would correspond to "$\text{GEN}(\text{The})$", while step 8 would correspond to "$\text{GEN}(\text{meows})$".
- REDUCE, which closes the most recent open/incomplete non-terminal symbol once it is complete (*i.e.,* all its corresponding children nodes had been generated). Note that the REDUCE action does not need to specify which non-terminal symbol to close, as it always closes the most recent open non-terminal symbol within the context. In the case of our running example in Fig. 2.1, the REDUCE action in step 6 would close the open "NP" non-terminal symbol, because it is the most recent open/incomplete non-terminal symbol at that point in the generation process. In contrast, step 11 would close the open

"S" non-terminal symbol, because, at that point in the generation process, all the other non-terminal nodes that had been generated thus far—"(NP *The hungry cat*)" and "(VP *meows*)"—had been closed off through REDUCE actions in steps 6 and 9, respectively.

Having defined the three types of RNNGs' actions, we decompose the prediction of the surface string $\mathbf{x}$ and the phrase-structure tree $\mathbf{y}$ into an action sequence $\mathbf{a}$. In Fig. 2.3 (rightmost column), we provide an example of the action sequence $\mathbf{a}$ that would generate our running example in Fig. 2.1. Formally, let $\mathbf{S}_t$ be the RNNGs' model state (as described in §2.4.2) at time step t, while $\mathbf{v}_t^S \in \mathbb{R}^n$ denotes the corresponding hidden state vector encoding of the model state $\mathbf{S}_t$. Furthermore, let $a_t \in \{\text{NT}(n), \text{GEN}(w), \text{REDUCE}\}$ be the next action that the model should take at time step t. RNNGs decompose the joint prediction of surface strings and their corresponding phrase-structure trees $(\mathbf{x}, \mathbf{y})$ as follows:[11]

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{|\mathbf{a}|} p_{\boldsymbol{\theta}}(a_t \mid \mathbf{v}_t^S). \tag{2.9}$$

2.4.2 Stack Data Structure and Composition

As shown in Eq. 2.9, RNNGs decompose the prediction of $(\mathbf{x}, \mathbf{y})$ into a sequence of local decisions that each predicts the next action $a_t \in \{\text{NT}(n), \text{GEN}(w), \text{REDUCE}\}$, conditional on an encoding $\mathbf{v}_t^S$ of the RNNGs' model state $\mathbf{S}_t$. Thus far, we have described how the action sequence $\mathbf{a}$ is derived by traversing $(\mathbf{x}, \mathbf{y})$ in a top-down, left-to-right fashion (§2.4.1). We now turn to describe RNNGs' model state $\mathbf{S}_t$, and aim to shed light on the following questions: How exactly does the RNNGs' model state $\mathbf{S}_t$ change in response to each possible $\{\text{NT}(n), \text{GEN}(w), \text{REDUCE}\}$ action? And what kinds of compositional mechanisms give rise to the explicit modelling of hierarchical structures within RNNGs?

Linearised/non-compositional syntactic language model. Before describing the RNNGs' model state, we begin with a brief description of the linearised syntactic language model of Choe and Charniak (2016). Like RNNGs, this model estimates the joint probability of strings and phrase-structure trees $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, albeit in a *sequential* and *non-compositional* fashion instead, thereby providing an informative contrast to hierarchical RNNGs.

[11] We can decompose the RNNGs' generation of the full syntax tree $(\mathbf{x}, \mathbf{y})$ into its corresponding action sequence $\mathbf{a}$ because, within the context of a top-down, left-to-right traversal of $(\mathbf{x}, \mathbf{y})$, there is a one-to-one mapping between the full syntax tree $(\mathbf{x}, \mathbf{y})$ and its corresponding action sequence $\mathbf{a}$. In other words, each valid action sequence $\mathbf{a}$ uniquely identifies a particular full syntax tree $(\mathbf{x}, \mathbf{y})$, and vice versa.

Concretely, under the sequential syntactic language model of Choe and Charniak (2016), each $a_t \in \{\mathrm{NT}(n), \mathrm{GEN}(w), \mathrm{REDUCE}\}$ action simply adds the relevant symbol—*e.g.,* the non-terminal symbol n in the event where $a_t = \mathrm{NT}(n)$, or the word terminal symbol w in the event where $a_t = \mathrm{GEN}(w)$, or, in the event where $a_t = \mathrm{REDUCE}$, the closed parenthesis symbol ")" to indicate the end of a phrase—onto the next model state $\mathbf{S}_{t+1}$. As each action a_t always appends a new symbol from a finite alphabet $\Sigma \cup N \cup \{\text{``)''}\}$ onto the end of the next model state $\mathbf{S}_{t+1}$, we can represent the Choe and Charniak (2016) model state through a **list data structure**, which supports the addition of a new element onto the end of list in constant time. In Fig. 2.2, we provide a concrete example of the Choe and Charniak (2016) model state $\mathbf{S}_t$ and its corresponding target action a_t, which we illustrate for each step of the top-down, left-to-right generation of our running example in Fig. 2.1.[12]

Choe and Charniak (2016) model state encoding function. We now discuss how to obtain a vector encoding $\mathbf{v}_t^S$ that represents the Choe and Charniak (2016) model state $\mathbf{S}_t$. As shown in Fig. 2.2, the transition from one model state $\mathbf{S}_t$ to the next one $\mathbf{S}_{t+1}$ is always marked by the addition of a new symbol—the exact nature of which is determined by the action a_t—onto the end of the list. Hence, in the case of the the Choe and Charniak (2016) syntactic language model, the model state $\mathbf{S}_t$ can simply be encoded by a conventional RNN architecture that operates in a purely sequential fashion, or any of its variants. In the case of the vanilla RNN (Eq. 2.8), the encoding $\mathbf{v}_t^S$ that represents the model state $\mathbf{S}_t$ is defined as:[13]

$$\mathbf{v}_t^S = \sigma\left(\mathbf{W}\mathbf{e}_t + \mathbf{U}\mathbf{v}_{t-1}^S + \mathbf{b}\right),$$

where $\mathbf{e}_t$ denotes the vector embedding of the input at time step t, which corresponds to the last element of the list $\mathbf{S}_t$. We remark that $\mathbf{e}_t$ can represent a vector embedding of either: (i) a non-terminal symbol $n \in N$, in the event that $a_{t-1} = \mathrm{NT}(n)$ (*e.g.,* step 2 of Fig. 2.2); (ii) a word terminal symbol $w \in \Sigma$, in the event that $a_{t-1} = \mathrm{GEN}(w)$ (*e.g.,* step 4 of Fig. 2.2); or (iii) a closed parenthesis ")", in the event that $a_{t-1} = \mathrm{REDUCE}$ (*e.g.,* step 7 of Fig. 2.2).

[12] As both the Choe and Charniak (2016) syntactic language model and RNNGs generate $(\mathbf{x}, \mathbf{y})$ in a top-down, left-to-right fashion, the action sequence $\mathbf{a}$ is shared across both models, although the model states $\mathbf{S}_t$ can nevertheless be different: RNNGs incorporate an explicit modelling of hierarchical structures through recursive compositions, whereas the Choe and Charniak (2016) model operates in a purely sequential fashion.

[13] In practice, Choe and Charniak (2016) used LSTM architectures (and not vanilla RNNs) to encode the model state $\mathbf{S}_t$, and additionally has an extra annotation that indicates the type of the non-terminal that is being closed, *e.g.,* ")NP", as the last element of step 7 in Fig. 2.2, rather than ")". We omit these differences for brevity, and to make our notation consistent across both RNNGs and the Choe and Charniak (2016) model.

	List content representing the CC'16 model state $\mathbf{S}_t$	Action a_t
1		NT(S)
2	(S	NT(NP)
3	(S | (NP	GEN(*The*)
4	(S | (NP | *The*	GEN(*hungry*)
5	(S | (NP | *The* | *hungry*	GEN(*cat*)
6	(S | (NP | *The* | *hungry* | *cat*	REDUCE
7	(S | (NP | *The* | *hungry* | *cat* |)	NT(VP)
8	(S | (NP | *The* | *hungry* | *cat* |) | (VP	GEN(*meows*)
9	(S | (NP | *The* | *hungry* | *cat* |) | (VP | *meows*	REDUCE
10	(S | (NP | *The* | *hungry* | *cat* |) | (VP | *meows* |)	GEN(.)
11	(S | (NP | *The* | *hungry* | *cat* |) | (VP | *meows* |) | .	REDUCE
12	(S | (NP | *The* | *hungry* | *cat* |) | (VP | *meows* |) | . |)	N/A

Figure 2.2: Joint generation of a sentence and a phrase-structure tree $(\mathbf{x}, \mathbf{y})$, under the Choe and Charniak (2016) sequential syntactic language model. As shown in Eq. 2.9, the generation of $(\mathbf{x}, \mathbf{y})$ is decomposed into a series of decisions that predict the next action a_t, conditional on an encoding of the current model state $\mathbf{S}_t$; the order in which each node is constructed during the top-down, left-to-right traversal of $(\mathbf{x}, \mathbf{y})$ is shown in Fig. 2.1. For the Choe and Charniak (2016) model, the model state $\mathbf{S}_t$ is represented by a list data structure. Here " | " indicates item separators; for example, at step 7 of the generation process, there are 6 elements on the list that represents $\mathbf{S}_t$: ["(S", "(NP", "*The*", "*hungry*", "*cat*", ")"].

	Stack content representing the RNNGs' model state $\mathbf{S}_t$	Action a_t
1		NT(S)
2	(S	NT(NP)
3	(S | (NP	GEN(*The*)
4	(S | (NP | *The*	GEN(*hungry*)
5	(S | (NP | *The* | *hungry*	GEN(*cat*)
6	(S | (NP | *The* | *hungry* | *cat*	REDUCE
7	(S | (NP *The hungry cat*)	NT(VP)
8	(S | (NP *The hungry cat*) | (VP	GEN(*meows*)
9	(S | (NP *The hungry cat*) | (VP | *meows*	REDUCE
10	(S | (NP *The hungry cat*) | (VP *meows*)	GEN(.)
11	(S | (NP *The hungry cat*) | (VP *meows*) | .	REDUCE
12	(S (NP *The hungry cat*) (VP *meows*) .)	N/A

Figure 2.3: Joint generation of a sentence and a phrase-structure tree $(\mathbf{x}, \mathbf{y})$, under RNNGs. Similar to the Choe and Charniak (2016) model, RNNGs generate $(\mathbf{x}, \mathbf{y})$ in a top-down, left-to-right fashon (Fig. 2.1), albeit with one key difference: Each REDUCE action would: (i) pop the top k elements from the stack, up to the last incomplete non-terminal; (ii) compose these elements to obtain a *single* composed phrasal representation, and (iii) push the composed phrasal representation back onto the stack. This is clearly shown in step 7.

The vector encoding $\mathbf{v}_t^S$ that represents the model state $\mathbf{S}_t$ can be used to define a probability distribution over all possible actions a_t through an affine transformation and a softmax activation, as defined in Eqs. 2.4 and 2.5, which we do not repeat here for brevity.

Crucially, the Choe and Charniak (2016) model shows an important distinction between syntactic language models and explicitly hierarchical ones.[14] While the Choe and Charniak (2016) model is indeed a syntactic language model, in the sense that it estimates the joint probability of $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, it nevertheless still operates in a purely sequential fashion—*without* explicit notions of hierarchical structures, where smaller linguistic units like words are recursively composed to form larger ones, such as phrases, clauses, and eventually sentences.

RNNGs' model state. Having described the model state of the Choe and Charniak (2016) sequential syntactic language model, we now describe the RNNGs' model state, and how explicit hierarchical operations are conducted within RNNGs. Whereas the Choe and Charniak (2016) model state is represented by a list, RNNGs' model state is instead represented by a **stack** data structure. Concretely, let push denote an operation that appends a new element onto the top of stack (*i.e.,* as the new last element of the model state $\mathbf{S}_{t+1}$), whereas a pop operation removes the topmost (*i.e.,* the last) element from the stack. The choice of the next action $a_t \in \{\mathrm{NT}(n), \mathrm{GEN}(w), \mathrm{REDUCE}\}$ would yield the following RNNG transitions:

- $a_t \in \{\mathrm{NT}(n), \mathrm{GEN}(w)\}$ would push the corresponding non-terminal symbol n (if $a_t = \mathrm{NT}(n)$) or terminal w (if $a_t = \mathrm{GEN}(w)$) onto the stack that represents the RNNGs' model state $\mathbf{S}_{t+1}$. For example, step 4 of Fig. 2.3 has "*The*" as the last symbol of the model state $\mathbf{S}_4$, because a_3 corresponds to the action $\mathrm{GEN}(The)$. We remark that $a_t \in \{\mathrm{NT}(n), \mathrm{GEN}(w)\}$ would result in the exact same model state transitions as in the case of the Choe and Charniak (2016) model, where the model states for both RNNGs (Fig. 2.3) and the Choe and Charniak (2016) model (Fig. 2.2) are identical for $\mathbf{S}_1, \cdots, \mathbf{S}_6$.
- In contrast, $a_t = \mathrm{REDUCE}$ results in a completely different transition than the Choe and Charniak (2016) model, which simply adds a closed parenthesis symbol ")" in a sequential fashion (*e.g.,* see $\mathbf{S}_7$ and a_6 of Fig. 2.2). In the case of RNNGs, a REDUCE

[14] The sets of syntactic language models and explicitly hierarchical language models are not disjoint. As described here, the Choe and Charniak (2016) model is a syntactic language model that operates sequentially and without explicit notions of hierarchical structures, whereas the ordered neuron LSTM (Shen et al., 2019) represents a non-syntactic language model that nevertheless takes inspiration from the modelling of hierarchical syntactic structures. In contrast, RNNGs are a class of syntactic language models that *also* feature explicit notions of hierarchical structures, which are achieved through recursive composition functions.

action would: (i) pop the last k elements from the preceding model state $\mathbf{S}_t$, up to the last open non-terminal symbol; (ii) **compose** these popped elements in order to obtain a single composed representation for the whole phrase; and (iii) push the resulting composed phrasal representation back as the last element of the new model state $\mathbf{S}_{t+1}$.

An example of the REDUCE action is provided in step 7 of Fig. 2.3. During the course of this REDUCE action, we apply the pop operation to remove the last four elements of $\mathbf{S}_6$, up to "(NP" as the last open/incomplete non-terminal symbol. The four elements that were popped from the stack—["(NP", "*The*", "*hungry*", "*cat*"]—are then composed to form a single phrasal representation for "(NP *The hungry cat*)"; we then push the composed representation back onto the stack, which becomes the topmost stack element at $\mathbf{S}_7$.

After this REDUCE action is complete, there are precisely two elements on the RNNGs' stack at step 7: (i) the *open* non-terminal symbol "(S", and (ii) the composed phrasal representation for "(NP *The hungry cat*)". We refer to these composed phrasal representations, such as "(NP *The hungry cat*)", as a completed/closed non-terminal symbol, as all its children had been generated, and the main non-terminal symbol "(NP" had been closed off through a corresponding REDUCE action. In contrast, under the sequential Choe and Charniak (2016) model that has no explicit composition, the phrase "(NP *The hungry cat*)" is instead represented by a sequence of 5 different symbols: ["(NP", "*The*", "*hungry*", "*cat*", ")"], as shown in step 7 of Fig. 2.2. At the end of the RNNGs' generation process (*e.g.,* step 12, Fig. 2.3), there is precisely one composed phrasal node on the stack, which represents the whole, completed phrase-structure tree "(S (NP *The hungry cat*) (VP *meows*) .)".

RNNGs' model state encoding function. We now turn to the question of how to encode the RNNGs' model state $\mathbf{S}_t$ into its vector representation $\mathbf{v}_t^S$. In the case of the sequential syntactic language model of Choe and Charniak (2016), the transition from $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$ always involves the addition of a new symbol that comes from a finite alphabet (*e.g.,* either $n \in N$, $w \in \Sigma$, or the closed parenthesis symbol ")", as shown in Fig. 2.2), and can thus be encoded by conventional sequential RNN architectures. In contrast, the RNNGs' model state is more complicated to encode for two reasons. First, the elements of the RNNGs' model state do not always consist of symbols from a discrete alphabet. Concretely, as shown in step 9 of Fig. 2.3, the RNNGs' model state not only includes terminals w and open non-terminals n—both of which are symbols from a discrete alphabet Σ and N, respectively—but also includes composed phrasal representations, such as “(NP *The hungry cat*)”, which can be understood as entire subtrees, and are therefore complex structured objects on their own account. Second, the RNNGs’ model state is represented by a stack, which is manipulated using both push and pop operations. For these reasons, it is infeasible to encode the RNNGs’ model state S_t using conventional RNN architectures that operate in a sequential fashion. Hence, the RNNGs’ model state is encoded through a stack LSTM architecture (Dyer et al., 2015), which supports an arbitrary number of not only: (i) standard push operations that add a new input to the LSTM, but also (ii) pop operations that simulate the removal of the topmost stack element by “rewinding” the LSTM pointers. We refer interested readers to Dyer et al. (2015) for more detail on how the stack LSTM architecture is operationalised.

RNNGs’ composition function. A critical component of the RNNGs’ REDUCE action is the composition function, which constructs a composed phrasal representation from its constituents. In the case of our running example (step 7 of Fig. 2.3, where $a_6 = \mathrm{REDUCE}$), the composition function would compose the individual representations for the four popped elements [“(NP”, “*The*”, “*hungry*”, “*cat*”], into a single composed representation for the phrase/subtree “(NP *The hungry cat*)”. In practice, RNNGs implement this composition through bidirectional LSTM networks, as illustrated in Fig. 2.4.[15] The composition function results in a single vector embedding $\mathbf{e}_{\text{composed}}$ that encodes the whole phrase/subtree “(NP *The hungry cat*)”, which serves as input to the stack LSTM at the next time step $t + 1$.

As shown in Fig. 2.4, the first vector read by the LSTM in both the forward and reverse directions is an embedding of the non-terminal label; in the case of our running example, this corresponds to a vector embedding of the “(NP” symbol. This is then followed by the embeddings of the children of the phrase, which can consist of other, smaller composed phrases (*i.e.,* subtrees), or terminal symbols, as is the case in our running example, where the three children are all terminal symbols [“*The*”, “*hungry*”, “*cat*”]. Intuitively, this ordering—that reads the open non-terminal label first, before any of the children nodes—serves to inform each left-to-right or right-to-left LSTM model of what sort of linguistic “headword” it should be looking for, as it processes each of the child node embeddings. In the case of our running example, we expect the composition function LSTM to leverage the fact that

[15] The RNNGs’ composition function differs from that of other work in two ways. First, it is compatible n-ary branching, which enables the composition of arbitrary-length phrases without the need for binarisation (Socher et al., 2013; Dyer et al., 2015). Second, compared to tree LSTMs (Tai et al., 2015) that also allow for n-ary branching, RNNGs’ composition function makes a more extensive use of non-terminal information.

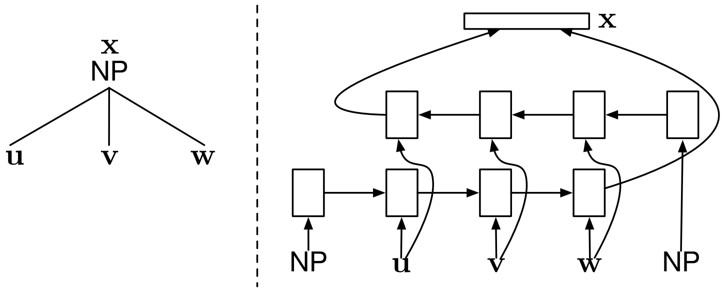


Figure 2.4: The RNNGs' composition function, which executes a bidirectional LSTM during a REDUCE action, in order to compose the elements that constitute the phrase in question. The network on the right models the phrasal/subtree structure on the left. In the case of our running example (step 7 of Fig. 2.3), this composition function combines the representations for ["(NP", "*The*", "*hungry*", "*cat*"] into a single composed representation for "(NP *The hungry cat*)", where the terminals "*The*", "*hungry*", and "*cat*" respectively correspond to the input vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ in this figure. This figure is taken from Dyer, Kuncoro, et al. (2016).

the most salient element of a noun phrase is often the rightmost noun (*e.g.,* the word "*cat*" is often considered the most syntactically salient element of "(NP *The hungry cat*)" under most syntactic theories), although—given the lack of explicit syntactic supervision[16] that identifies the headword of a phrase—RNNGs need to infer this property directly from data.

We remark that the RNNGs' composition function is **recursive** in nature: The exact same bidirectional LSTM composition function is applied at all levels of the tree. In other words, the composition of words into phrases, smaller phrases into larger ones, and larger phrases into clauses and eventually sentences, is handled by the same bidirectional LSTM.

2.4.3 Training

Let $D = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathbf{x}^{(K)}, \mathbf{y}^{(K)})\}$ be a dataset of $K = |D|$ sentences and their corresponding phrase-structure trees. When gold-standard, manually annotated phrase-structure trees are not available for a particular dataset, we use off-the-shelf phrase-structure

[16]Syntactic supervision in the form of which word in a sentence is more syntactically salient than another—*i.e.,* which word is the head, and which word is the dependent—is the key underlying principle behind the dependency grammar formalism (Tesnière, 1959; Mel'čuk, 1988). Nevertheless, as RNNGs make use of phrase-structure grammar as opposed to dependency grammar, they do not have access to syntactic annotations that define which constituent is the most salient element (*i.e.,* the syntactic head) of a particular phrase.

parsers to obtain an automatically generated, predicted phrase-structure tree $\hat{\mathbf{y}}^{(k)}$ for a particular input sentence $\mathbf{x}^{(k)}$. We then treat the predicted tree $\hat{\mathbf{y}}^{(k)}$ as a "silver-grade" phrase-structure tree that the RNNG model aims to generate jointly with the sentence $\mathbf{x}^{(k)}$. While this assumption is reasonable for English due to the existence of accurate and publicly available phrase-structure parsers, this assumption may not necessarily hold for low-resource languages, which constitutes one of the key limitations of supervised RNNGs.[17]

Let $\mathbf{a}^{(k)}$ be the sequence of actions that would generate $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ in a top-down, left-to-right fashion, as outlined in §2.4.1. Furthermore, let $a_t^{(k)}$ be the action that the RNNG model must take at time step t in order to generate $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$. Let $\mathbf{S}_t^{(k)}$ be the RNNGs' model state at time t, after completing all the preceding actions $\mathbf{a}_{<t}^{(k)}$ (see §2.4.2 for more detail on the deterministic transition function that defines the transition from $\mathbf{S}_t^{(k)}$ to $\mathbf{S}_{t+1}^{(k)}$, in response to the action $a_t^{(k)}$). During training, we aim to find the set of RNNG parameters $\hat{\boldsymbol{\theta}}$, such that:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} - \sum_{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in D} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \tag{2.10}$$

$$= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{a}^{(k)} \in D} \sum_{t=1}^{|\mathbf{a}^{(k)}|} \log p_{\boldsymbol{\theta}}(a_t^{(k)} \mid \mathbf{v}_t^{\mathbf{S}^{(k)}}), \tag{2.11}$$

where $\mathbf{v}_t^{\mathbf{S}^{(k)}}$ denotes the stack LSTM (Dyer et al., 2015) vector encoding of the RNNGs' model state $\mathbf{S}_t^{(k)}$. Here $\boldsymbol{\theta}$ denotes the set of all model parameters, which include: (i) the parameters of the stack LSTM that encodes the model state $\mathbf{S}_t^{(k)}$; (ii) the parameters of the bidirectional LSTM that is used as the recursive composition function; and (iii) the two lookup embedding matrices (§2.2, Eq. 2.3) that contain the embedding vector of each possible non-terminal symbol $n \in N$ and each word terminal symbol $w \in \Sigma$, respectively.

2.4.4 Inference

As shown in Eq. 2.10, RNNGs are trained to estimate the joint probability of sentences and phrase-structure trees $p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})$. In order to evaluate RNNGs as a language model—and compare their performance with conventional LSTM or Transformer-based language models that directly estimate $p_{\psi}(\mathbf{x})$—we need to compute the RNNGs' marginal probability $p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \sum_{\mathbf{y} \in Y(\mathbf{x})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})$, under the set of trained RNNG model parameters $\hat{\boldsymbol{\theta}}$. Here we

[17] Unsupervised RNNGs (Kim et al., 2019) exist, although they perform worse on measures of syntactic competence. Furthermore, unsupervised RNNGs are slower to train than their supervised counterparts due to the need to marginalise over the set of all phrase-structure trees $Y(\mathbf{x})$, which in the case of unsupervised RNNGs is achieved through variational inference (Jordan et al., 1999; Wainwright and Jordan, 2008).

use $Y(\mathbf{x})$ to denote the set of all possible phrase-structure trees that are compatible with the input $\mathbf{x}$, where the cardinality of $Y(\mathbf{x})$ is exponential in the input length $|\mathbf{x}|$.

Beyond the language modelling task, RNNGs can also be used as a syntactic parser: Given a sentence $\mathbf{x}$, we aim to find the *maximum a posteriori* (MAP) tree $\hat{\mathbf{y}}$ under the trained RNNG model, where $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in Y(\mathbf{x})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{y} \mid \mathbf{x})$, and $p_{\hat{\boldsymbol{\theta}}}(\mathbf{y} \mid \mathbf{x}) \propto p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})$. Note that the parsing task involves an $\arg\max$ operation that similarly enumerates over all possible phrase-structure trees $Y(\mathbf{x})$. Unfortunately, the use of LSTM networks within RNNGs—both to encode the stack S_t, and to compose completed constituents (Fig. 2.4)—means that the prediction of the next action a_t depends on the *full* history of symbols that had been generated thus far, *without* making any conditional independence assumptions (*e.g.,* Markov assumption, Eq 2.7) that can make exact inference tractable. While this means that exact inference under RNNGs is intractable, we outline a few approximate inference procedures that had been proposed by prior work, both for the language modelling and parsing tasks.

Importance sampling. In the original RNNG paper, Dyer et al. (2016) used a variant of the importance sampling procedure (Doucet and Johansen, 2011) for approximate inference. Concretely, their importance sampling algorithm uses a conditional proposal distribution $q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$—where $\boldsymbol{\phi}$ denotes the set of the proposal model parameters that is separate from the trained RNNG model parameters $\hat{\boldsymbol{\theta}}$—with the following properties: (i) $p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}) > 0 \implies q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x}) > 0$; (ii) samples $\mathbf{y}^{(k)} \sim q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$ can be obtained efficiently; and (iii) the conditional probabilities $q_{\boldsymbol{\phi}}(\mathbf{y}^{(k)} \mid \mathbf{x})$ of these samples are known. To that end, Dyer et al. (2016) defined a *discriminative* version of RNNGs that estimate the probability of phrase-structure trees *conditional* on the input sentence—*i.e.,* $q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$—through a similar set of action sequences and model state transitions as for the generative RNNGs (Fig. 2.3). One key difference is that the discriminative RNNG model assumes that the input sentence is observed, which circumvents the need to predict the surface string $\mathbf{x}$ jointly with the tree structure $\mathbf{y}$; in other words, discriminative RNNGs primarily differ from generative ones by virtue of not predicting GEN(w) actions. We refer interested readers to Dyer et al. (2016) for more technical details on discriminative RNNGs, although we briefly remark on the suitability of discriminative RNNGs as the proposal model. First, discriminative RNNGs lend themselves well to sampling $\mathbf{y}^{(k)} \sim q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$, which can be done through a simple ancestral sampling procedure. Second, the conditional probability of a tree sample $\mathbf{y}^{(k)}$ can easily be obtained through a product of local action probabilities a_t, conditional on the discriminative RNNGs' model state $\mathbf{S}_t$, through a similar top-down, left-to-right structure-building decompositions as in the case of the generative RNNGs (Eqs. 2.10 and 2.11).

Having defined the proposal model that provides an estimate of $q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$, we describe how the importance sampling procedure is used to derive an estimate of $p_{\hat{\boldsymbol{\theta}}}(\mathbf{x})$ under the trained RNNGs. Concretely, importance sampling uses **importance weights**, which are defined as $w(\mathbf{x}, \mathbf{y}) = \dfrac{p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})}$. Under this definition, the estimator is defined as follows:

$$
\begin{aligned}
p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) &= \sum_{\mathbf{y} \in Y(\mathbf{x})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}) \\
&= \sum_{\mathbf{y} \in Y(\mathbf{x})} q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})\, w(\mathbf{x}, \mathbf{y}) && \text{(importance weight definition)} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})}\, w(\mathbf{x}, \mathbf{y}). && \text{(definition of expectation)} \qquad (2.12)
\end{aligned}
$$

We then replace the expectation in Eq. 2.12 with its Monte Carlo estimate, as follows:

$$
\mathbf{y}^{(k)} \sim q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x}) \quad \text{for } k \in \{1, 2, \cdots, K\} \qquad (2.13)
$$

$$
p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})}\, w(\mathbf{x}, \mathbf{y}) \approx \frac{1}{K} \sum_{k=1}^{K} w(\mathbf{x}, \mathbf{y}^{(k)}) \qquad \text{(Monte Carlo approximation)}
$$

Parse reranking. Given a set of trees $\{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\}$ that were sampled according to the proposal model distribution $q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$—as previously shown in Eq. 2.13—we obtain an estimate of the MAP tree $\hat{\mathbf{y}}$ by simply choosing the sampled tree with the highest probability under the generative RNNG model. Formally, $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\}} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})$. This setup is often known as *parse reranking* (Collins and Koo, 2005; Charniak and Johnson, 2005), where we rescore the set of trees produced by the first, proposal model, and return the tree that has the highest score under the second, reranking model. Nevertheless, whereas prior work often used a discriminative model to rerank a set of tree candidates produced by a generative model, the work of Dyer et al. (2016) differs by virtue of conducting the reranking in the opposite direction, where the tree samples produced by a discriminative model is reranked according to their joint probabilities under generative RNNGs.

Lower bound on marginal probability. In the case where the trees $\{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\}$ are sampled without replacement (more formally, $\mathbf{y}^{(i)} \neq \mathbf{y}^{(j)}$ for all $i, j \in \{1, \cdots, K\}$ and $i \neq j$), we can obtain a lower bound on the RNNGs' marginal probability by simply summing over the set of unique tree samples:

$$
\begin{aligned}
p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) &= \sum_{\mathbf{y} \in Y(\mathbf{x})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}) \\
&\geq \sum_{\mathbf{y}^{(k)} \in \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\}} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}^{(k)}).
\end{aligned} \tag{2.14}
$$

This bound holds because the set of the unique sampled trees is a subset of all possible phrase-structure trees that are compatible with the sentence $\mathbf{x}$; *i.e.,* $\{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\} \subseteq Y(\mathbf{x})$.

Word-synchronised beam search. To find the MAP tree $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in Y(\mathbf{x})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y})$, Fried et al. (2017) proposed an approximate beam search procedure that aims to find a set of K high probability trees—denoted as RNNGBeamSearch$(\mathbf{x}; \hat{\boldsymbol{\theta}}) = \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\} \subseteq Y(\mathbf{x})$—under the trained RNNG model. Nevertheless, they found that the standard **action-synchronous** beam search procedure—which has been shown to work well in the case of discriminative neural parsers (Vinyals et al., 2015) by maintaining a beam of K partial trees that share the exact same number of actions taken—fails to work well in the case of generative RNNGs that we explore in this work. To that end, Fried et al. (2017) attribute the failure of such action-synchronised beam search procedure to the probability imbalance between the structure-building actions, such as the open non-terminal action $a_t = \text{NT}(n)$, and the action that would generate the next observed word in the sentence x_j, *i.e.,* $a_t = \text{GEN}(x_j)$. Concretely, the probability of generating the specific next word $a_t = \text{GEN}(x_j)$—out of all the other, tens of thousands of actions that the model could take at time step t instead—tends to be much smaller than the probability of opening *any* non-terminal symbol $a_t = \text{NT}(n), \forall n \in N$. In practice, this probability imbalance often means that the correct action sequence can get pushed off the beam, in favour of bad action sequences that simply prefer to keep opening new non-terminals through a long chain of $a_t = \text{NT}(n)$ actions.

To that end, Fried et al. (2017) proposed a **word-synchronised** beam search procedure that compares parse candidates on a word-by-word level, as opposed to comparing parse candidates that share the same number of actions, as done in the aforementioned action-synchronised beam search procedure. In other words, the word-synchronised beam search procedure compares parse candidates that had generated the same number of words in the sentence, hence circumventing the probability imbalance between actions that generate the correct next word $\text{GEN}(x_j)$, and actions $\text{NT}(n)$ that would open a new non-terminal symbol n. We refer interested readers to the original work of Fried et al. (2017) for more details on this approximate inference procedure. Throughout this thesis, we use the word-synchronised beam search procedure as a means to obtain a set of high-probability trees $\text{RNNGBeamSearch}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ under the RNNG model for a given input sentence $\mathbf{x}$, *without* the need to rely on tree samples from an auxiliary proposal model that estimates $q_{\boldsymbol{\phi}}(\mathbf{y} \mid \mathbf{x})$.

After running the word-synchronised beam search procedure, we can then use the set of K high-probability phrase-structure trees $\text{RNNGBeamSearch}(\mathbf{x}; \hat{\boldsymbol{\theta}}) = \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)}\}$ in three ways. First, we can approximate the MAP tree $\hat{\mathbf{y}}$ by selecting the phrase-structure tree that has the highest probability in the beam. Second, as $\text{RNNGBeamSearch}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ contains a list of K unique phrase-structure trees, we can derive a lower bound on the RNNGs' marginal log likelihood using these trees, as follows:

$$p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) \geq \sum_{\mathbf{y}^{(k)} \in \text{RNNGBeamSearch}(\mathbf{x}; \hat{\boldsymbol{\theta}})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}^{(k)}), \tag{2.15}$$

which holds because $\text{RNNGBeamSearch}(\mathbf{x}; \hat{\boldsymbol{\theta}}) \subseteq Y(\mathbf{x})$. We remark that Eq. 2.15 differs from Eq. 2.14 by virtue of deriving the subset of trees for marginalisation directly from the RNNG model through the word-synchronised beam search procedure, while Eq. 2.14 derives the subset of trees by sampling from the proposal model q. Third, the word-synchronised beam search procedure is *incremental* in nature; hence, we can apply the same procedure to obtain a set of high-probability *partial* phrase-structure trees $\text{RNNGBeamSearch}(\mathbf{x}_{<t}; \hat{\boldsymbol{\theta}})$ that are compatible with a prefix $\mathbf{x}_{<t}$, which does not correspond to a full sentence. Throughout this thesis, we need to obtain the RNNGs' marginal probability for predicting the next word conditional on its history, *i.e.,* $p_{\hat{\boldsymbol{\theta}}}(x_t \mid \mathbf{x}_{<t})$. In Ch. 3, we approximate the RNNGs' marginal next-word distribution by leveraging the resulting set of partial trees $\text{RNNGBeamSearch}(\mathbf{x}_{<t}; \hat{\boldsymbol{\theta}})$, which are derived by running the word-synchronised beam search procedure on $\mathbf{x}_{<t}$, as outlined in §3.5.1 and Eqs. 3.3- 3.6.

Chapter 3

Assessing Syntactic Generalisation in Neural Language Models

Chapter Abstract

In this chapter, we assess the ability of sequential recurrent neural network language models (RNNLMs) to capture long-range syntactic dependencies. We find that, while RNNLMs perform well at a number agreement task designed to test long-range syntactic competence, they partially rely on spurious sequential (rather than structural) cues to succeed in this task; when such sequential cues are not indicative of syntactic structures, their performance may drop considerably. In contrast, a model that has an explicit bias for modelling syntactic structures—recurrent neural network grammars (RNNGs)—substantially outperforms sequential RNNLMs at number agreement, and remains far more robust to sequential perturbations. Altogether, these findings motivate the rest of this thesis by highlighting the benefits of explicit modelling of syntactic structures, even for powerful neural sequence models; in Chs. 4 and 5 we propose a way to inject such syntactic biases in a way that scales well to large datasets.

3.1 Introduction

Recurrent neural networks (RNNs) are remarkably effective models of sequential data, achieving excellent results in both language modeling (Melis et al., 2018) and conditional

The material in this chapter is adapted from Kuncoro et al. (2018a) and Kuncoro et al. (2018b).

generation tasks, such as machine translation (Bahdanau et al., 2015) and text summarisation (See et al., 2017).[1] Despite this success, RNNs operate by modelling word sequences as a linear sequence of symbols—*without* explicit notions of hierarchical syntactic tree structures. A natural question, then, is the following: As RNNs lack explicit modelling of hierarchical syntactic structures, to what extent are these models able to understand and capture long-range syntactic dependencies directly from data, even when they are provided with no explicit supervision? Does the success of RNNs in many tasks—even without explicit modelling of syntactic structures or any hand-crafted syntactic features—mean that such models already exhibit a similar degree of long-range syntactic competence that one would expect from a human language learner? In this chapter, we answer these questions through a number agreement task (Linzen et al., 2016), which aims to assess the ability of language models to capture long-range, syntax-sensitive dependencies that exist in natural language. All in all, our findings in this chapter: (i) highlight the benefits of more explicit syntactic inductive biases for improving syntactic competence at the small data scale ($\sim$3 million words), even in the case of powerful RNN-based neural sequence models that have already been shown to perform well at various sequence modelling tasks; (ii) demonstrate the importance of appropriately controlling for various cues—including cues that humans are unlikely to employ—that a computational model can exploit to succeed at a cognitive task like number agreement, albeit without acquiring the deeper competence that the task was designed to evaluate in the first place; and (iii) showcase the fact that—in the case of RNNGs that build phrase-structure trees through recursive compositions—*how* the tree structure is built also affects their syntactic generalisation; to that end, we find the top-down traversal to perform best for challenging number agreement cases with multiple attractors.

3.2 Background: Number Agreement Evaluation

We begin with an overview of the number agreement task (§3.2), as defined by Linzen et al. (2016), and motivate *why* the task is an important evaluation for language models, above and

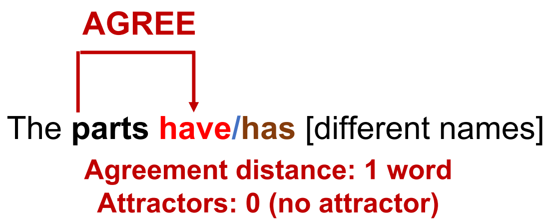
[1]More recently, Transformer architectures (Vaswani et al., 2017) have supplanted RNNs and LSTMs as the state of the art model in many NLP tasks. Most pre-trained large-scale language models, such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020), are based on Transformers, which can scale more easily to large datasets using specialised hardwares like GPUs and TPUs. In this chapter (and also in Ch. 4), we focus on RNN-based architectures, which still constitute a key backbone of many NLP systems in production, and can even outperform Transformers in some cases (Chen et al., 2018a; Lei, 2021). We later shift our focus to Transformers in Ch. 5, where we propose a way to inject syntactic inductive biases into Transformers.

beyond standard perplexity evaluations. We provide an illustration of the number agreement task through the examples in Fig. 3.1. In the first example (Fig. 3.1a), the plural form of the verb "***have***" agrees with the plural subject "***parts***"; hence the singular form of the verb, "***has***", would have been syntactically illicit. In this simple case, the subject "***parts***" and the verb "***have***" also happen to be adjacent in the sequence. Nevertheless, syntactically-related words may not always be adjacent in the sequence, as illustrated in Fig. 3.1b: Even though the word sequences have changed considerably, the syntactic dependency between "***parts***" and "***have***" remains the same. While both examples feature the exact same subject-verb relations, the example in Fig. 3.1b nevertheless poses a more difficult challenge—for human language learners and neural language models alike—for two reasons. First, the number agreement dependency between "***parts***" and "***have***" spans a long distance in the sequence, where there are nine intervening words between the subject and the verb. Hence to succeed in these difficult cases, models would need to capture and propagate syntactic information for a large, potentially unbounded number of steps.[2] Second, the most difficult number agreement cases feature the presence of *attractors* (Bock and Miller, 1991)—defined as intervening nouns with an opposite number from the subject. In our running example (Fig. 3.1b), the four attractors are "*river*", "*valley*", "*commune*", and "*France*" (Fig. 3.1b, denoted in blue). Note that these attractors are all: (i) singular, unlike the plural subject "***Parts***", and (ii) sequentially more adjacent to the main verb "*have*". Hence, models that rely on sequential heuristics and always predict verbs that agree with the most recent noun in the sequence—rather than actually understanding the underlying syntactic structure of the sentence—would fail in the face of attractors, and predict the ungrammatical verb "*has*".

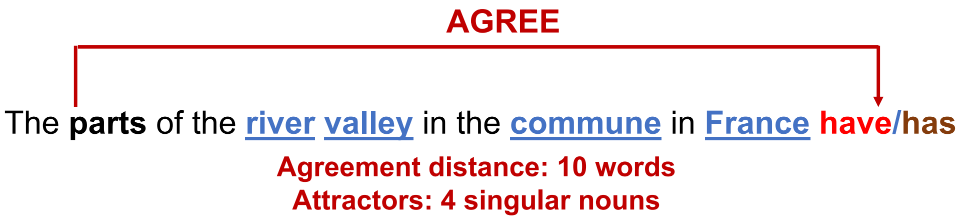
Why should we care about number agreement evaluation? Having described the number agreement task, we outline three reasons why number agreement is a compelling evaluation task for language models.

- Number agreement provides an avenue to measure model generalisation to *long-tail* syntactic phenomena. As shown in Table 3.1, the vast majority ($\sim$95%) of number agreement dependencies are simple, where the subject is also the most adjacent noun to

[2]Subject-verb agreement is a type of syntactic relation that occurs *within* a sentence. Since there is no limit on the number of words in a sentence, the subject-verb agreement dependency can potentially span an unbounded number of intervening tokens. In practice, however, the average sentence length in English falls between 15 to 25. This fact, combined with inherent limitations of the human memory (Miller and Isard, 1964), means that even the longest number agreement dependencies in our dataset rarely exceed 20 tokens.



(a) A simple example of the number agreement task, with a subject-verb distance of one, and no intervening nouns of the opposite number (*i.e.,* attractors) occurring between the subject and the verb.



(b) A difficult example of the number agreement task, with a subject-verb sequential distance of ten words, and four intervening attractors occurring between the subject and the verb. Here the attractors "*river*", "*valley*", "*commune*", and "*France*" (denoted in blue) are all: (i) singular, unlike the plural subject "***Parts***", and (ii) sequentially closer—compared to the distant subject "***Parts***"—to the main verb "*have*". Hence these attractors can distract the model into predicting the ungrammatical, singular verb "*has*", as opposed to the grammatical verb "*have*", which agrees with the distant subject "***Parts***".

Figure 3.1: Two examples of the number agreement task: A simple example (a), and a syntactically more challenging example (b).

the verb (*e.g.,* Fig. 3.1a).[3] Does this fact mean that RNN language models (RNNLMs)—which lack explicit modelling of hierarchical syntactic structures—would rely on a simple sequential heuristic, and always predict verbs that agree with the most recent noun? Or can these models infer the underlying syntactic dependencies that would help them succeed in the *rare* cases where there are intervening attractors between the subject and the verb—even when they are provided with no explicit syntactic supervision to do so? Number agreement evaluation enables us to empirically test these competing hypotheses.

- Number agreement provides a way to measure the ability of neural sequence models to capture long-range dependencies—potentially spanning an unbounded[2] number of tokens—in a *naturalistic* domain that is well-grounded in syntactic theory (Chomsky, 1957; Miller and Isard, 1964). The number agreement task can thus complement other

[3]The fact that long-range number agreement with multiple attractors rarely occurs in the training data further motivates the development of *targeted* evaluation metrics that assess this ability. Indeed, the ability to succeed at challenging number agreement cases with multiple attractors may be obscured in aggregate metrics such as perplexity, where the prediction of the next word rarely requires a deep mastery of syntax.

widely used tasks that target long-range dependencies, including synthetic tasks like copying and additions with long time lags (Hochreiter and Schmidhuber, 1997; Arjovsky et al., 2016), in addition to vision-based tasks like pixel-by-pixel MNIST (Le et al., 2015).

- Despite the apparent difficulty of number agreement with multiple attractors, humans *can* in fact perform quite well, even for these challenging number agreement cases.[4] Such performance is even more striking considering that human learners mostly have access to linguistic stimuli of the simpler form, where there are no attractors between the subject and the verb (*e.g.,* Fig. 3.1a)—a phenomenon known as the poverty of the stimulus (Chomsky, 1971, 1980).[5] While there is little doubt that humans *can* achieve a strong syntactic competence, *how* humans are able to do so (*e.g.,* learning the necessary rules to succeed at number agreement with multiple attractors) from a limited amount of data remains an open research question. Are innate syntactic constraints (Chomsky, 1980) a necessary prerequisite for learning the syntactic rules that modulate number agreement dependencies, or can such rules be acquired, even by learners that lack an explicit preference for hierarchical syntactic structures (*e.g.,* Perfors et al., 2011)? This work makes an empirical contribution to this longstanding debate in cognitive science: If we found that RNNLMs can perform well at number agreement with multiple attractors, even *without* an explicit preference for hierarchical syntactic structures or any explicit syntactic supervision, then this would provide evidence against the necessity of innate structural constraints for building language learners with a strong syntactic competence.

Task formulation. Having described the number agreement task (Linzen et al., 2016) and its importance, we now define the task in more detail. Let $\mathbf{x}$ be a naturally occurring sentence in a text corpus, *e.g.,* $\mathbf{x}$ = "*The* ***parts*** *of the* <u>*river valley*</u> *in the* <u>*commune*</u> *in* <u>*France*</u> ***have*** *different names*" (Fig. 3.1b), where the subject and the verb are denoted in bold, and the four attractors are denoted in underline. Let $\tilde{\mathbf{x}}$ be a corrupted version of $\mathbf{x}$, where the main verb of the sentence is replaced by the opposite number; hence this procedure yields an ungrammatical yet *minimally different* variant of $\mathbf{x}$. In our running example,

[4]Humans nevertheless occasionally make agreement errors, although their prevalence depends on various factors, such as whether or not the attractor is plural (Bock and Miller, 1991).

[5]The poverty of the stimulus argument has been disputed in multiple ways. Pullum and Scholz (2002) showed that syntactically informative examples (*e.g.,* subject-verb agreement with multiple attractors) are not as rare as previously expected. Another line of work argued that learners without explicit hierarchical architectures can nevertheless exhibit a preference for hierarchical rules, even when syntactically complex examples are rarely (or even never) seen at training time (Perfors et al., 2011; McCoy et al., 2018).

$\tilde{\mathbf{x}}$ = "*The **parts** of the river valley in the commune in France ***has** different names*". Let $p_{\boldsymbol{\theta}}(\mathbf{x})$ be the probability assigned to the sentence $\mathbf{x}$ by a trained neural language model (of any architecture), where $\boldsymbol{\theta}$ denotes the set of all model parameters. The language model succeeds at this particular instance if and only if $p_{\boldsymbol{\theta}}(\mathbf{x}) > p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})$, *i.e.,* the model assigns a higher probability to the grammatical sentence (with the correct number agreement), than the ungrammatical one. In practice, we follow Linzen et al. (2016), and—rather than scoring the whole sequence—only score the probability of the two verbs given the prefix, *e.g.,* checking whether $p_{\boldsymbol{\theta}}(\text{have} \mid \text{The parts of the river valley in the commune in France}) > p_{\boldsymbol{\theta}}(\text{has} \mid \text{The parts of the river valley in the commune in France})$. We break down the test set performance based on the number of attractors in order to determine whether, and to what extent, number agreement performance deteriorates with more attractors.

Further remark. Number agreement is an interesting syntactic phenomenon with a long and venerable history in cognitive science (Bock and Miller, 1991), which justifies our choice to use it as our primary instrument of evaluation in this chapter. However, more evaluation tasks are needed to obtain a holistic assessment of language models' syntactic competence. To that end, recent work has proposed a diverse set of syntactic evaluation tasks that can be applied to language models, including grammatical acceptability judgments (Warstadt et al., 2018, 2020), targeted syntactic evaluations covering reflexive anaphora and negative polarity items (Marvin and Linzen, 2018), and a suite of 34 syntactic phenomena introduced by Hu et al. (2020). We hypothesise that model performance in number agreement would be indicative of their ability to master other types of syntactic competence. Later in Chs. 4 and 5, we shall conduct experiments with a more diverse set of syntactic evaluation benchmarks.

3.3 Number Agreement with LSTM Language Models

Having defined the number agreement task and its importance, we now revisit the question asked by Linzen et al. (2016): As RNNs model the surface word sequences without any explicit notion of hierarchical syntactic structures, to what extent are these models able to capture complex subject-verb agreement dependencies that exist in natural language?

	Train	Test
Sentences	141,948	1,211,080
Word types	10,025	10,025
Word tokens	3,159,622	26,512,851

# Attractors	# Instances	% Instances
$n = 0$	1,146,330	94.7%
$n = 1$	52,599	4.3%
$n = 2$	9,380	0.77%
$n = 3$	2,051	0.17%
$n = 4$	561	0.05%
$n = 5$	159	0.01%

Table 3.1: Corpus statistics for the Linzen et al. (2016) number agreement dataset.

Experimental Settings. We use the same parsed Wikipedia corpus, verb inflectors, preprocessing steps, and dataset split as Linzen et al. (2016).[6] Concretely, we use the predicted dependency parse trees[7] of each sentence to determine the subject and the verb of each sentence. We then use a verb inflector to derive an ungrammatical (yet minimally different) variant of each test sentence by changing the number of the verb (*e.g.,* "*Bob likes*" to "*Bob like*"). We only include test cases where *all* intervening nouns are of the opposite number than the subject noun, and discard the rest; for example, if the subject is singular, then we only consider cases where all intervening nouns between the subject and the verb are plural. Following Linzen et al. (2016), we map all word types beyond the most frequent 10,000 to their respective part-of-speech tags. We summarise the dataset statistics—along with the distribution of the number of attractors on the test set—in Table 3.1. Given the difficulty of training vanilla RNNs to capture long-range dependencies (Bengio et al., 1993; Pascanu et al., 2013), we conduct all experiments using long short-term memory (Hochreiter and Schmidhuber, 1997, LSTM) networks. We implement all models using the DyNet library (Neubig et al., 2017a), and report the hyper-parameters at each relevant section.

Training objective. We train the LSTM network with a sentence-level left-to-right language modelling objective that decomposes the sentence in a left-to-right fashion (§2.3). Formally, let $D = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{|D|}\right]$ be our training set, which consists of $|D|$ sentences.

[6]The dataset and scripts are obtained from `https://github.com/TalLinzen/rnn_agreement`.

[7]We resort to predicted dependency parse trees because gold standard syntactic annotations are not available for this dataset. While the use of predicted parse trees means that parsing errors are inevitable, our manual inspection of 50 randomly sampled trees finds such parsing errors to be fairly minimal.

The training process aims to find a set of LSTM model parameters $\hat{\boldsymbol{\theta}}$, such that:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x} \in D} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \tag{3.1}$$

$$= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x} \in D} \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(x_t \mid \mathbf{x}_{<t}), \quad \text{(chain rule decomposition of } p_{\boldsymbol{\theta}}(\mathbf{x})\text{)} \tag{3.2}$$

where T denotes the sequence length of $\mathbf{x}$. Importantly, the model is not *explicitly trained* to predict the correct number of the verb given the prefix. In our setup, the model only has access to the generic language modelling objective at training time; we only apply number agreement evaluation (in conjunction with other evaluation metrics like perplexity) to the trained model at test time. This is because we are not interested in whether LSTMs can succeed at number agreement when they are explicitly trained to do so (indeed, Linzen et al. (2016) already showed that they could), but rather in whether or not an understanding of complex syntactic structures can emerge from the language modelling objective alone, under models like LSTMs that have no explicit preference for hierarchical syntactic structures.

Hyper-parameters. We report the performance of a few different LSTM model sizes in order to understand the impact of *model capacity*—a key driver of recent language modelling progress (Kaplan et al., 2020; Brown et al., 2020)—on number agreement performance; we select all other hyper-parameters based on a grid search. Concretely, we use the following hyper-parameters that work well across different hidden layer sizes: a 1-layer LSTM, optimised with mini-batch gradient descent with an initial learning rate of 0.2 and an exponential learning rate decay of 0.10 that is applied at every epoch after the tenth, a dropout rate of 0.2 that is applied on the word embedding layer, and on the connections between the hidden states and the output logits (but not on the recurrent connections), an input embedding dimension of 50, and a batch size of 10 sentences. Our use of single-layer LSTMs and 50-dimensional word embedding as one of the baselines is consistent with that of Linzen et al. (2016), and hence renders our results directly comparable to theirs.

Larger-scale language model. Following Linzen et al. (2016), we ensure that our findings generalise to a language model trained on more data, by running the same number agreement experiment with the pre-trained model of Jozefowicz et al. (2016). This larger-scale model: (i) was trained on the One Billion Word Benchmark that is 300 times larger than our training

	n=0	n=1	n=2	n=3	n=4
Random	50.0	50.0	50.0	50.0	50.0
Majority	32.0	32.0	32.0	32.0	32.0
Linzen et al. (2016) LSTM, H=50†	6.8	32.6	≈50	≈65	≈70
Our LSTM, H=50	2.4	8.0	15.7	26.1	34.65
Our LSTM, H=150	1.5	4.5	9.0	14.3	17.6
Our LSTM, H=250	1.4	3.3	5.9	**9.7**	13.9
Our LSTM, H=350	1.3	**3.0**	**5.7**	**9.7**	**13.8**
1B Word LSTM (repl)	2.8	8.0	14.0	21.8	20.0
Character LSTM (Melis et al., 2018)	**1.2**	5.5	11.8	20.4	27.8

Table 3.2: Number agreement error rates (**lower is better**) for various LSTM language models, broken down by the number of attractors. The top two rows represent the random and majority class baselines, while the next row (†) contains the reported results from Linzen et al. (2016) for an LSTM language model with 50 hidden units. Some entries in this row, denoted by $\approx$, are approximately derived from a chart, since Linzen et al. (2016) did not provide a full table of results. We report the results of our LSTM implementations of varying hidden layer sizes—along with our re-run of the pre-trained larger-scale Jozefowicz et al. (2016) language model—in the next five rows. In the last row, we report the performance of a state of the art character LSTM baseline with a large model capacity (Melis et al., 2018).

set (Table 3.1), and (ii) used far more parameters than our LSTM models.[8] We test this pre-trained model on the number agreement test set without fine-tuning, although one key difference is that we do not map infrequent word types to their part-of-speech tags, because such preprocessing is not done on the Jozefowicz et al. (2016) training set. Due to the prohibitive cost of running this model on > 1 million test sentences (Table 3.1), we sample a subset of 500 test instances for each number of attractors between zero and four.

Can LSTMs succeed at long-range number agreement with multiple attractors? Contrary to the findings of Linzen et al. (2016), our results in Table 3.2 indicate that some LSTM language models (*e.g.,* Our LSTM, H=350) *are* seemingly able to perform well at the most difficult number agreement cases with multiple attractors—even when they are not provided with syntactic supervision or explicit modelling of syntactic structures. Nevertheless, we observe that this capability is highly contingent on model capacity: For cases with multiple attractors, the LSTM language model with 50 hidden units trails far behind its larger counterparts, despite comparable performance for zero attractor cases. This finding suggests that

[8]The pre-trained large-scale language model is obtained from `https://github.com/tensorflow/models/tree/master/research/lm_1b`.

network capacity plays an especially important role in propagating structural information across a large distance. This trend is also evident from comparing the results with H=150 and H=250: Whereas both models achieve near-identical performance for zero attractor, the model with H=250 performs much better for cases with multiple attractors. Our experiment thus independently derives the same finding as the contemporaneous work of Gulordava et al. (2018), who also found that LSTMs trained with language modeling objectives are, in fact, able to learn number agreement well; here we additionally identify model capacity as one of the reasons for the discrepancy with the findings of Linzen et al. (2016).

Would a larger-scale pre-trained model necessarily do better at number agreement? While the larger-scale model of Jozefowicz et al. (2016) has certain advantages in terms of model capacity, more training data, and richer vocabulary, their model performs worse than our best model (our H=350 LSTM). We attribute this gap to differences between their training domain (the One Billion Word benchmark) and the number agreement test set, whereas our LSTM models benefit from a training dataset that comes from the same domain (*i.e.,* Wikipedia) as the number agreement test set. Nevertheless, their model still performs reasonably well at number agreement, where their model outperforms the majority-class baseline by a large margin, even for the most difficult cases with multiple attractors.

Can character-based models also perform well at number agreement? Prior work has confirmed the notion that, in many cases, statistical models are able to achieve a good performance under some aggregate metric by overfitting to patterns that are predictive in most cases, often at the expense of more difficult, infrequent instances that require deeper language understanding abilities (Rimell et al., 2009; Jia and Liang, 2017). In most cases, syntactic dependencies between subjects and verbs overlap with sequential recency (Table 3.1): It is very often the case that the most recent noun to the verb also happens to be the subject of the sentence. Nevertheless, the fact that number agreement accuracy gets worse as the number of attractors increases is consistent with a *sequential recency* bias in LSTMs: Under this conjecture, identifying the syntactic dependency becomes harder when there are more adjacent nouns that have different numbers from the subject.

If the sequential recency conjecture is correct, then LSTMs would perform worse when the structural dependency is more distant in the sequences, compared to cases where the structural dependency is more adjacent. We empirically test this conjecture by running a strong character-based LSTM language model of Melis et al. (2018) that achieved state of the art results on EnWiki8 from the Hutter Prize dataset (Hutter, 2012), with 1,800 hidden units and 10 million parameters, which is far larger than our best-performing word-based H=350 LSTM model (Table 3.2). The character LSTM is trained, validated, and tested on the same split of the Linzen et al. (2016) dataset. For testing, we similarly measure number agreement accuracy by comparing the probability of the correct and incorrect verbs given the prefix, where the prefix and verbs are represented by their respective character sequences.

The question of whether character-based LSTMs can acquire the same degree of syntactic competence as their word-based counterparts presents an interesting test case. On one hand, character LSTMs have the opportunity to exploit informative morphological cues (an important prerequisite for number agreement), such as the fact that plural nouns in English tend to end with '*s*', while for present-tense verbs, it is singular (and not plural) verbs that often end with '*s*'. Nevertheless, character LSTMs have two key shortcomings. First, character LSTMs lack explicit word segmentations;[9] hence succeeding in this task requires identifying structural dependencies between two *sequences* of character tokens, where the subject and the verb are each represented by arbitrary-length character sequences. In contrast, word-based LSTMs only need to resolve dependencies between two word *tokens*: The word token corresponding to the subject, and the word token corresponding to the verb. Second, by nature of modeling characters, syntactic dependencies are sequentially much further apart than in the case of the word-based models, which is problematic since LSTMs are known to struggle with very long-range dependencies (Arjovsky et al., 2016).

Despite our use of a strong character-based LSTM language model (with state of the art performance on the competitive Hutter Prize dataset c. 2018), we find that the character model performs much worse at number agreement with multiple attractors compared to its word-based counterparts (last row of Table 3.2). This finding supports the earlier conclusion of Sennrich (2017), who found that character-level decoders in neural machine translation perform worse than subword models in capturing morphosyntactic agreement. To some extent, our finding demonstrates the limitations that character LSTMs face in learning structure from language modeling objectives, despite earlier evidence that character LSTM language models are able to implicitly acquire a lexicon (Le Godais et al., 2017). Hence, how

[9]It remains an open question whether LSTMs working directly with character sequences can effectively exploit word segmentations without explicit inductive biases to do so (Kawakami et al., 2019), *e.g.,* whether they can infer that the character sequence "*T h e <space> d o g <space> b i t e s*" has three underlying words.

we can design character-based models with the right inductive biases to learn long-range syntactic dependencies from data remains an important open research question in the field.

3.4 Right for the Wrong Reasons: Diagnosing Sequential Heuristics in Number Agreement

In §3.3, we observe that neural language models with enough capacity are seemingly able to succeed at challenging number agreement cases with multiple attractors, even in the case of LSTM language models that lack syntactic supervision or explicit modelling of syntactic structures. We now continue with an important follow-up question: Does this success *really* mean that LSTMs can already learn complex syntactic dependencies directly from data? Or can model success instead be partially attributed to spurious signals that *just so happen* to be predictive on this dataset, but in reality have little to do with syntactic structures?

Sequential heuristic: first noun memorisation. We begin with a simple observation: In the vast majority of cases (>70% of the sentences within the Linzen et al. (2016) number agreement dataset), the subject of the sentence also happens to be the first noun. This observation is evident from our running example in Fig. 3.1, where the word "*Parts*" is not only the subject of the verb "*have*", but also happens to be the first noun of the sentence. This observation means that there are at least two ways in which the language model can succeed at predicting the correct number of the verb:

- **First hypothesis:** The model succeeds at number agreement by memorising the number of the first noun of the sentence, and predicting the verb that simply agrees with the number of the first noun. In our running example, this means that the LSTM model would store the fact that the first noun of the sentence, "*Parts*", is plural, propagate this information for multiple time steps, and later assign a higher probability to the plural verb "*have*" as opposed to "*has*". Under this hypothesis, model success at number agreement can be attributed to this *sequential*, structure-independent heuristic; hence a strong number agreement performance does not necessarily mean that the model has learnt about the underlying syntactic rules. Under this hypothesis, the model is right for the wrong reasons.
- **Second hypothesis**: The model succeeds at number agreement by identifying the syntactic structure of the sentence, including the subject and the verb. In our running example,

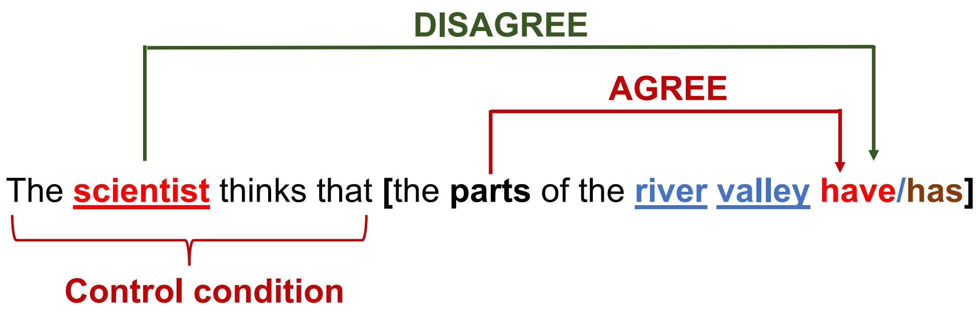


Figure 3.2: An example of our constructed control condition for a case with two attractors (blue), where the first noun "*scientist*" has a different number than the verb. Here the correct verb is "*have*", which agrees with the subject "***parts***", as opposed to the first noun "*scientist*".

this means that the model is able to correctly identify "*Parts*" as the subject of "*have*", regardless of whether "*Parts*" is the first noun of the sentence. Under this hypothesis, the model is right because it has appropriately acquired the syntactic competence that the number agreement task was designed to evaluate in the first place.

Control condition. Here we aim to understand *how exactly* LSTMs succeed at number agreement, and empirically validate which of these two competing hypotheses is correct. To that end, we introduce a control condition through the insertion of a matrix clause, such that the number of the first noun is not predictive of the number of the verb. We provide an illustration of this control condition in Fig. 3.2. We remark that this control condition perturbation is *sequential* in nature: The syntactic dependency between the subject and the verb (here between "***parts***" and "***have***") remains exactly the same as in the case without the matrix clause insertion. Under this setup, we train the model in the same way as before, and only apply this control condition perturbation at test time. We argue that models that can capture the latent syntactic structure of the sentence—instead of incorrectly relying on the number of the first noun to predict the number of the verb—should be robust to this sequential perturbation, and thus exhibit a similar level of number agreement performance as in the case without the control condition. In some sense, this control condition can be understood as a template for creating adversarial examples (Szegedy et al., 2014; Goodfellow et al., 2015)—specially crafted inputs that have little impact on human predictions, but can nevertheless mislead artificial models into making the wrong predictions.

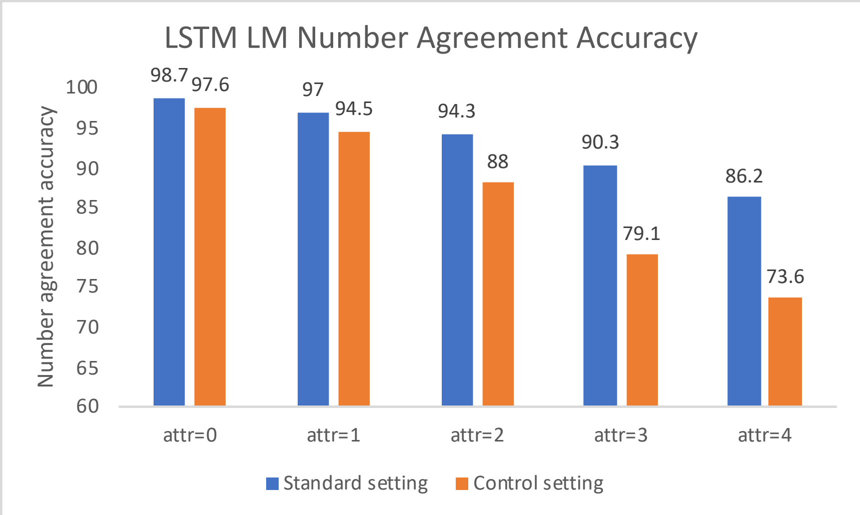


Figure 3.3: Number agreement accuracy for LSTMs (**higher is better**), under the standard condition (left bars, shown in blue) and the control condition (right bars, shown in orange). The results are broken down by the number of attractors between the subject and the verb.

Discussion. As shown in Fig. 3.3, we find that the LSTM performance degrades under the control condition; this degradation is present throughout all numbers of attractors. Whereas the degradation is fairly small for the simplest cases with zero and one attractor, we observe a substantial degradation for the most difficult cases with three and four attractors, where the model performance degrades by up to 13% in terms of absolute performance difference. In fact, under the control condition, model performance with four attractors (73.6%, Fig. 3.3) is hardly better than the majority-class baseline (68%; Table 3.2). Altogether, this pattern of results suggests that the model relies more on the first-noun heuristic for the most syntactically complex cases where there are multiple attractors—in addition to a long sequential distance—between the subject and the verb. Nevertheless, the model does not rely on this first-noun heuristic as much for the simplest cases with zero and one attractor, for which the model can already have a near-perfect accuracy under the standard condition.

All in all, we conclude that LSTM language models *do* partially rely on this structure-independent, first-noun heuristic to succeed at number agreement with multiple attractors, and that strong model performance under the standard condition is not necessarily indicative of its mastery of complex syntactic structures. We remark on two further implications of our findings. First, our findings represent a step towards a better understanding of the potential flaws in generalisation that RNNs make in learning language. Second, our findings highlight the risks associated with uncritically using diagnostics from cognitive science as assessment tools in machine learning. In fact, the exact same control experiment that decreases model performance would be uninteresting to run in a human experiment, because nouns in matrix clauses (*e.g.,* "*the scientist*" in Fig. 3.2) are thought to have little impact on the processing of subordinate clauses (cf. Franck et al., 2006), and are thus very unlikely to affect human number agreement performance in a material way. Hence, our findings affirm the importance of appropriately controlling for various cues—including ones that humans are unlikely to employ—that an artificial learner can exploit to succeed at a cognitive task, albeit without acquiring the deeper competence that the task was designed to evaluate in the first place.

Future work. While our findings establish that, to some extent, LSTMs succeed at complex number agreement for the wrong reasons, we remark on two potential avenues for improvement, which we leave to future work. First, we have thus far only investigated one sequential heuristic that exploits the strong correlation between the number of the first noun and the number of the verb. It is conceivable (and rather likely) that the model may also rely on other sequential heuristics to succeed at number agreement; a better understanding of what these other potential sequential heuristics are remains within the realm of future work. Indeed, the model may exploit *multiple* such sequential heuristics, which may work in conjunction to account for their apparent success at syntax-sensitive tasks, without necessarily acquiring the expected syntactic competence. Second, we investigated the first noun heuristic by manually noticing that, in the vast majority of cases, the first noun coincides with the subject of the sentence. How we can automatically detect such heuristics—for instance, through more advanced interpretability and explainability techniques that can shed light on *why* the model makes a particular decision—is an important avenue for future work.

3.5 Number Agreement with RNNGs

Having established that LSTM language models do not always succeed at complex number agreement cases for the right reasons, can we do better with a model that explicitly incorporates hierarchical syntactic structures as an inductive bias? Here we focus on recurrent neural network grammars (Dyer et al., 2016, RNNGs), which jointly model the probability of strings $\mathbf{x}$ and phrase-structure trees $\mathbf{y}$, *i.e.,* $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, through structure-building actions and recursive compositions. We provide a detailed review of RNNGs in §2.4.

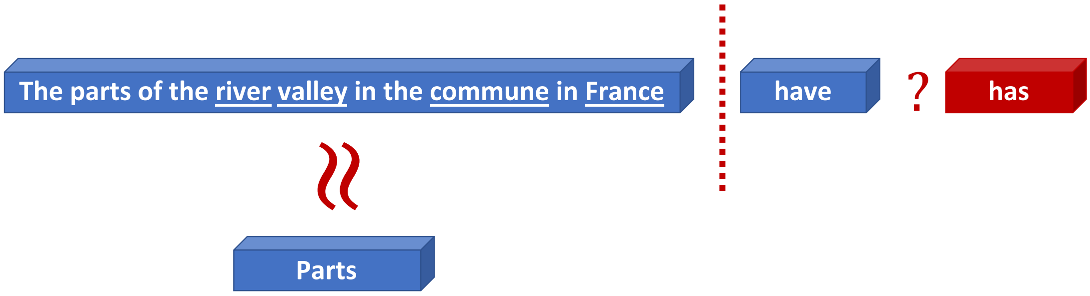


Figure 3.4: An example of how RNNGs' syntactic inductive biases and explicit composition can facilitate a better modelling of long-range dependencies. Since RNNGs are known to capture syntactic headedness (Kuncoro et al., 2017), the composed representation for the phrase "*The **parts** of the river valley in the commune in France*" would be more similar to the syntactic head "***parts***" as the most salient element of the phrase, as opposed to the four attractors (underlined). The close proximity of "***parts***" and "***have***" after the RNNGs' explicit compositions would make it easier to predict the correct number of the verb, despite the large sequential distance—and the occurrence of four attractor nouns—between them.

Our choice of RNNGs is motivated by the findings of Kuncoro et al. (2017), who found evidence for syntactic headedness in RNNGs' composed phrasal representations. Intuitively, the ability to learn syntactic heads would be beneficial for this task, because the composed representation for the phrase "*The **parts** of the river valley in the commune in France*" (running example, Fig. 3.1b) would be more similar to the plural syntactic head "***parts***", rather than the four singular attractors "*river*", "*valley*", "*commune*", and "*France*". In some sense, the composition operator injects an explicit *structural recency* bias into RNNGs (as opposed to the sequential recency bias of LSTMs, §3.3), as subjects and verbs that are sequentially far apart—such as "***parts***" and "***have***"—are nevertheless close together in the composed phrasal representations (Fig. 3.4). Put simply, we hypothesise that the syntactic inductive biases and recursive compositions within RNNGs endow them with the ability to model *long-range sequential* dependencies as *short-range structural* dependencies.

3.5.1 Experiments

We now summarise the experimental settings for running RNNGs on the number agreement dataset, and discuss the key empirical findings.

Experimental settings. Since the Linzen et al. (2016) dataset lacks gold-standard phrase-structure tree annotations, we obtain predicted phrase-structure trees for each sentence using a publicly available discriminative model[10] trained on the Penn Treebank (Marcus et al., 1993). We then train the RNNG on these predicted phrase-structure trees of the Linzen et al. (2016) dataset, with about 140 thousand training instances (Table 3.1).[11] At test time, we compare the RNNGs' marginal probabilities of generating the correct and incorrect verbs given the prefix. Concretely, let x_i be the main verb of the sentence $\mathbf{x}$. Under the RNNG, the marginal probability for generating x_i—conditional on its prefix $\mathbf{x}_{<i}$—is defined as follows:

$$p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}) = \sum_{\mathbf{y}_{<i} \in Y(\mathbf{x}_{<i})} p_{\hat{\boldsymbol{\theta}}}(x_i, \mathbf{y}_{<i} \,|\, \mathbf{x}_{<i}) \tag{3.3}$$

$$= \sum_{\mathbf{y}_{<i} \in Y(\mathbf{x}_{<i})} p_{\hat{\boldsymbol{\theta}}}(\mathbf{y}_{<i} \,|\, \mathbf{x}_{<i})\ p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}, \mathbf{y}_{<i}) \quad \text{(chain rule decomposition)} \tag{3.4}$$

$$= \mathbb{E}_{p_{\hat{\boldsymbol{\theta}}}(\mathbf{y}_{<i} \,|\, \mathbf{x}_{<i})} \left[p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}, \mathbf{y}_{<i}) \right], \quad \text{(definition of expectation)} \tag{3.5}$$

where $Y(\mathbf{x}_{<i})$ denotes the set of all possible, partially completed[12] phrase-structure trees that are compatible with the observed prefix $\mathbf{x}_{<i}$, whereas $\hat{\boldsymbol{\theta}}$ denotes the parameters of the trained RNNG model. Unfortunately, computing Eq. 3.4 is expensive because it requires exact marginalisation over all possible partially completed phrase-structure trees $\mathbf{y}_{<i} \in Y(\mathbf{x}_{<i})$, which is intractable because the cardinality of this set is exponential in the length of $\mathbf{x}_{<i}$.

Given the computational costs of exact marginalisation, we employ two approximations to compute the expectation in Eq. 3.5. First, we only consider a smaller set of partial phrase-structure trees derived from a beam search procedure. Concretely, let $\text{RNNGBeamSearch}(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})$ denote the set of partial phrase-structure trees obtained by running the incremental, word-synchronised RNNG beam search procedure of (Fried et al., 2017, §2.4.4), with $\mathbf{x}_{<i}$ as the input, and $\hat{\boldsymbol{\theta}}$ as the trained RNNG parameters. Furthermore, let $B = \left|\text{RNNGBeamSearch}(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})\right|$ denote the size of the beam. The aim of this beam search procedure is to find a smaller set of B partial phrase-structure trees that are deemed to have high probabilities (conditional on $\mathbf{x}_{<i}$) under the trained RNNG model. By only

[10] `https://github.com/clab/rnng`

[11] Earlier work on RNNGs (Dyer et al., 2016; Kuncoro et al., 2017) trained the model on gold-standard phrase-structure trees from the Penn Treebank, whereas here we train the RNNG model based on predicted, "silver-grade" trees from another parser, which introduces the possibility of parse errors. However, our manual inspection of a randomly sampled set of 30 predicted trees finds such parse errors to be minimal.

[12] The set of trees $Y(\mathbf{x}_{<i})$ in Eq. 3.4 only includes all possible *partially completed* trees that are compatible with the observed prefix $\mathbf{x}_{<i}$, because the phrase-structure trees are only complete when the entire sentence (of length T), $\mathbf{x} = x_1, x_2, \cdots, x_T$, has been observed.

considering the top B trees that are returned by the beam search procedure, this approximation means that we consider all other partial trees $\tilde{\mathbf{y}}_{<i}$ that are not in the beam to have a zero probability, *i.e.,* $p_{\hat{\boldsymbol{\theta}}}(\tilde{\mathbf{y}}_{<i} \,|\, \mathbf{x}_{<i}) \approx 0, \ \forall \, \tilde{\mathbf{y}}_{<i} \notin \text{RNNGBeamSearch}(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})$.

Second, we assume that all partial trees that *are* in the beam have a uniform probability, *i.e.,* $p_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{y}}_{<i} \,|\, \mathbf{x}_{<i}) \approx \frac{1}{B}, \ \forall \, \hat{\mathbf{y}}_{<i} \in \text{RNNGBeamSearch}(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})$. With these two assumptions in mind, we approximate Eq. 3.5 as follows:

$$\begin{aligned} p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}) &= \mathbb{E}_{p_{\hat{\boldsymbol{\theta}}}(\mathbf{y}_{<i} \,|\, \mathbf{x}_{<i})} \left[p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}, \mathbf{y}_{<i}) \right], && \text{(Eq. 3.5, reproduced)} \\ &\approx \frac{1}{B} \sum_{\hat{\mathbf{y}}_{<i} \in \text{RNNGBeamSearch}(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})} p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}, \hat{\mathbf{y}}_{<i}). && (3.6) \end{aligned}$$

Note that $B \ll |Y(\mathbf{x}_{<i})|$, which substantially reduces the computation costs of our approximation compared to exact marginalisation (Eq. 3.4).[13] Similar to the LSTM experiments (§3.3), we compare the probability of the correct verb given the prefix, *i.e.,* $p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i})$, against the probability of the incorrect verb given the same prefix, $p_{\hat{\boldsymbol{\theta}}}(\tilde{x}_i \,|\, \mathbf{x}_{<i})$,[14] where each estimate is computed using the approximation defined in Eq. 3.6. The model is deemed correct for a number agreement test instance $\mathbf{x}$ if and only if $p_{\hat{\boldsymbol{\theta}}}(x_i \,|\, \mathbf{x}_{<i}) > p_{\hat{\boldsymbol{\theta}}}(\tilde{x}_i \,|\, \mathbf{x}_{<i})$.

Sequential syntactic LSTM baseline. RNNGs have two key differences from standard, sequential LSTM language models: (i) They are trained with phrase-structure syntactic annotations, and aim to model the joint probability of strings and phrase-structure trees, *i.e.,* $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, and (ii) they incorporate explicit compositional mechanisms, where the ordering of these recursive compositions is determined by hierarchical syntax trees; in other words, the model architecture is determined by syntax.[15] Here we ask: To what extent is it important to have explicit recursive compositions? Would a syntactic language model that similarly estimates $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ and leverages phrase-structure syntactic annotations at training time—but *without* explicit compositional operations—do as well as RNNGs? We answer these questions by running a **sequential syntactic LSTM** baseline of Choe and Charniak (2016, §2.4), which models strings and phrase-structure trees (a similar $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ training objective

[13] A better alternative than using the trees from word-synchronised beam search is to use unbiased sampling methods to sample partially completed trees from $p_{\hat{\boldsymbol{\theta}}}(\mathbf{y}_{<i} \,|\, \mathbf{x}_{<i})$, although it is not trivial to do so given the exponential nature of the sample space and the sequential nature of RNNG predictions. We leave the question of how we can design efficient and unbiased partial-tree sampling methods from RNNGs to future work.

[14] Since the prefix $\mathbf{x}_{<i}$ is shared between the correct and incorrect verb choices, we only need to run the word-synchronised beam search procedure once to obtain the set of tree samples RNNGBeamSearch$(\mathbf{x}_{<i}; \hat{\boldsymbol{\theta}})$.

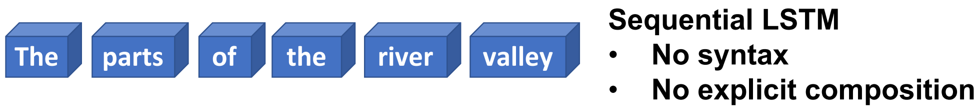
[15] Prior work has introduced other neural models with similar tree-based compositions as RNNGs, such as tree LSTMs (Tai et al., 2015) and neural module networks (Andreas et al., 2016a,b).

as RNNGs), albeit in a sequential fashion, and without explicit modelling of hierarchical tree structures. In Fig. 3.5, we illustrate the key differences between the three models we use: (i) Sequential LSTMs without syntax, (ii) sequential syntactic LSTMs, and (iii) RNNGs.

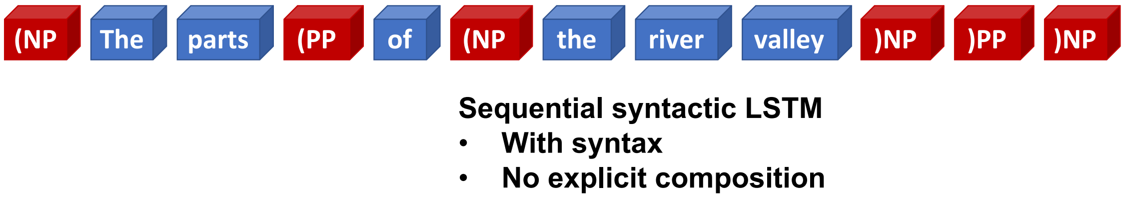
3.5.2 Empirical Findings

Discussion. We report the number agreement performance of the three models in Fig. 3.6. We observe that RNNGs (rightmost, in grey) achieve a much better number agreement accuracy compared to LSTM language models (leftmost) for the most difficult cases with four and five attractors, with a substantial $\sim$30% relative error rate reduction each. Concretely, for cases with five attractors, LSTMs achieve 17.7% error rates (82.3% accuracy), while RNNGs achieve 12% error rates (88% accuracy), corresponding to a 31% relative error rate reduction. For four attractors, LSTMs achieve 13.8% error rates (86.2% accuracy) compared to 9.4% for RNNGs (90.6% accuracy), corresponding to a 32% relative error rate reduction. For cases with three attractors, RNNGs achieve a still-substantial 11% relative error rate reduction (from 9.7% to 8.6%) compared to the sequential LSTM model with no syntax. We attribute the slightly worse performance of RNNGs on cases with zero and one attractor to the presence of intervening structure-building actions that separate the subject and the verb, such as a REDUCE (step 6 in Fig. 3.8(a)) action to complete the noun phrase, and at least one action to predict a verb phrase (step 15 in Fig. 3.8(a)), before the verb is introduced. Hence, one downside of RNNGs is that there is a longer dependency in the simplest cases with no attractor, whereas LSTMs benefit from shorter dependencies in these cases.

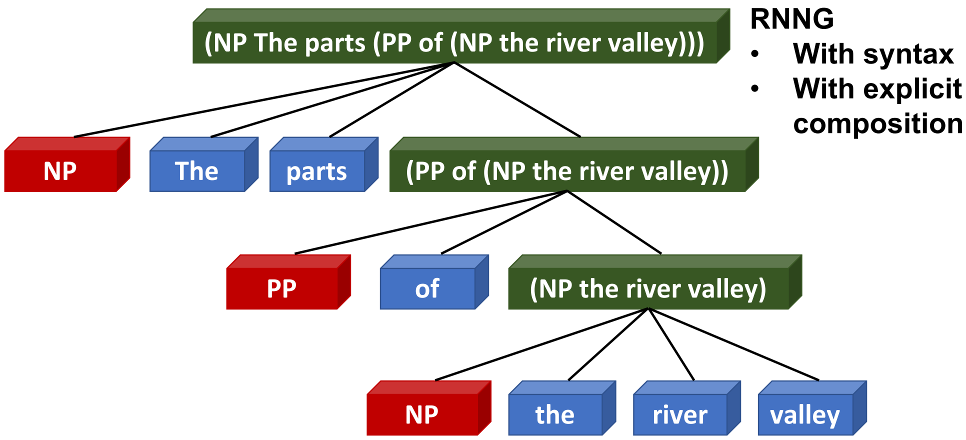
Modelling syntax without explicit composionality is not enough. Our findings in Fig. 3.6 suggest that the sequential syntactic LSTMs (centre), which incorporate syntactic control symbols, albeit without explicit modelling of compositional tree structures, perform comparably with sequential LSTMs without syntax. Notably, this model consistently underperforms RNNGs for nearly all attractors; furthermore, RNNGs perform substantially better for cases with three to five attractors. Indeed, for cases with three, four, and five attractors, the performance of sequential syntactic LSTMs is much more comparable to that of the sequential LSTM without syntax baseline. This result suggests that modelling syntax alone, without explicit compositonal inductive biases, is *not* enough to build language models that exhibit a strong syntactic competence, as measured by number agreement accuracy. Indeed, our findings are consistent with that of Yogatama et al. (2018), who found that introducing



(a) The sequential LSTM model without syntax. This model represents the context as a linear sequence of terminal symbols (*i.e.,* only the words, denoted in blue), with no access to the syntactic phrase-structure annotations (*e.g.,* "(NP", "(PP", etc., as shown in Figs. 3.5b and 3.5c); nor does this model benefit from explicit composed representations of the context.



(b) The sequential syntactic LSTM model of Choe and Charniak (2016). This model has access to syntactic phrase-structure annotations (*e.g.,* "(NP" and "(PP", denoted in red), and—like RNNGs but unlike sequential LSTMs without syntax—estimates the joint probability of strings and phrase-structure trees, *i.e.,* $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. Nevertheless, this model still represents the context (which now contains word terminals, syntactic non-terminals, and the closed parentheses symbols) as a *linear* sequence, and lacks explicit modelling of hierarchical syntactic structures through recursive compositions.



(c) The RNNG model (Dyer et al., 2016). RNNGs not only have access to syntactic phrase-structure annotations, but also leverage explicit composed representations of the context by virtue of a recursive composition function—unlike sequential syntactic LSTMs (Fig. 3.5b) that still represent the context as a linear sequence of symbols. Note that the specific instantiation of the RNNG architecture is determined by the syntax tree. The entries in green denote composed phrasal representations.

Figure 3.5: An illustration of how the following three models: (a) sequential LSTMs without syntax, (b) sequential syntactic LSTMs, and (c) RNNGs represent the same context "(NP *The parts* (PP *of* (NP *the river valley*)))" in different ways.

elements of hierarchical modeling through a stack-structured memory—in essence introducing similar hierarchical inductive biases as RNNGs, albeit through different means—is beneficial for number agreement; their stack-structured memory model outperforms LSTMs

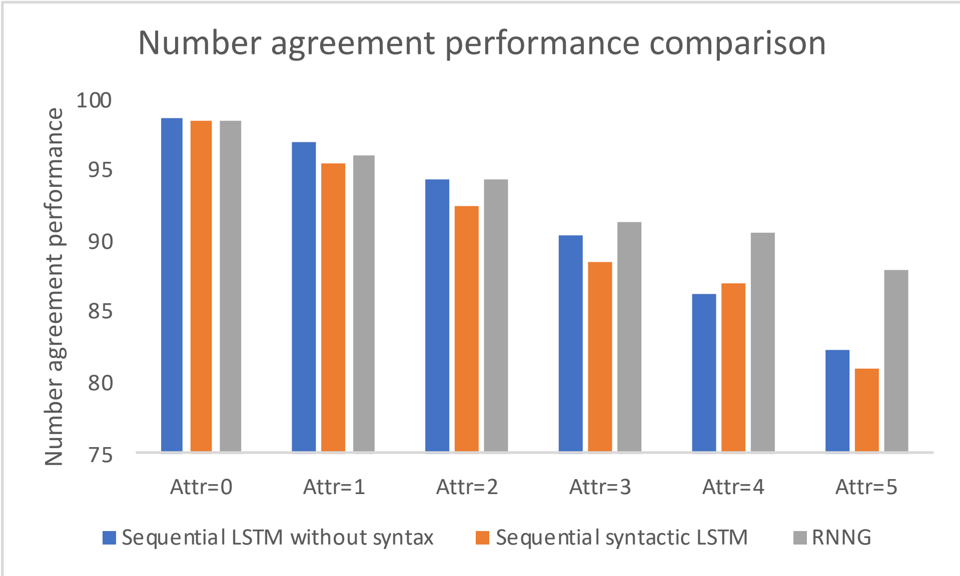


Figure 3.6: Number agreement performance (**higher is better**) for sequential LSTMs without syntax (left, in blue), sequential syntactic LSTMs (centre, in orange), and RNNGs (right, in grey).

and attention-augmented LSTMs by increasing margins as the number of attractors grows.

RNNGs are much more robust to sequential perturbations. While sequential LSTMs without syntax could achieve a decent performance at number agreement, we partially attribute their success to a sequential heuristic that memorises the first noun of the sentence—*without* necessarily understanding the syntactic structure of the sentence (§3.4). Do RNNGs also rely on the same sequential cues to succeed at number agreement? We answer this by evaluating RNNGs on the control condition instances as previously described in Fig. 3.2, where we construct adversarial examples through sequential perturbations that break the correlation between the number of the first noun and that of the verb. If RNNGs are able to learn the syntactic rules that govern number agreement dependencies, then they would be robust to such sequential perturbations, and perform comparably under the control condition.

Our findings in Fig. 3.7 affirm that this is largely the case: For all numbers of attractors, RNNG performance is rather consistent under both the standard setup and the control condition. For four attractors, whereas LSTMs achieved 73.6% number agreement accuracy under the control condition—hardly better than the majority class baseline that gets 68%, and a substantial $\sim$13% drop from 86.2% under the standard setup (Fig. 3.3)—RNNG performance only dropped a little from 90.6% to 88%. We attribute the RNNGs' slightly

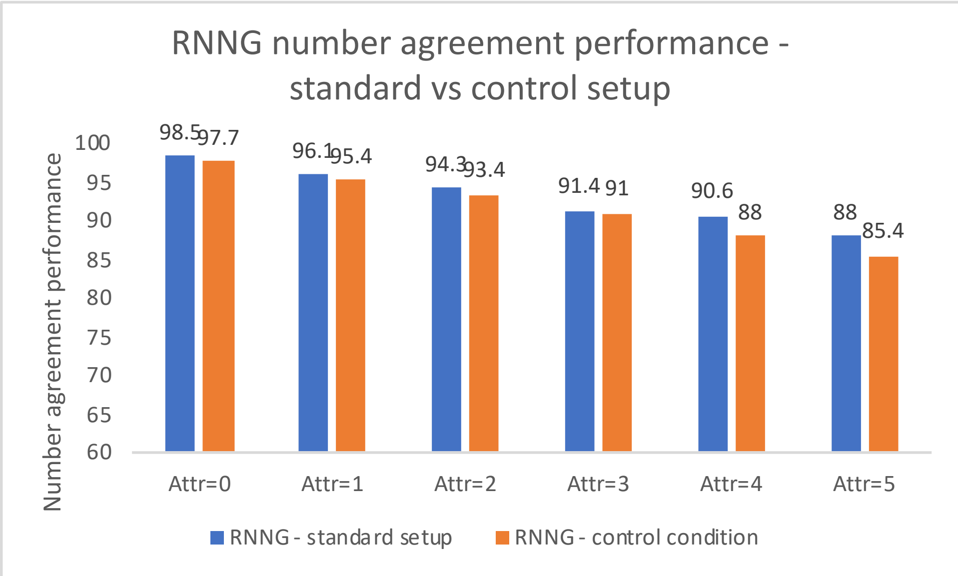


Figure 3.7: RNNG performance (**higher is better**) under the standard (blue) and the control (orange) conditions.

worse performance under the control condition to the increased complexity introduced by the matrix clause insertion (see Fig. 3.2), although the decline is far smaller than for LSTMs (Fig. 3.3). Hence, we conclude that—unlike LSTMs—RNNGs do *not* succeed at number agreement by partially relying on a sequential heuristic that memorises the number of the first noun of the sentence, and that RNNGs are able to display a much more robust and human-like syntactic competence under sequential perturbations.

Language modelling perplexity does not perfectly correlate with number agreement performance. Within the language modelling task, perplexity on a held-out set has been used as the primary metric—and often the only metric—for measuring language modelling progress. But to what extent can we rely on perplexity to measure language models' syntactic competence? Should we supplement perplexity with additional metrics that specifically aim to assess syntactic competence, such as number agreement? We now answer this question by measuring the validation perplexity of each of the three models in Fig. 3.6, and compare how the perplexity of each model aligns with its performance at number agreement.

To measure perplexity, we need an estimate of $p_{\boldsymbol{\theta}}(\mathbf{x})$ from all three models. This estimate is straightforward to obtain for the sequential LSTM without syntax, although estimating this quantity from sequential syntactic LSTMs and RNNGs necessitates an intractable marginalisation over all possible trees for each test sentence $\mathbf{x}$; as before, we denote the

	Validation ppl.
LSTM LM	**72.6**
Seq. Syntactic LSTM	79.2
RNNGs	77.9

Table 3.3: Validation set perplexity (**lower is better**) of the three models: sequential LSTM without syntax, sequential syntactic LSTM, and RNNG. We report a lower bound on perplexity (Eq. 3.7) for the last two models, which are both syntactic language models.

set of such trees as $Y(\mathbf{x})$. Hence for both models, we instead report a lower-bound on the marginal probability by summing over a smaller set of trees. For each test sentence $\mathbf{x}$, we sample a set of 100 trees without replacement from a publicly available, off-the-shelf parser,[16] and marginalise over this smaller set of trees. More formally,

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{y} \in Y(\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \geq \sum_{\mathbf{y} \in \text{SampledTrees}(\mathbf{x}, \boldsymbol{\psi})} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}), \tag{3.7}$$

where SampledTrees$(\mathbf{x}, \boldsymbol{\psi})$ denotes the set of 100 sampled trees obtained from a separately trained parser that was trained to estimate $q_{\boldsymbol{\psi}}(\mathbf{y} \mid \mathbf{x})$; here $\boldsymbol{\psi}$ denotes the parameters of this trained "tree proposal" model. The bound holds because the sampled trees are guaranteed to be a subset of all plausible trees for a sentence $\mathbf{x}$, *i.e.,* SampledTrees$(\mathbf{x}, \boldsymbol{\psi}) \subseteq Y(\mathbf{x})$.

As shown in Table 3.3, the sequential LSTM without syntax has the lowest validation set perplexity—despite its substantially worse performance than RNNGs at number agreement with multiple attractors (Fig. 3.6). This finding suggests that lower perplexity is not perfectly correlated with number agreement success, and highlights the need to supplement perplexity evaluation with more targeted measures of syntactic competence, which can benchmark progress on language models' ability to acquire a human-like syntactic competence.

3.6 Top-Down, Left-Corner, and Bottom-Up Traversals

Having established the importance of explicit modelling of compositional syntactic structures, we explore the hypothesis that *how* we build the structure also affects the model's ability to identify structural dependencies in English. As RNNGs build phrase-structure trees through top-down operations, we propose extensions to the structure-building sequences and model architecture that enable bottom-up (Chelba and Jelinek, 2000; Emami and Jelinek, 2005, §3.6.1) and left-corner (Henderson, 2003, 2004, §3.6.2) RNNG generation orders.

[16] `https://github.com/clab/rnng`

Why is the build order important? Here we state our hypothesis on why the build order matters. The three generation strategies represent different chain rule decompositions of the joint probability of strings and phrase-structure trees, $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, thereby imposing different inductive biases on the learner. Indeed, earlier work in *parsing* has characterised the cognitive plausibility of top-down, left-corner, and bottom-up strategies as viable candidates of human sentence processing, especially in terms of memory constraints and human difficulties with centre embedding constructions (Johnson-Laird, 1983; Pulman, 1986; Abney and Johnson, 1991; Resnik, 1992, *inter alia*), along with neurophysiological evidence in human sentence processing (Nelson et al., 2017). In this work, we move away from the realm of parsing, and evaluate the three strategies as models of *generation* instead, and address the following empirical question: Which RNNG generation order has the most appropriate inductive biases to model complex structural dependencies in English, as indicated by number agreement accuracy? Since all three variants still generate the words sequentially from left-to-right, the models can thus be compared using the same number agreement diagnostic.[17]

Furthermore, these different build orders organise the learning problem so as to yield intermediate states that condition on different aspects of the grammatical structure. In number agreement, this amounts to making the subject, such as the word "*flowers*" in Fig. 3.8, more or less salient. If it is more salient, the model should be able to better predict the number of the verb—in agreement with this subject—without getting distracted by the attractors. We compare the three build orders in Fig. 3.8, which shows the respective configurations (*i.e.,* the prefix) when generating the main verb, for a sentence with a single attractor.[18] In addition, we show the concrete action sequences in each relevant subsection.

3.6.1 Bottom-Up Traversal

In bottom-up traversals, phrases are recursively constructed and labelled with their non-terminal type once all their constituents have been generated, as illustrated in Fig. 3.9. Bottom-up traversals thus benefit from shorter stack depths compared to top-down traversals—a fact we attribute to the lack of incomplete non-terminals on the stack (Table 3.4). On the other hand, the commitment to label the non-terminal type of a phrase is delayed until its

[17]Only the order in which these models (top-down, left-corner, and bottom-up) build the non-terminal symbols is different, while the terminal symbols are still generated in a left-to-right manner in all variants.

[18]For the example in Fig. 3.8, the stack configuration at the time of the verb generation varies only slightly, although the configurations encountered during the history of the full generation can vary considerably in the kinds of actions they predict.

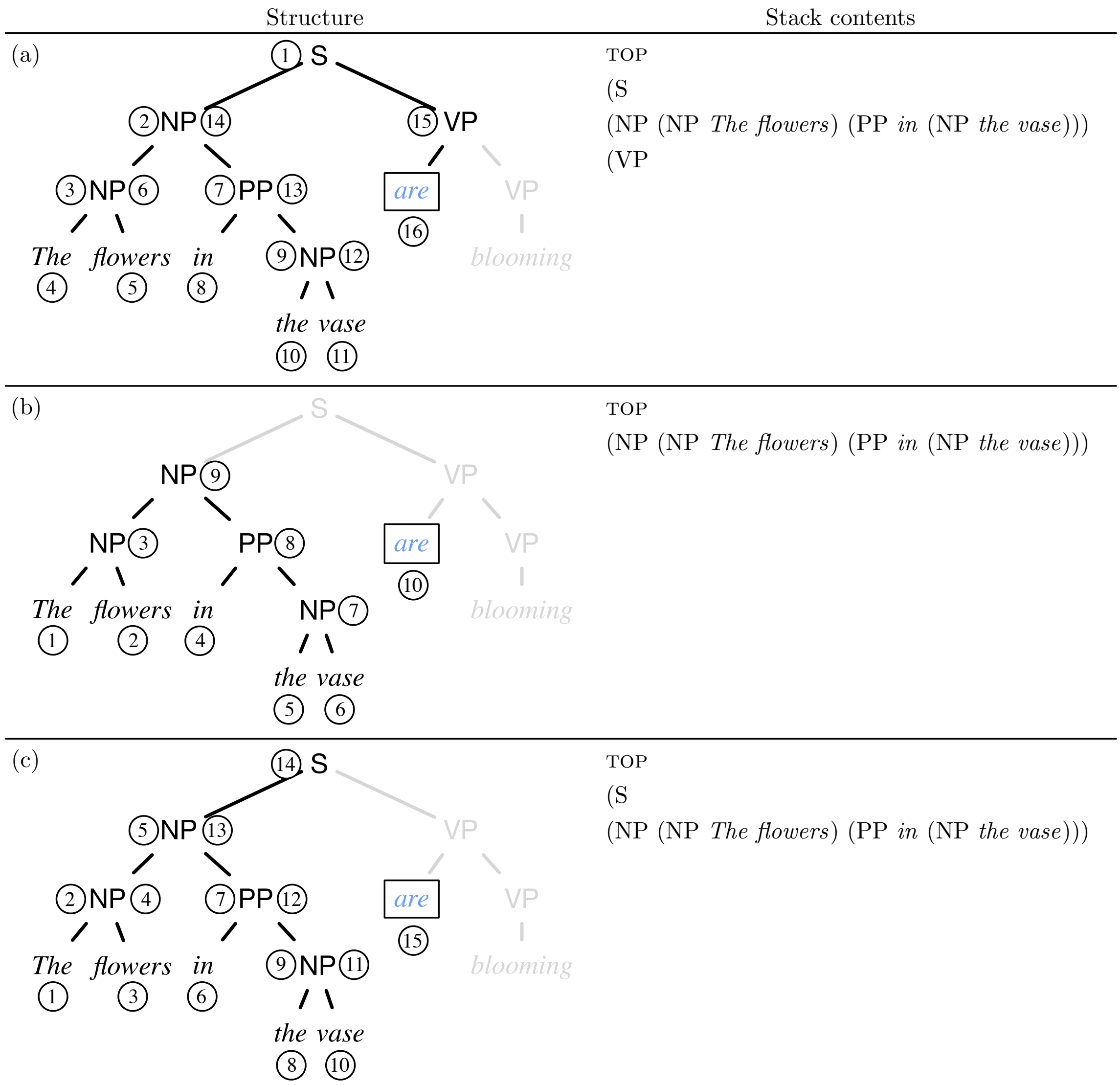


Figure 3.8: The (a) top-down, (b) bottom-up, and (c) left-corner build order variants, showing in black the structure that exists—as well as the generator's stack contents—when the verb "*are*" is generated during the derivation of the sentence "*The flowers in the vase are blooming*". Structure in grey indicates material that will be generated subsequent to this. Circled numbers indicate the time when the corresponding structure/word is constructed. In (a) and (c), non-terminals are generated by a matched pair of NT and REDUCE operations, whereas in (b), they are introduced by a single complex REDUCE operation (§3.6.1).

constituents are complete; this means that the generation of a child node cannot condition on the label of its parent node, which top-down RNNGs are able to do (Fig. 3.8a).

In n-ary branching trees, bottom-up completion of constituents requires a procedure for determining how many of the most recent elements on the stack should be children of

Input: *The fox eats worms*

	Stack	Action
0		GEN(*The*)
1	*The*	GEN(*fox*)
2	*The* | *fox*	REDUCE(NP,2)
3	(NP *The fox*)	GEN(*eats*)
4	(NP *The fox*) | *eats*	GEN(*worms*)
5	(NP *The fox*) | *eats* | *worms*	REDUCE(NP,1)
6	(NP *The fox*) | *eats* | (NP *worms*)	REDUCE(VP,2)
7	(NP *The fox*) | (VP *eats* (NP *worms*))	REDUCE(S,2)
8	(S (NP *The fox*) (VP *eats* (NP *worms*))	STOP

Figure 3.9: Example derivation for bottom-up traversal. ' | ' indicates separate elements on the stack. The "$\text{REDUCE}(\text{X}, n)$" action takes the top n elements on the stack, and creates a new constituent of type X using the composition function.

the node that is being constructed.[19] Conceptually, rather than having a single REDUCE operation as we have before with top-down RNNGs, we have a complex $\text{REDUCE}(\text{X}, n)$ operation that must determine the type of the constituent (*i.e.,* X) as well as the number of children (*i.e.,* n). Concretely, in step 5 of Fig. 3.9, the newly formed NP constituent only covers the terminal "*worms*", and neither the unattached terminal "*eats*" nor the constituent "(NP *The fox*)" is part of this new noun phrase. We implement this decision using a stick-breaking construction—using the stack LSTM encoding, a single-layer feedforward network, and a logistic output layer—which decides whether the top element on the stack should be the leftmost child of the new constituent (*i.e.,* whether or not the new constituent is complete after popping the current topmost stack element), as illustrated in Fig. 3.10. If not, the process is then repeated after the topmost stack element is popped. Once the extent of the new non-terminal has been decided, we parameterise the decision of the non-terminal label type; in Fig. 3.10, this is an "NP". A second difference to top-down generation is that, when a single constituent remains on the stack, the sentence is not necessarily complete (see step 3 of Fig. 3.9 for an example of where this happens). We thus introduce an explicit STOP action (step 8, Fig. 3.9) indicating the tree is complete, which is assigned a non-zero probability mass only when the stack has a single composed phrase.

[19] This mechanism is not necessary with strictly binary branching trees, since each new non-terminal always consists of the two children at the topmost of the stack.

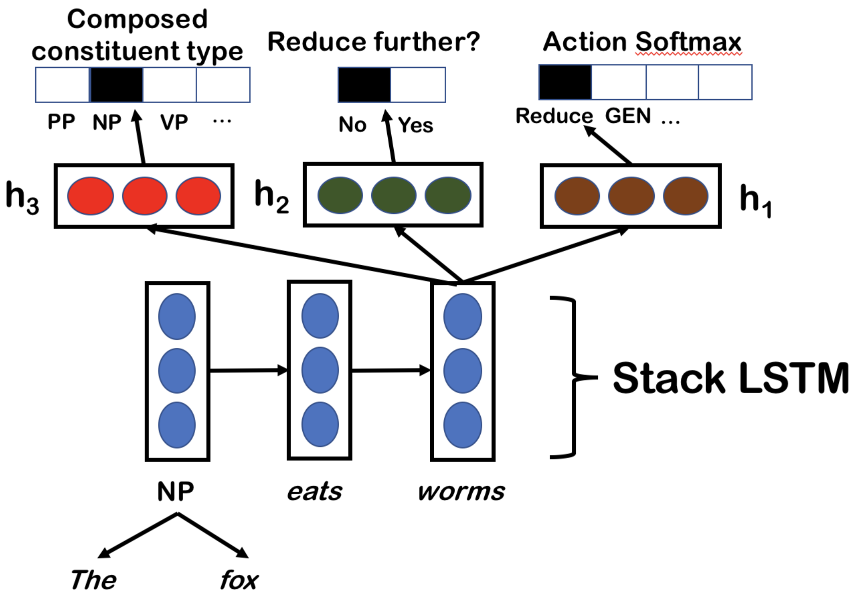


Figure 3.10: Model architecture to determine the type and the extent of the span for new constituents during a REDUCE action for bottom-up generation.

3.6.2 Left-Corner Traversal

Left-corner traversals combine some aspects of top-down and bottom-up processing. As illustrated in Fig. 3.11, this works by first generating the leftmost terminal of the tree, "*The*" (step 0), before proceeding bottom-up to predict its parent "NP" (step 1), and then top-down to predict the rest of its children (step 2). A REDUCE action similarly calls the composition operator once the phrase is complete (*e.g.,* step 3). The complete constituent "(NP *The fox*)" is the leftmost child of its parent node, thus an "NT_SW(S)" action is done next (step 4).

Concretely, the "NT_SW(X)" action is reminiscent of the "NT(X)" action from the top-down generator; both introduce an open non-terminal node and must be matched later by a corresponding REDUCE operation. Nevertheless, the "NT_SW(X)" action additionally swaps the two topmost elements at the top of the stack. This is necessary because the parent non-terminal node is not built until after its leftmost child has been constructed. In step 1 of Fig. 3.11, with a single element "*The*" on the stack, the action "NT_SW(NP)" adds the open non-terminal symbol "NP" to become the topmost stack element, but after applying the swap operator, the stack now contains "(NP | *The*" (step 2).

Input: *The fox eats worms*

	Stack	Action
0		GEN(*The*)
1	*The*	NT_SW(NP)
2	(NP $\mid$ *The*	GEN(*fox*)
3	(NP $\mid$ *The* $\mid$ *fox*	REDUCE
4	(NP *The fox*)	NT_SW(S)
5	(S $\mid$ (NP *The fox*)	GEN(*eats*)
6	(S $\mid$ (NP *The fox*) $\mid$ *eats*	NT_SW(VP)
7	(S $\mid$ (NP *The fox*) $\mid$ (VP $\mid$ *eats*	GEN(*worms*)
8	(S $\mid$ (NP *The fox*) $\mid$ (VP $\mid$ *eats* $\mid$ *worms*	NT_SW(NP)
9	(S $\mid$ (NP *The fox*) $\mid$ (VP $\mid$ *eats* $\mid$ (NP $\mid$ *worms*	REDUCE
10	(S $\mid$ (NP *The fox*) $\mid$ (VP $\mid$ *eats* $\mid$ (NP *worms*)	REDUCE
11	(S $\mid$ (NP *The fox*) $\mid$ (VP *eats* (NP *worms*))	REDUCE
12	(S (NP *The fox*) (VP *eats* (NP *worms*)))	N/A

Figure 3.11: Example derivation for left-corner traversal. Each "NT_SW(X)" action adds the open non-terminal symbol "(X" onto the stack, followed by a deterministic "**swap**" operator that swaps the top two elements on the stack.

	Avg. stack depth	**Ppl.**
TD	12.29	**94.90**
LC	11.45	95.86
BU	7.41	96.53

Table 3.4: Average stack depth and validation set perplexity for top-down (TD), left-corner (LC), and bottom-up (BU) RNNGs.

3.6.3 Experiments

We optimise the hyper-parameters of each RNNG variant using grid searches based on validation set perplexity. Table 3.4 summarises the average stack depths and perplexities[20] on the Linzen et al. (2016) validation set. We evaluate each of the variants in terms of number agreement accuracy as evidence of the model's suitability to model long-range syntactic dependencies in English; we present the results in Table 3.5. To account for randomness in training, we report the error rate summary statistics of ten runs from different random seeds.

[20] Unlike in §3.5.2, all variants (top-down, bottom-up, and left-corner RNNGs) explored here are syntactic language models that provide an estimate of $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. Hence, we compare the models' perplexity based on $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, where $\mathbf{y}$ is the "silver-grade" tree on the Linzen et al. (2016) dataset, without marginalising over $\mathbf{y}$.

	Avg.(±sdev)/min/max		
	n=2	**n=3**	**n=4**
LM	5.8(±0.2)/5.5/6.0	9.6(±0.7)/8.8/10.1	14.1(±1.2)/13.0/15.3
TD	5.5(±0.4)/4.9/5.8	**7.8**(±0.6)/7.4/8.0	**8.9**(±1.1)/7.9/9.8
LC	**5.4**(±0.3)/5.2/5.5	8.2(±0.4)/7.9/8.7	9.9(±1.3)/8.8/11.5
BU	5.7(±0.3) 5.5/5.8	8.5(±0.7)/8.0/9.3	9.7(±1.1)/9.0/11.3

Table 3.5: Number agreement error rates (**lower is better**) for top-down (TD), left-corner (LC), and bottom-up (BU) RNNGs, broken down by the number of attractors. LM indicates the best sequential language model without syntax baseline (§3.3). We report the mean, standard deviation, and minimum/maximum of 10 different random seeds of each model.

Discussion. In Table 3.5, we focus on the most difficult number agreement with multiple attractors, corresponding to cases with two, three, and four attractors. Notably, all three RNNG variants consistently outperform the sequential LSTM without syntax baseline for these cases. Nevertheless, the top-down variant outperforms both left-corner and bottom-up strategies for the challenging cases with three or more attractors, suggesting that the top-down strategy is most appropriately biased to model difficult number agreement dependencies in English. We run an approximate randomisation test by stratifying the output and permuting within each stratum (Yeh, 2000) and find that, for four attractors, the performance difference between the top-down RNNG and the others is statistically significant ($p < 0.05$).

The success of the top-down traversal for number-agreement is consistent with a classical view in parsing; this view argues that top-down parsing is the most human-like parsing strategy since it is the most *anticipatory*. Only anticipatory representations, it is argued, could explain the rapid, incremental processing that humans seem to exhibit (Marslen-Wilson, 1973; Tanenhaus et al., 1995); this line of thinking similarly motivates Charniak (2010), among others. While most work in this domain has been concerned with the parsing problem, our findings suggest that the anticipatory mechanisms of top-down traversals are also beneficial for capturing long-range syntactic dependencies in language modeling. We note that our results are achieved using models that, in theory, are able to condition on the entire derivation history, whereas earlier work in sentence processing has focused on cognitive memory considerations, such as the memory-bounded model of Schuler et al. (2010). While our finding suggests that top-down predictions help choose correctly-agreeing verb forms for the most difficult cases with multiple attractors, a more general statement regarding the best syntactic generation strategy remains within the realm of future work.

3.7 Limitations

We remark on four existing limitations of this work.

- Despite a long and venerable history in cognitive science, number agreement comprises only one type of syntax-sensitive dependencies; model performance at number agreement may not necessarily be indicative of their performance at other types of important syntax-sensitive dependencies, such as reflexive anaphora and negative polarity items. We shall experiment with a more comprehensive set of syntactic evaluations in §4.
- While we conducted our experiments on LSTM-based language models, recent work has mostly used Transformer-based architectures (Vaswani et al., 2017), which can outperform LSTMs on various benchmarks, at least when they are trained on vast amounts of data. Goldberg (2019) found that BERT masked language models, which are based on Transformers, can master syntax-sensitive dependencies like number agreement to a much better extent than previous LSTM-based models. In §5, we explore how Transformer-based architectures can also benefit from more explicit syntactic inductive biases.
- While our findings suggest that top-down traversals outperform their left-corner and bottom-up counterparts in capturing complex number agreement dependencies, these findings are potentially dependent on the specific parameterisation that we used. Indeed, it is conceivable that other parameterisations of left-corner and bottom-up traversals can outperform top-down ones. Furthermore, our findings on the superior performance of top-down traversals may not necessarily generalise to other grammar formalisms. In this work, we have also not explored other traversal strategies beyond pre-defined top-down, left-corner, and bottom-up traversals; one avenue for future work is to *automatically learn* the best traversal order from data, instead of relying on pre-defined traversal strategies.
- The question of how exactly human language learners can capture complex syntactic dependencies like number agreement with multiple attractors, while mostly being exposed to much simpler linguistic stimuli (*i.e.,* poverty of the stimulus, §3.2), remains a subject of ongoing debate in cognitive science. Our findings suggest that, under the control condition (Fig. 3.2), LSTM learners without explicit compositional inductive biases hardly perform better than the majority-class baseline at number agreement with four attractors, although these findings are not necessarily generalisable to human language acquisition for at least two reasons. First, there is a large domain mismatch in the linguistic

stimuli that our models are exposed to, compared to human language learners: Our LSTM models are trained on Wikipedia news articles—a very different domain from the (much simpler) linguistic input that human children are exposed to. Second, the modalities of the linguistic input are also substantially different: Our LSTM models operate on word sequences that have clear segmentations and word boundaries, whereas human language learners have access to linguistic input in the form of raw audio of utterances. We expect the human language acquisition problem to be much harder than our experimental setup would otherwise suggest, because human language learners need to additionally infer how to segment discrete individual words out of the continuous signal that forms the acoustic stream, amongst other important challenges.

3.8 Conclusion

Given enough capacity, LSTMs trained on the language modeling objective are apparently able to master syntax-sensitive dependencies, as evidenced by their accurate number agreement predictions, even for the most difficult cases with multiple attractors. Despite this strong performance, we find that LSTMs succeed in the most challenging number agreement cases with multiple attractors—as the task is defined in the existing benchmark of Linzen et al. (2016)—partially by exploiting a sequential heuristic that memorises the number of the first noun, *without* necessarily capturing the underlying syntactic structure. In contrast, we find that explicit modelling of syntactic structures can substantially improve the language model's ability to capture long-range syntactic dependencies with multiple attractors. To that end, recurrent neural network grammars (RNNGs), which jointly model the probability of strings and phrase-structure trees—and additionally employ recursive compositions—substantially outperform both LSTMs and the Choe and Charniak (2016) syntactic language model that lacks explicit composition. Unlike sequential LSTMs without syntax, RNNGs' number agreement performance is also far more robust to sequential perturbations. All in all, our findings highlight the importance of hierarchical inductive biases for modelling syntactic dependencies. Lastly, we explore the possibility that how the structure is built also affects number agreement performance. Through novel extensions to RNNGs that enable the use of left-corner and bottom-up generation strategies, we discover that this is indeed the case: The three RNNG variants have different generalisation properties for number agreement, with the top-down generation strategy performing best for cases with multiple attractors.

Chapter 4

Syntactic Structure Distillation Training for Left-to-Right Language Models

Chapter Abstract

In the last chapter, we have demonstrated that, at least on small amounts of data, RNNGs that explicitly model hierarchical syntactic structures are able to capture long-range syntactic dependencies much more successfully than their sequential LSTM counterparts. In practice, however, it is infeasible to train RNNGs on large amounts of data; we attribute this limitation to the difficulty of manually batching operations within tree-structured neural architectures like RNNGs. Given the importance of data scale for recent language modelling success (Liu et al., 2019b; Brown et al., 2020), we ask in this chapter: Can we get the best of both worlds? In other words, can we design a language model that not only can be trained on vast amounts of data, but also benefits from syntactic biases at the same time? And would we continue to observe the benefits of syntactic inductive biases, even at larger data scales? In this chapter, we answer these questions by devising a novel knowledge distillation (KD) strategy for injecting syntactic inductive biases into sequential LSTMs; we then demonstrate that the resulting distilled syntax-aware (DSA) LSTM performs well on a challenging and diverse benchmark of syntactic competence. All in all, our findings demonstrate the continued benefits of syntactic inductive biases, even for scalable neural architectures that learn from larger amounts of data.

The material in this chapter is adapted from Kuncoro et al. (2019).

4.1 Introduction

In Ch. 3, we have demonstrated that RNNGs—with their explicit modelling of hierarchical syntactic structures—substantially outperform sequential LSTM language models on a subject-verb agreement measure of syntactic competence. Beyond number agreement success, recent work has arrived at similar conclusions, and independently demonstrated the superior performance of RNNGs on a diverse set of syntactic competence tasks (Wilcox et al., 2019; Futrell et al., 2019; Hu et al., 2020). Furthermore, contemporaneous work has highlighted other benefits of modelling syntactic structures that extend above and beyond improved syntactic competence. Indeed, on small amounts of data, RNNGs have *also* been shown to outperform comparable LSTMs in terms of language modelling perplexity (Kim et al., 2019), parse reranking accuracy (Kuncoro et al., 2017; Fried et al., 2017), and even correlation with human brain encephalography signals (Hale et al., 2018). Nevertheless, these advantages come at the expense of significantly increased computation costs: On specialised hardwares like GPUs, RNNGs are more than ten times slower to train than sequential LSTM language models with comparable numbers of parameters (§4.2).

Two natural questions, therefore, are the following: Can we get the best of both worlds, and design neural language models that not only can be trained on larger amounts of data, but also simultaneously benefit from stronger syntactic biases? And given the remarkable recent success of language models that lack explicit modelling of hierarchical syntactic structures—but are nevertheless trained on large amounts of data (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020, *inter alia*)—are there *still* any benefits from more explicit modelling of hierarchical syntactic structures, even for such models that can learn from larger amounts of textual data? In this chapter, we answer these questions by introducing an efficient knowledge distillation (Bucilă et al., 2006; Hinton et al., 2015) procedure that transfers knowledge from an RNNG *teacher* model (trained on a smaller dataset due to its lack of scalability), into a scalable LSTM *student* model that is trained to emulate the RNNG teacher's predictions, thereby creating the distilled syntax-aware LSTM (**DSA-LSTM**). This procedure enables the DSA-LSTM to develop a more structurally sensitive representation of the larger training data it learns from. On a diverse suite of targeted syntactic evaluations (Marvin and Linzen, 2018), we find that the DSA-LSTM model substantially improves over all baselines, yields a new state of the art, and—on certain syntactic constructions—can even outperform a bidirectional BERT model (Devlin et al., 2019; Goldberg, 2019) trained on 30 times more data. All in all, our findings reaffirm the continued benefits of stronger syntactic biases, even for LSTM models that can learn from larger amounts of data.

4.2 Computational Costs of RNNGs

We begin with a brief overview of *why* RNNGs are harder to scale than conventional LSTMs.

On the importance of mini-batching and specialised hardwares in deep learning. For the most part, current deep learning progress relies on long-established backpropagation techniques (Linnainmaa, 1970; Werbos, 1982; Rumelhart et al., 1986) to find model parameters $\hat{\boldsymbol{\theta}}$ that would minimise the loss through gradient descent. Building on the long history of backpropagation, *recent* deep learning progress hinges on the availability of specialised hardwares—such as graphics and tensor processing units (GPUs and TPUs, respectively)—that are well-suited to handle matrix multiplication processes in parallel (Raina et al., 2009; Ciresan et al., 2012; Krizhevsky et al., 2012), which can substantially speed up neural network training. In order to leverage this parallelism, neural models largely operate over a *mini-batch* of inputs that are processed in parallel; within NLP, such mini-batch often comprises n sequences of tokens, where each sequence shares the same length m.[1]

RNNGs are hard to batch. Despite the importance of mini-batching for accelerating neural network training—and hence for training them on large datasets—not all neural architectures are equally straightforward to batch. We first consider models that are based on sequential RNNs (and by extension, sequential LSTMs), which are amenable to mini-batching in a straightforward fashion. Concretely, sequential RNNs model each sequence as a linear and flat sequence of symbols. This property crucially means that the *ordering* of the RNN computation is exactly the same for each sequence in the batch: For every sequence, the model always begins by processing the first input token x_1 to construct the first hidden state $\mathbf{h_1}$, and proceeds by constructing the second hidden state $\mathbf{h_2}$ based on both the current

[1]Given the variable-length nature of sentences and documents, mini-batching in NLP is typically done either by: (i) sampling a contiguous sequence of m tokens (*e.g.,* 512 contiguous word tokens) from a longer sequence, such as a passage within a book or a Wikipedia article, and repeating this sampling procedure n times to construct the mini-batch; or (ii) "padding" shorter sequences of length $< m$ using special pad symbols (and masking out the loss such that the model does not need to predict these special pad tokens), while setting m to be the longest sequence length in the dataset, or alternatively truncating longer sequences of length $> m$.

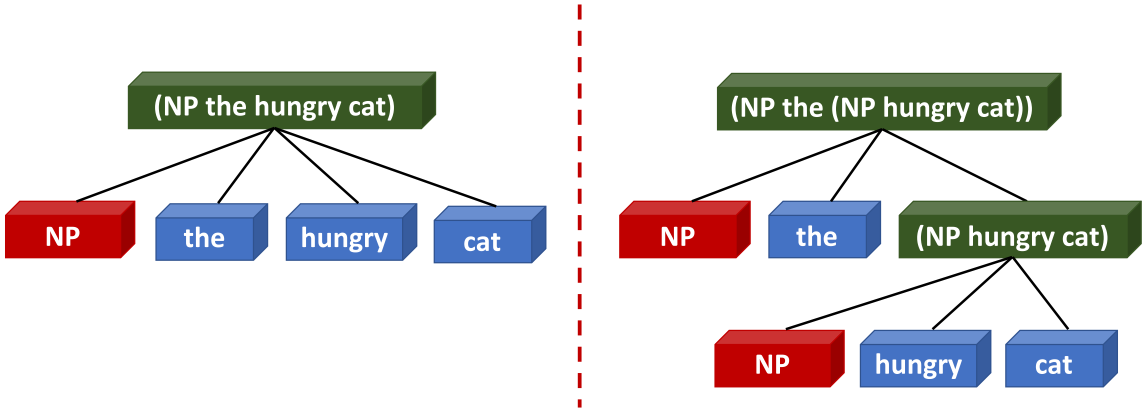


Figure 4.1: Two examples of how RNNGs construct the phrasal representation for the exact same words "*the hungry cat*", albeit in two different ways—each representing a different syntactic tree structure. In the example on the left, the RNNG model would compose "*the hungry cat*" into "(NP *the hungry cat*)" in a single operation, whereas in the example on the right, the RNNG would first compose "*hungry cat*" into "(NP *hungry cat*)", before composing this with "*the*" to obtain a final representation for "(NP *the* (NP *hungry cat*))".

input token x_2 and the previous hidden state $\mathbf{h_1}$, and so forth.[2] Unfortunately, the same easy-to-batch property of RNNs does not extend to RNNGs, since the ordering of the RNNG computation is determined by a syntactic tree structure that is unique to each sequence, hence rendering them much less amenable to manual batching than RNNs. We illustrate this issue in Fig. 4.1, where we provide an example of how RNNGs construct a phrasal representation for the exact same phrase "*The hungry cat*", albeit for two different tree structures. As is clearly evident from Fig. 4.1, even though both phrases feature the exact same words, the *ordering* of RNNG operations is nevertheless unique to each example: Whereas the example on the left directly composes "*the hungry cat*" into the final phrasal representation "(NP *the hungry cat*)", the example on the right works in a two-stage process by first composing "*hungry cat*" into "(NP *hungry cat*)", before doing a second-stage composition with "*the*" in order to form the final representation for "(NP *the* (NP *hungry cat*))". Beyond this simple example, in practice each sentence in the dataset would have substantially different tree structures, which renders manual batching—and by extension accelerating model training with specialised hardwares like GPUs or TPUs—much more challenging for RNNGs.[3]

[2] At training time, Transformers (Vaswani et al., 2017) can benefit from batching and specialised hardwares like GPUs to an even larger extent than RNN-based models. This is because—within each Transformer layer—the processing of each word can be done in parallel. In contrast, RNN processing is sequential in nature: The hidden state $\mathbf{h_i}$ can only be computed once all its preceding hidden states $\mathbf{h_{<i}}$ have been computed.

[3] The difficulty of manually batching RNNGs also extends to other cases where the model architecture is determined by a non-sequential structure that is unique to each sequence, such as Tree LSTMs (Tai et al.,

4.3 Approach: Scalable Syntax-Aware Language Modelling through Knowledge Distillation

In this section, we propose a knowledge distillation approach in response to our overarching question: Given the trade-off between explicit compositional operations and model scalability, how can we get the best of both worlds, and design language models that not only benefit from stronger syntactic biases, but also can be easily scalable at the same time?

4.3.1 Background: Cross-Entropy Language Modelling Loss

Before describing the proposed knowledge distillation approach, we begin by demonstrating how the standard language modelling cross-entropy loss is equivalent (up to a constant) to minimising the Kullback-Leibler (KL) divergence between the true data-generating process $p^{\star}(\mathbf{x})$ and a language model's distribution $q_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta}$ denotes the set of all language model parameters. This background material sets the stage for the knowledge distillation approach in §4.3.2, which shall similarly be described through the lens of KL divergence.

Let $\hat{\boldsymbol{\theta}}_{\text{LM}}$ be the set of model parameters that would minimise[4] the following KL divergence, which we henceforth denote as $D_{\text{KL}}(\cdot||\cdot)$:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{LM}} &= \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p^{\star}(\mathbf{x}) \,||\, q_{\boldsymbol{\theta}}(\mathbf{x})\right), \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{\mathbf{x} \in \Sigma^{\star}} p^{\star}(\mathbf{x}) \, \log p^{\star}(\mathbf{x}) - p^{\star}(\mathbf{x}) \, \log q_{\boldsymbol{\theta}}(\mathbf{x}), \quad \text{(definition of KL)} \\
&= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x} \in \Sigma^{\star}} p^{\star}(\mathbf{x}) \, \log q_{\boldsymbol{\theta}}(\mathbf{x}). \qquad \text{(removing } \boldsymbol{\theta}\text{-independent terms)}
\end{aligned} \tag{4.1}$$

Here Σ denotes the set of all word types in the vocabulary, and $\Sigma^{\star}$ denotes the set of all possible sentences in natural language. In practice, Eq. 4.1 cannot be computed exactly for two reasons: (i) The true data-generating process $p^{\star}(\mathbf{x})$ is inaccessible, and (ii) Eq. 4.1 involves an intractable summation over the set of all possible sentences $\Sigma^{\star}$, where each sentence can have an arbitrary (and even theoretically infinite) length. Hence, in practice, we approximate Eq. 4.1 through a (finite) dataset of sentences $D = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(|D|)}\right]$ that is assumed to be sampled from the data-generating process $p^{\star}(\mathbf{x})$ in an independent

2015), stack LSTMs (Dyer et al., 2015), and neural module networks (Andreas et al., 2016a,b).

[4]In practice, due to the non-convex nature of neural network parameter optimisation, we are not guaranteed to find a global minimum for $\hat{\boldsymbol{\theta}}_{\text{LM}}$ that is guaranteed to have a lower loss than all other possible values of $\boldsymbol{\theta}$.

and identically distributed (i.i.d.) fashion. We then characterise the empirical distribution as $p(\mathbf{x}^{(n)}) = \frac{1}{|D|} \;\forall\, \mathbf{x}^{(n)} \in D$, and otherwise assign a probability of 0 to all sentences that are not in the dataset D. With these assumptions in mind, we approximate Eq. 4.1 as follows:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{LM}} &\approx \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D|} \sum_{\mathbf{x}^{(n)} \in D} \log q_{\boldsymbol{\theta}}(\mathbf{x}^{(n)}), \\
&= \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D|} \sum_{\mathbf{x}^{(n)} \in D} \sum_{i=1}^{|\mathbf{x}^{(n)}|} \log q_{\boldsymbol{\theta}}(x_i^{(n)} \mid \mathbf{x}_{<i}^{(n)}), \quad \text{(chain rule decomposition)}
\end{aligned} \tag{4.2}
$$

which reduces to the standard, left-to-right language modelling cross-entropy loss.

4.3.2 Knowledge Distillation (KD)

The goal of KD—as originally proposed by Hinton et al. (2015)—is to find a set of student model parameters $\hat{\boldsymbol{\theta}}_{\text{KD}}$ that would minimise the KL divergence between a teacher model's marginal probability, denoted as $t_{\boldsymbol{\psi}}(\mathbf{x})$, and a student model that estimates $q_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ denote the set of teacher and student model parameters, respectively.[5] More formally:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{KD}} &= \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(t_{\boldsymbol{\psi}}(\mathbf{x}) \;||\; q_{\boldsymbol{\theta}}(\mathbf{x})\right), \\
&= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x} \in \Sigma^*} t_{\boldsymbol{\psi}}(\mathbf{x}) \log q_{\boldsymbol{\theta}}(\mathbf{x}). \quad \text{(expand KL \& remove } \boldsymbol{\theta}\text{-independent terms)}
\end{aligned} \tag{4.3}
$$

We remark that Eq. 4.3 shares some similarity with Eq. 4.1, albeit with one key difference: In the case of KD, the "teacher distribution" does not come from the true data-generating process $p^\star(\mathbf{x})$—which is inaccessible to us—but rather is derived from another, teacher model $t_{\boldsymbol{\psi}}(\mathbf{x})$, where the model parameters $\boldsymbol{\psi}$ are indeed accessible. Nevertheless, Eq. 4.3 still involves an intractable summation over the set of all sentences in natural language, Σ^*. As exact computation of Eq. 4.3 is intractable, we employ a Monte Carlo approximation, where we construct a dataset D' by sampling sentences from the trained teacher model. Formally, let $D' = \left[\mathbf{x}'^{(1)}, \cdots, \mathbf{x}'^{(|D'|)}\right]$, where each sentence $\mathbf{x}'^{(n)} \in D'$ was sampled in an i.i.d. fashion according to $t_{\boldsymbol{\psi}}(\mathbf{x})$. Assuming that the teacher model is a left-to-right neural LM that works at the sentence-level (as is the case in this work), each sentence $\mathbf{x}'^{(n)} \in D'$ can be obtained through a simple ancestral sampling procedure, where we

[5]This commonly-used formulation of KD means that training the student model requires the existence an already-trained teacher model (as parameterised by $\boldsymbol{\psi}$). Hence, this means that KD training is typically done in a two-stage fashion, where we train the teacher model first, before training the student model.

sequentially sample a token from the teacher model's categorical softmax distribution. Formally, $\mathbf{x}'^{(n)} = \left[x_1'^{(n)}, \cdots, x_{|\mathbf{x}'^{(n)}|}'^{(n)}\right]$, where each token $x_i'^{(n)}$ in the sentence $\mathbf{x}'^{(n)}$ is sampled according to the teacher model's categorical softmax distribution for predicting the next word, *i.e.,* $x_i'^{(n)} \sim t_{\boldsymbol{\psi}}(\cdot \mid \mathbf{x}_{<i}'^{(n)})$, where $x_i'^{(n)} \in \Sigma$. We continue sampling the next token $x_{i+1}'^{(n)}$ until an end-of-sentence symbol—which is part of the vocabulary Σ—is sampled.

Since D' was obtained through an unbiased ancestral sampling procedure from $t_{\boldsymbol{\psi}}(\mathbf{x})$, we define a Monte Carlo approximation of the KD objective (Eq. 4.3) as follows:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{KD}} &= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x}\in\Sigma^*} t_{\boldsymbol{\psi}}(\mathbf{x}) \log q_{\boldsymbol{\theta}}(\mathbf{x}) && \text{(Eq. 4.3, reproduced)} \\
&\approx \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D'|} \sum_{\mathbf{x}'^{(n)}\in D'} \log q_{\boldsymbol{\theta}}(\mathbf{x}'^{(n)}) \\
&= \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D'|} \sum_{\mathbf{x}'^{(n)}\in D'} \sum_{i=1}^{|\mathbf{x}'^{(n)}|} \log q_{\boldsymbol{\theta}}(x_i'^{(n)} \mid \mathbf{x}_{<i}'^{(n)}). && (4.4)
\end{aligned}$$

Note that—after applying the approximation—the KD objective in Eq. 4.4 has a similar form to the standard LM objective in Eq. 4.2. One key difference is that we train the KD student model (with a standard cross-entropy loss) on the sampled sentences from the teacher model $\mathbf{x}'^{(n)} \in D'$, as opposed to the ground-truth sentences on the training data $\mathbf{x}^{(n)} \in D$.

Preliminary experiments. We begin by asking: How effective is our Monte Carlo approximation (Eq. 4.4) to the KD objective? To answer this question, we conduct the following preliminary experiment. First, we train an LSTM language model trained on the standard language modelling cross-entropy loss (Eq. 4.2). Concretely, we train this model on the preprocessed Wikipedia dataset of Gulordava et al. (2018), henceforth denoted as D; this dataset consists of 3 million training sentences. We then use this trained LSTM model as the *teacher* model, and construct a dataset D' by sampling the same number of sentences from the trained teacher model (*i.e.,* $|D'| = |D|$). We then train a separate, *student* model using the approximated KD objective in Eq. 4.4, which employs a cross-entropy loss on the sampled sentences D'. To enable fair comparison, we use the exact same model architecture to train both the teacher and student models: We use a two-layer LSTM network with 650 hidden units in each layer, and further apply the same learning rate and other hyper-parameters to both models. As both models are trained with the exact same cross-entropy loss, the only difference is whether the model is trained on the true Wikipedia sentences D (as is the case for the teacher model), or on the sampled sentences D' (as is the case for the student model).

Preliminary findings. If the approximated KD loss in Eq. 4.4 is an effective way of transferring knowledge from the teacher model to the student model, we would expect the student model to achieve a similar validation set perplexity as the teacher model.[6] Alas, we do not find this to be the case: Even though both the teacher and student models share the exact same architecture, hyper-parameters, and training data size, the teacher model has a much lower (and thus better) test set perplexity of 53.73, while the student model has a much higher test perplexity of $> 1,000$. We attribute this performance gap to high variance due to the enormity of the sample space (*i.e.,* the infinite set of all possible strings in language, Σ^*), and the discrete nature of the sampling procedure. Due to computational costs, we do not attempt increasing the size of the sampled sentences D' to more than 3 million sentences.

4.3.3 Word-level KD Approximation

Given the poor performance of the Monte Carlo approximation to the KD objective (Eq. 4.4), we instead choose to minimise the KL divergence between the teacher and student models at the *local word-level*, as follows:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{KD-word}} &= \arg\min_{\boldsymbol{\theta}} \sum_{\mathbf{x}\in\Sigma^\star} p^\star(\mathbf{x}) \sum_{i=1}^{|\mathbf{x}|} D_{\text{KL}}\left(t_{\boldsymbol{\psi}}(x_i \,|\, \mathbf{x}_{<i}) \;||\; q_{\boldsymbol{\theta}}(x_i \,|\, \mathbf{x}_{<i})\right) \\
&= \arg\min_{\boldsymbol{\theta}} - \sum_{\mathbf{x}\in\Sigma^\star} p^\star(\mathbf{x}) \sum_{i=1}^{|\mathbf{x}|} \sum_{w\in\Sigma} t_{\boldsymbol{\psi}}(w \,|\, \mathbf{x}_{<i}) \, \log q_{\boldsymbol{\theta}}(w \,|\, \mathbf{x}_{<i}) \qquad (4.5) \\
&\approx \arg\min_{\boldsymbol{\theta}} - \frac{1}{|D|} \sum_{\mathbf{x}^{(n)}\in D} \underbrace{\sum_{i=1}^{|\mathbf{x}^{(n)}|} \sum_{w\in\Sigma} t_{\boldsymbol{\psi}}(w \,|\, \mathbf{x}^{(n)}_{<i}) \, \log q_{\boldsymbol{\theta}}(w \,|\, \mathbf{x}^{(n)}_{<i})}_{\ell_{\text{KD-word}}\left(\mathbf{x}^{(n)};\boldsymbol{\theta}\right)} . \qquad (4.6)
\end{aligned}$$

Here Eq. 4.5 follows from expanding the KL term and removing terms that are independent of $\boldsymbol{\theta}$, while Eq. 4.6 follows from the same i.i.d. assumption as in Eq. 4.2, where we assume that each sentence $\mathbf{x}^{(n)} \in D$ is sampled from the true data-generating process $p^\star(\mathbf{x})$ in an i.i.d. fashion. Here $t_{\boldsymbol{\psi}}(w \,|\, \mathbf{x}^{(n)}_{<i})$ and $q_{\boldsymbol{\theta}}(w \,|\, \mathbf{x}^{(n)}_{<i})$ respectively parameterise the (marginal) probability of generating the next-word continuation $w \in \Sigma$, conditional on the "ground-truth" prefix $\mathbf{x}^{(n)}_{<i}$, under the teacher and student models. In earlier work, the same local

[6] We remark that the student model is not directly trained to minimise cross-entropy with respect to the gold sentences in D, but rather aims to minimise the KL divergence with respect to the teacher distribution through its samples D' (Eqs. 4.3 and 4.4). Nevertheless, perplexity on a held-out set is still the primary metric for language model evaluation. Hence, we evaluate the student model based on its perplexity on a held-out validation split of the true dataset D, and *not* on how well the student model can fit the sampled sentences D'.

word-level approximation to the KD objective has been shown to work surprisingly well for sequence modelling tasks, in the context of both neural machine translation[7] (Kim and Rush, 2016) and language modelling (Furlanello et al., 2018, Born-Again Networks) tasks.

4.3.4 Interpolated Word-level KD Objective

The RNNG teacher is an expert on syntax by virtue of its explicit modelling of hierarchical syntactic structures, although it lacks the opportunity to acquire the relevant semantic and common-sense knowledge from large textual corpora. Hence, our goal here is to design a student model that combines the best of both worlds: We want the more scalable student model to not only inherit the strong syntactic inductive biases of RNNGs, but *also* leverage the rich training signal afforded by data scale. To that end, we interpolate the word-level KD loss in Eq. 4.6—enabling the student model to replicate the RNNG teacher's strong syntactic generalisation—with the standard language modelling loss, which enables the student model to learn about the relevant semantics and world knowledge from textual data:[8]

$$\hat{\boldsymbol{\theta}}_{\alpha\text{-int}} = \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D|} \sum_{\mathbf{x}^{(n)} \in D} \left[\alpha \underbrace{\ell_{\text{KD-word}}\left(\mathbf{x}^{(n)}; \boldsymbol{\theta}\right)}_{\text{Word KD loss (Eq. 4.6)}} + (1-\alpha) \underbrace{\sum_{i=1}^{|\mathbf{x}^{(n)}|} \log q_{\boldsymbol{\theta}}(x_i^{(n)} \mid \mathbf{x}_{<i}^{(n)})}_{\text{LM loss (Eq. 4.2)}} \right], \tag{4.7}$$

where α is the interpolation coefficient. We later illustrate the effect of this interpolation in Fig. 4.2. Unless otherwise stated, we set $\alpha = 0.5$, which assigns equal weight to the KD loss and the LM loss. We henceforth use this interpolated objective to train the DSA-LSTM.

Connection with label smoothing. In some sense, the interpolated objective (Eq. 4.7) can be understood as a form of label smoothing (Szegedy et al., 2016; Pereyra et al., 2017). A key difference is that we use the softmax distribution of the RNNG as the smoothing distribution, whereas standard label smoothing techniques often employ the uniform distribution.

[7]While Kim and Rush (2016) proposed a technique for approximating the sequence-level KD objective (Eq. 4.3) through beam search for machine translation (a *conditional* language generation problem), the same technique is not directly applicable to language modelling, which is an *unconditional* generation problem.

[8]Prior work has demonstrated the effectiveness of the language modelling loss on large textual corpora for acquiring semantics (primarily distributional semantics) and world knowledge, as evidenced by the success of pre-trained LMs like GPT-3 (Brown et al., 2020). While the word-level KD objective in Eq. 4.6 is effective for transferring syntactic knowledge, it nevertheless differs from the language modelling loss, as the word-level KD loss $\ell_{\text{KD-word}}\left(\mathbf{x}^{(n)}; \boldsymbol{\theta}\right)$ does not depend on the true identity of the next word $x_i^{(n)}$ (see Eq. 4.6).

4.3.5 Intuition

In Fig. 4.2, we provide an intuition about *why* an interpolation of the KD loss and the LM loss could inject syntactic inductive biases into a sequential (albeit more scalable) DSA-LSTM. Concretely, we consider the interpolated target with $\alpha = 0.5$ for a prefix "*The* ***parts*** *of the river valley*", where the correct continuation is the plural verb "***have***" in order to agree with the subject "***parts***", despite the presence of two adjacent attractor nouns "*river*" and "*valley*"; a detailed overview of this subject-verb agreement task is provided in §3.2.

Under the standard language modelling loss (Fig. 4.2, middle), the student model incurs a zero loss *iff* it assigns a probability of 1.0 to the correct next word (*i.e.,* the plural verb "***have***"), whilst assigning a probability of zero to every other word in the vocabulary. Hence, under this loss, it is only in expectation (*i.e.,* across training contexts) that syntactic regularities are inferred. In other words, the sequential student model—which lacks an explicit inductive bias for modelling syntactic structures—would likely require a large number of examples to infer that the verb should agree with the more distant subject "***parts***", as opposed to the more adjacent two attractor nouns. This problem is further exacerbated by the fact that such syntactically disambiguating examples, where the subject is different from the sequentially most adjacent noun to the verb, rarely occur in naturalistic data (§3.2)

In contrast, the interpolated loss that we use to train the student model (Fig. 4.2, bottom) assigns a minimum probability of 0.5 to the correct label, but crucially contains additional information about the *plausibility* of every alternative based on the RNNG teacher's predictions. Under this objective, the set of plural verbs, such as "***are***" and "***meander***", are assigned relatively high probability mass, because they fit *both* the syntactic and other constraints of the context (*e.g.,* Parts of the river valley often meander). In contrast, we expect singular verbs such as "***has***", "***meanders***", and "***is***" to be assigned a much lower probability mass, as these verbs disagree with the subject "***parts***", and are therefore ungrammatical. Thus, as long as the RNNG teacher makes the accurate syntactic generalisations (and we will show that it largely does in §4.4.4), every training instance—through the predictions of the RNNG teacher—provides the LSTM student with a wealth of information about the underlying syntactic regularities, thereby making it easier for the student to acquire the appropriate syntactic constraints and generalisations in a much more data-efficient fashion.

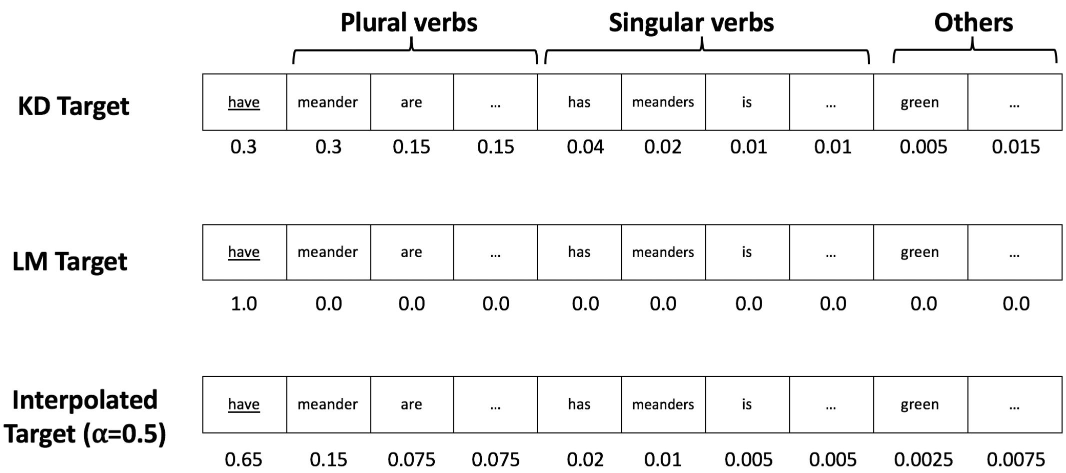


Figure 4.2: Examples of the KD teacher distribution (top), the standard LM one-hot distribution (middle), and the interpolated KD and LM teacher distribution used to train the DSA-LSTM (bottom). In all experiments, we set $\alpha = 0.5$. Here we use the prefix (showing only the terminals) "*The **parts** of the river valley*", where the correct continuation is "***have***" due to the plural subject "***parts***", despite the presence of two attractors (underlined).

4.3.6 Marginal Probability Approximation

The KD objective in Eq. 4.6—and by extension, the interpolated loss in Eq. 4.7—requires computing the teacher model's marginal next-word distribution $t_{\psi}(w \mid \mathbf{x}_{<i}^{(n)})$. As the RNNG teacher models the joint probability of strings and phrase-structure trees $t_{\psi}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$—as opposed to $t_{\psi}(\mathbf{x}^{(n)})$ for non-syntactic LMs that only model the word sequences without the syntactic tree structures—obtaining the RNNG's estimate of $t_{\psi}(w \mid \mathbf{x}_{<i}^{(n)})$ therefore necessitates an expensive marginalisation procedure over the set of all possible partial phrase-structure trees $\mathbf{y}_{<i}^{(n)}$, which is exponential in the input length $|\mathbf{x}_{<i}^{(n)}|$. Formally:

$$
\begin{aligned}
t_{\psi}(w \mid \mathbf{x}_{<i}^{(n)}) &= \sum_{\mathbf{y}_{<i}^{(n)} \in Y(\mathbf{x}_{<i}^{(n)})} t_{\psi}(w, \mathbf{y}_{<i}^{(n)} \mid \mathbf{x}_{<i}^{(n)}) \\
&= \sum_{\mathbf{y}_{<i}^{(n)} \in Y(\mathbf{x}_{<i}^{(n)})} t_{\psi}(\mathbf{y}_{<i}^{(n)} \mid \mathbf{x}_{<i}^{(n)})\; t_{\psi}(w \mid \mathbf{x}_{<i}^{(n)}, \mathbf{y}_{<i}^{(n)}) \quad \text{(chain rule)} \qquad (4.8) \\
&= \mathbb{E}_{t_{\psi}(\mathbf{y}_{<i}^{(n)} \mid \mathbf{x}_{<i}^{(n)})} \left[t_{\psi}(w \mid \mathbf{x}_{<i}^{(n)}, \mathbf{y}_{<i}^{(n)}) \right]. \quad \text{(definition of expectation)} \qquad (4.9)
\end{aligned}
$$

One way to approximate the expectation in Eq. 4.9—which we need to compute for *every single word position* in the whole training set (see the summation over all training sentences $\mathbf{x}^{(n)} \in D$ in Eq. 4.6)—is to use a word-synchronised RNNG beam search procedure (Fried et al., 2017) to obtain a much smaller set of highly probable partial trees $\mathbf{y}_{<i}^{(n)}$, as we did in

§3.5.1. In practice, however, such word-synchronised beam search procedure is extremely expensive to run: Running the beam search procedure on a moderately-sized training set of 80 million words (about 3 million sentences) would require nearly 30 years of CPU time.[9]

For this reason, here we employ a simpler approximation procedure. Let $\hat{\mathbf{y}}^{(n)}_{\text{berk}}(\mathbf{x}^{(n)})$ be the one-best predicted phrase-structure tree for an input sentence $\mathbf{x}^{(n)}$, which we obtain from an off-the-shelf, pre-trained Berkeley parser (Petrov et al., 2006; Petrov and Klein, 2007). Furthermore, let $\hat{\mathbf{y}}^{(n)}_{\text{berk-}<i}(\mathbf{x}^{(n)})$ denote all the non-terminals in $\hat{\mathbf{y}}^{(n)}_{\text{berk}}(\mathbf{x}^{(n)})$ that occur before the i-th token. For the purpose of this approximation, we shall assume that $t_{\boldsymbol{\psi}}(\mathbf{y}^{(n)}_{<i} \mid \mathbf{x}^{(n)}_{<i}) \approx 1$ *iff* $\mathbf{y}^{(n)}_{<i} = \hat{\mathbf{y}}^{(n)}_{\text{berk-}<i}(\mathbf{x}^{(n)})$, and 0 otherwise. We can thus approximate Eq. 4.8 as follows:

$$\begin{aligned} t_{\boldsymbol{\psi}}(w \mid \mathbf{x}^{(n)}_{<i}) &= \sum_{\mathbf{y}^{(n)}_{<i} \in Y(\mathbf{x}^{(n)}_{<i})} t_{\boldsymbol{\psi}}(\mathbf{y}^{(n)}_{<i} \mid \mathbf{x}^{(n)}_{<i})\ t_{\boldsymbol{\psi}}(w \mid \mathbf{x}^{(n)}_{<i}, \mathbf{y}^{(n)}_{<i}) \qquad \text{(Eq. 4.8, reproduced)} \\ &\approx t_{\boldsymbol{\psi}}(w \mid \mathbf{x}^{(n)}_{<i}, \hat{\mathbf{y}}^{(n)}_{\text{berk-}<i}(\mathbf{x}^{(n)})). \end{aligned} \tag{4.10}$$

Further remarks on the approximation. The approximation in Eq. 4.10 is much more efficient, because it relies on the one-best predictions of an off-the-shelf, pre-trained Berkeley parser to quickly obtain a "proposal tree" $\hat{\mathbf{y}}^{(n)}_{\text{berk}}(\mathbf{x}^{(n)})$ for a particular input sentence $\mathbf{x}^{(n)}$.[10] Nevertheless, this approximation has two key shortcomings. First, the syntactic prefix $\hat{\mathbf{y}}^{(n)}_{\text{berk-}<i}(\mathbf{x}^{(n)})$ is derived from a *non-incremental* parser that predicts the phrase-structure tree, conditional on the *entire* input sentence $\mathbf{x}^{(n)}$. In contrast, Eq. 4.8 requires the syntactic prefix $\mathbf{y}^{(n)}_{<i}$ to be computed solely conditional on $\mathbf{x}^{(n)}_{<i}$—without access to yet-unseen words $\mathbf{x}^{(n)}_{\geq i}$. Despite the non-incremental nature of the RNNG teacher's approximated marginal next-word distribution, we remark that the DSA-LSTM student model—the primary model that we are proposing this work—still operates strictly incrementally, and can thus be fairly compared against other incremental language model baselines.[11] Second, Eq. 4.8 requires the distribution over partial phrase-structure trees, $t_{\boldsymbol{\psi}}(\mathbf{y}^{(n)}_{<i} \mid \mathbf{x}^{(n)}_{<i})$, to be parameterised by the same teacher model with parameters $\boldsymbol{\psi}$. Our approximation departs from this requirement,

[9] As the RNNG word-synchronised beam search procedure is difficult to batch (even more so than RNNG training), we observe no improvement in inference speed from running on GPUs (as opposed to CPUs).

[10] For a moderately-sized training set of about 3 million sentences and 80 million words, we use 100 CPU cores and data parallelism to obtain the one-best predicted phrase-structure tree for each sentence. Under this setup, parsing the whole training set with an off-the-shelf, pre-trained Berkeley parser takes takes about a day.

[11] The RNNG teacher's approximated marginal next-word distribution is only used as an auxiliary objective (Eq. 4.7) for the DSA-LSTM student at *training* time. Once the DSA-LSTM student has finished training, the RNNG teacher's predictions are no longer made available to the student, which operates strictly incrementally.

by virtue of relying on predicted phrase-structure trees that come from a separately-trained Berkeley parser that has its own set of model parameters and training data.

4.3.7 Novelty of the Knowledge Distillation Setup

Our approach departs from the predominant view of knowledge distillation[12] primarily as a means of *compressing* knowledge—either from a large teacher model or an ensemble of models, into a more compact student (Ba and Caruana, 2014; Kim and Rush, 2016; Liu et al., 2018, *inter alia*)—in two important ways. First, here the teacher and student models differ in character, and not just in size: We aim to transfer knowledge from an RNNG teacher that models the joint probability of strings and phrase-structure trees through explicit syntax-based *hierarchical* operations, into a student model that only estimates the probability of surface strings through *sequential* operations. This setup thus presents an interesting dynamic: The DSA-LSTM student has to mimic the predictions of the RNNG teacher, which conditions on syntactic annotation to guide recursive compositions, even though the DSA-LSTM itself lacks access to any syntactic annotations or explicit compositions at all.

Second, knowledge distillation thus far has mostly been applied in settings where the teacher and student models are trained on the same data. For scalability reasons, here we can only train the RNNG teacher on a smaller subset of the data; we then run approximate inference to obtain the RNNG's "soft" next-word predictions (Eqs. 4.6 and 4.7) on the rest of the training set. Despite this difference in data size, we hypothesise that the predictions of the RNNG teacher—although they come from a model that is trained on a smaller training set—can nevertheless encourage the more scalable (albeit sequential) DSA-LSTM student to develop a more structurally sensitive representation of the larger dataset it learns from.

4.3.8 Putting It All Together

Having described all components of the proposed approach, we provide an overview of the DSA-LSTM training procedure, which is accomplished by the following three simple steps.

[12]While the term "knowledge distillation" was initially proposed by Hinton et al. (2015) as a means of compressing an ensemble of models to a single and more efficient one, often with a temperature on the softmax, this term has recently been used more broadly to refer to the transfer of knowledge between two models, typically from a larger and more expensive teacher model to a smaller and more efficient student model.

- In the first step, we train the RNNG teacher on as much data as we feasibly can. In practice, due to the high training costs of hierarchical neural architectures (§4.2), we are only able to train the RNNG teacher on a smaller subset of the data (here a 20% subset).
- In the second step, once the RNNG teacher has finished training, we run approximate inference (Eq. 4.10) to compute its next-word distributions on the whole training set.
- In the third and final step, we train a DSA-LSTM student on an interpolation of the standard LM cross-entropy objective (Eq. 4.2) and the word-level KD objective (Eq. 4.6). This interpolated objective (Eq. 4.7) enables the DSA-LSTM to exploit the best of both model scalability (by virtue of learning from the true next word $x_i^{(n)}$) and syntactic inductive biases (by learning from what the RNNG teacher thinks the next word should be, which contains valuable information about the underlying syntactic regularities, Fig. 4.2).

4.4 Experimental Setup and Baselines

Here we describe our experimental setup, and provide a summary of the targeted syntactic evaluations benchmark (§4.4.2), which serves as our primary evaluation instrument. Given the importance of comparing proposed approaches against strong, well-tuned baselines (Melis et al., 2018; Henderson et al., 2018), we conduct a replication of targeted syntactic evaluations of both LSTMs (§4.4.3) and RNNGs (§4.4.4), which shall serve as baselines to our proposed DSA-LSTM approach. In our replication experiments, we strive to achieve more scientific rigour by carefully tuning the hyper-parameters of each model based on a grid search, and account for the variance in results that arises from different random seeds.

4.4.1 Wikipedia Dataset

Following Marvin and Linzen (2018), we use the preprocessed English Wikipedia dataset of Gulordava et al. (2018). With 3 million training sentences and 80 million words, this dataset is approximately 100x larger than the Penn Treebank, and thus constitutes a medium-sized language modelling test bed; we shall experiment with even-larger corpora in Ch. 5.

The Gulordava et al. (2018) Wikipedia dataset is preprocessed with both tokenisation and infrequent word mapping, where word types beyond the top 50,000 most frequent ones are mapped to a special "<UNKNOWN>" word symbol. Whereas the original validation split of this dataset contains more than 300,000 sentences, in order to speed up training, we select a random sample of 10,000 validation sentences as the basis for early stopping and hyper-parameter search; we then evaluate model perplexity on the *full* validation set after training has concluded. Since the targeted syntactic evaluations benchmark is thus far only available in English (§4.4.2), we only use the English Wikipedia for training the model.

4.4.2 Targeted Syntactic Evaluations

As in Ch. 3, here we focus on the language model's ability to acquire a high degree of syntactic competence—which constitutes an important part of human language understanding ability—and leave the extension to other tasks (*i.e.,* where exactly would syntactic inductive biases be advantageous in different types of NLP tasks, above and beyond improved syntactic competence?) to Ch. 5. To that end, we use the targeted syntactic evaluations (Marvin and Linzen, 2018) benchmark as our primary means of language model evaluation. We motivate this choice for two key reasons. First, this benchmark assesses a model's syntactic competence on a wide variety of 15 syntactic constructions, whereas earlier datasets (Linzen et al., 2016; Gulordava et al., 2018, §3.2) put a sole emphasis on subject-verb agreement in the presence of attractors. Hence, targeted syntactic evaluations offer a more *holistic* assessment of language models' syntactic competence. Second, prior work on subject-verb agreement evaluation (Linzen et al., 2016; Gulordava et al., 2018) constructs the test examples from *naturalistic* text. Unfortunately, this approach makes it difficult to find a sufficient number of syntactically challenging examples, which rarely occur in naturalistic text, and potentially introduces non-syntactic confounding factors that models can exploit to succeed in the task, such as the strong correlation between the number of the first noun and the number of the verb (§3.4). In contrast, the targeted syntactic evaluations benchmark was created through a set of templates that are based on non-recursive context-free grammars. This approach thus enables the creation of a large number of syntactically challenging examples that reflect different types of syntactic competence, whilst controlling for the presence of non-syntactic confounding factors that may otherwise be present in naturalistic text.[13]

Concretely, the targeted syntactic evaluations benchmark assesses three broad types of syntactic competence: Number agreement, reflexive anaphoras, and negative polarity items (NPIs). Here we provide an illustrative example for each of these syntactic phenomena, and refer interested readers to the work of Marvin and Linzen (2018) for more detail.

[13]Despite its benefits, constructing the test examples from a template has a key drawback: The generated examples can be more artificial (and potentially more nonsensical) than naturalistic text that occurs in the wild.

- **Number agreement**: Under this construction, the model needs to determine the correct number of the verb, often in the presence of multiple nouns (but only one of which is the agreement controller). We remark that the agreement controller is not necessarily the subject of the sentence, such as in the case of *agreement in a sentential complement*: "*The bankers knew the* ***officer smiles****/*smile*", where the agreement controller "***officer***" and the correct verb "***smiles***" are both denoted in bold, while the incorrect verb "*smile*" is denoted in underline. Under this construction, the agreement controller coincides with the most adjacent noun to the verb. In contrast, under the *agreement across an object relative clause* construction: "*The* ***farmer*** *that the parents love* **swims**/**swim*", the agreement controller coincides with the subject and first noun of the sentence, "***farmer***". Hence, due to the variety of syntactic constructions in the benchmark, models that rely on certain non-syntactic heuristics, such as predicting verbs that agree with the number of the most recent noun (or conversely, the number of the first noun), would fail at some constructions.

- **Reflexive anaphoras**: In English, a reflexive pronoun such as "*himself*" or "*themselves*" requires an antecedent in a particular structural configuration, which forms the basis of its interpretation (Chomsky, 1981). An example of this is provided under the *reflexive in a sentential complement* construction: "*The bankers thought the* **pilot** *embarrassed* ***himself****/*themselves*". In this case, the only structurally available antecedent for a reflexive is the singular noun "***pilot***", as opposed to the plural noun "*bankers*", which coincides with the subject of the sentence. Hence, the reflexive pronoun "***himself***" (or alternatively, "***herself***") is syntactically licit, while the alternative "**themselves*" is syntactically illicit.

- **Negative polarity items (NPIs)**: Negative polarity items are words that occur within the context of a negation. An example of this is provided under the *NPI across a relative clause* construction, where the use of the word "***ever***" in the following sentence is syntactically licit: "***No*** *authors that* **the** *security guards like have* **ever** *been famous*". In contrast, simply swapping the words "***No***" and "***the***" in the aforementioned example would result in a minimally different yet ungrammatical sentence: "****The*** *authors that* ***no*** *security guards like have ever been famous*", where the use of the word "*ever*" is not licensed in this case. Succeeding in this task therefore necessitates an understanding of the scope of the negation, where the syntactic node that dominates the negative noun phrase must also dominate the NPI "***ever***". We refer interested readers to the work of Baker (1970) and Giannakidou et al. (1998), which discuss this phenomenon in detail.

Altogether, the targeted syntactic evaluations benchmark covers 15 different syntactic constructions, with 10 types of number agreement constructions, 3 types of reflexive anaphora constructions, and 2 types of NPI constructions. We shall report model performance on each individual construction, and further report the average performance across each broad category of syntactic competence (number agreement, reflexive anaphoras, and NPIs), alongside the average performance across all 15 constructions. We remark that the targeted syntactic evaluations benchmark is an *evaluation-only* dataset, and thus has no corresponding training or validation splits. Here we use the Gulordava et al. (2018) Wikipedia dataset (§4.4.1) to train and validate the LM, and only apply targeted syntactic evaluations at test time.

Metric. Similar to the number agreement dataset that we used in Ch. 3, the primary evaluation metric on targeted syntactic evaluations is *model accuracy*. Concretely, let $\mathbf{x}$ be a syntactically licit evaluation sentence (*e.g.,* "*The bankers thought the* ***pilot*** *embarrassed* ***himself***"). Let $\tilde{\mathbf{x}}$ be a syntactically illicit evaluation sentence that is minimally different from $\mathbf{x}$ (*e.g.,* "**The bankers thought the* ***pilot*** *embarrassed* <u>*themselves*</u>"). Each evaluation instance in the targeted syntactic evaluations benchmark is a tuple $(\mathbf{x}, \tilde{\mathbf{x}})$. A pre-trained LM (with model parameters $\boldsymbol{\theta}$) is considered correct for a particular evaluation instance if and only if it assigns a higher probability to the grammatical sentence, *i.e.,* $p_{\boldsymbol{\theta}}(\mathbf{x}) > p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})$.[14]

4.4.3 Baseline: Targeted Syntactic Evaluations with LSTMs

We begin our experiments by asking: To what extent can LSTM LMs, which lack explicit modelling of syntactic structures, succeed at a challenging and diverse benchmark of syntactic competence? While Marvin and Linzen (2018) have previously evaluated LSTM LMs on targeted syntactic evaluations, our investigation here differs by virtue of a more rigorous evaluation. To that end, we: (i) carefully tune the hyper-parameters of the LSTM LM, as the choice of hyper-parameters can account for a large variation in performance (Melis et al., 2018), and (ii) report the performance of multiple models trained from different random seeds, revealing the extent to which the observed results vary across different runs.

[14]In practice, such "forced choice" evaluation overlooks the probability difference that the LM assigns to the two sentences, *i.e.,* how much does the LM prefer the grammatical sentence $\mathbf{x}$ over the ungrammatical one $\tilde{\mathbf{x}}$? This can be interpreted as a proxy for the strength of the model's preference for grammaticality.

Hyper-parameters. To facilitate a fair comparison with Marvin and Linzen (2018), we use the same LSTM architecture with 2 layers and 650 hidden units per layer. We similarly train the LSTM LM on the training and (subsampled) validation splits of the Gulordava et al. (2018) dataset (§4.4.1), and only apply targeted syntactic evaluations once training has concluded. We optimise all other hyper-parameters through a grid search, and select the best hyper-parameters based on validation perplexity. Based on the grid search, we use mini-batch gradient descent with an initial learning rate of 0.45, which is decayed exponentially by a factor of 0.9 at every epoch after the tenth. We further apply dropout with a rate of 0.2 on the connections between: (i) the input layer and the first hidden layer, and (ii) the first and the second hidden layers. We implement all models using DyNet (Neubig et al., 2017a) with automatic dynamic batching (Neubig et al., 2017b), and use a batch size of 20 sentences. Each LSTM LM is trained on a V100 GPU, which takes about three days.

Discussion. We present our findings in Table 4.1 (**"Our LSTM"**); for all our models, we report the mean and standard deviation of 10 identical models trained from different random seeds. Based on the findings in Table 4.1, we remark on three key observations. First, our LSTM LM achieves a much better perplexity than the LSTM LM of Marvin and Linzen (2018, **"M&L-LSTM")**, with a 32% relative ppl. reduction; furthermore, our LSTM LM *also* achieves a better perplexity than the multi-task LSTM (**"M&L-Multi"**, 12% relative ppl. reduction) that benefits from explicit syntactic annotations in the form of combinatory categorial grammars (Steedman, 2000, CCGs). As our LSTM LM has the exact same number of LSTM layers and hidden units, we attribute this gap to differences in codebases and hyper-parameter optimisation, which we select based on an extensive grid search of possible hyper-parameter values. Second, contrary to the findings of Marvin and Linzen (2018), our LSTM model is, in fact, able to perform fairly well at targeted syntactic evaluations. In terms of overall accuracy, our conventional LSTM LM outperforms the multi-task LSTM model (**"M&L-Multi"**) that leverages explicit syntactic annotations in the form of CCGs, and is even able to match or exceed human performance on 7 out of all 15 constructions. Our conclusion is thus consistent with prior findings on the ability of neural language models to acquire complex syntactic generalisations when they are trained on large amounts of data in an unsupervised fashion, even when they are provided with no

	Marvin & Linzen models		Ours	Ours (small training)		
	M&L-LSTM	**M&L-Multi**	**Our LSTM**	**Small LSTM**†	**RNNG**†	**Humans**
Gulordava et al. (2018) **test perplexity**	78.65	61.10	**53.73** (±0.16)	94.54 (±0.21)	**92.30** (±0.27)	N/A
NUMBER AGREEMENT						
Simple	0.94	**1.00**	**1.00** (±0.00)	0.89 (±0.03)	**0.99** (±0.01)	0.96
In a sentential complement	**0.99**	0.93	0.97 (±0.02)	0.89 (±0.01)	**0.93** (±0.02)	0.93
Short VP coordination	0.90	0.90	**0.96** (±0.02)	0.90 (±0.03)	**0.96** (±0.02)	0.94
Long VP coordination	0.61	0.81	**0.82** (±0.05)	0.78 (±0.03)	**0.94** (±0.03)	0.82
Across a prepositional phrase	0.57	0.69	**0.89** (±0.02)	0.83 (±0.02)	**0.95** (±0.01)	0.85
Across a subject relative clause	0.56	0.74	**0.87** (±0.02)	0.81 (±0.04)	**0.95** (±0.03)	0.88
Across an object relative clause	0.50	0.57	**0.77** (±0.11)	0.54 (±0.08)	**0.95** (±0.03)	0.85
Across an object relative clause (no *that*)	0.52	0.52	**0.70** (±0.05)	0.55 (±0.07)	**0.93** (±0.02)	0.82
In an object relative clause	0.84	0.89	**0.90** (±0.03)	0.79 (±0.05)	**0.96** (±0.01)	0.78
In an object relative clause (no *that*)	0.71	0.81	**0.86** (±0.05)	0.72 (±0.03)	**0.96** (±0.02)	0.79
Average of number agreement	0.71	0.79	**0.87** (±0.02)	0.77 (±0.02)	**0.95** (±0.01)	0.86
REFLEXIVE ANAPHORA						
Simple	0.83	0.86	**0.91** (±0.01)	**0.93** (±0.01)	0.83 (±0.02)	0.96
In a sentential complement	**0.86**	0.83	0.81 (±0.02)	**0.77** (±0.03)	0.46 (±0.05)	0.91
Across a relative clause	0.55	0.56	**0.64** (±0.02)	0.63 (±0.02)	**0.82** (±0.02)	0.87
Average of reflexive anaphora	0.75	0.75	**0.79** (±0.01)	**0.78** (±0.01)	0.70 (±0.02)	0.91
NEGATIVE POLARITY ITEMS						
Simple	0.40	0.48	**0.96** (±0.04)	**0.93** (±0.06)	0.28 (±0.05)	0.98
Across a relative clause	0.41	0.73	**0.75** (±0.12)	**0.82** (±0.09)	0.78 (±0.06)	0.81
Average of negative polarity items	0.41	0.61	**0.86** (±0.06)	**0.88** (±0.05)	0.53 (±0.04)	0.90
Average of all constructions	0.68	0.75	**0.85** (±0.02)	0.79 (±0.02)	**0.85** (±0.02)	0.88

Table 4.1: Our replication of the Marvin and Linzen (2018) results. **M&L-Multi** is the Marvin and Linzen (2018) LSTM trained on a combination of LM and CCG supertagging (Bangalore and Joshi, 1999; Clark and Curran, 2007) losses, with an interpolation factor of 0.5. We report **our LSTM**, **small LSTM**†, and **RNNG**† performance (†smaller training data) in the format of *mean (±standard deviation)* of 10 identical models from different seeds. Results in bold denote the best among models trained on similar amounts of data.

explicit syntactic supervision or inductive biases (Gulordava et al., 2018; Goldberg, 2019).[15]

Third, despite the small variance in perplexity (stdev 0.16 ppl.), our trained LMs can nevertheless exhibit a large variance in accuracy for some constructions (*e.g.,* stdev 0.12 for NPI across a relative clause). This observation is consistent with earlier findings that models with similar perplexity may exhibit different patterns of syntactic generalisation (Kuncoro et al., 2018b; Tran et al., 2018), indicating a partial dissociation between perplexity and a model's ability to acquire syntactic competence across a diverse set of constructions. More broadly, our finding offers an important lesson for evaluating the syntactic competence of neural LMs: We discourage future work from reporting results based on single runs, as whether or not a model is able to master a certain syntactic construction can vary substantially across different random seeds, even for the exact same architecture and hyper-parameters.

[15]Our goal in this work is not to show that conventional LSTM LMs are unable to acquire a high degree of syntactic competence (indeed, our findings have shown that they can), but rather to show that these models can do *even better* if we augment them with more explicit syntactic inductive biases, in addition to data scale.

4.4.4 Baseline: Targeted Syntactic Evaluations with RNNGs

Having established that LSTM LMs can perform quite well at targeted syntactic evaluations (albeit imperfectly, as evidenced by the existing gap with human performance in terms of overall performance, last row of Table 4.1), we proceed to our second empirical question: To what extent can an RNNG model, which leverages explicit syntactic inductive biases and annotation, do well on targeted syntactic evaluations, even when it is trained on less data?

Experimental settings. We implement the RNNG model on the DyNet library (Neubig et al., 2017a), and employ the DyNet automatic batching feature (Neubig et al., 2017b) in order to speed up RNNG training. Here we train the RNNG model in a supervised fashion, which necessitates the presence of annotated phrase-structure trees (§2.4.3, Eq. 2.10). As gold-standard, manually-labelled phrase-structure tree annotations are not available on the Gulordava et al. (2018) Wikipedia dataset, we use an off-the-shelf, pre-trained Berkeley parser (Petrov and Klein, 2007) in order to obtain the predicted phrase-structure trees.

Marginal probability approximation. Evaluating the RNNG model—both in terms of language modelling perplexity and targeted syntactic evaluations—necessitates an estimate of $t_{\boldsymbol{\psi}}(\mathbf{x}) = \sum_{\mathbf{y} \in Y(\mathbf{x})} t_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{y})$, where $\boldsymbol{\psi}$ denotes the set of RNNG model parameters, and $Y(\mathbf{x})$ is the set of all possible phrase-structure trees for a sentence $\mathbf{x}$. As exact marginalisation is intractable, we employ the same word-synchronised beam search approximation as detailed in §3.5.1 and Eq. 3.6. Under this approximation, we obtain a lower bound on $t_{\boldsymbol{\psi}}(\mathbf{x})$ by marginalising over a small number of high-probability trees returned by the beam search:

$$t_{\boldsymbol{\psi}}(\mathbf{x}) = \sum_{\mathbf{y} \in Y(\mathbf{x})} t_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{y}) \geq \sum_{\mathbf{y}^{(n)}_{\text{beam}} \in \text{RNNGBeamSearch}(\mathbf{x}; \boldsymbol{\psi})} t_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{y}^{(n)}_{\text{beam}}),$$

where $\text{RNNGBeamSearch}(\mathbf{x}; \boldsymbol{\psi}) \subseteq Y(\mathbf{x})$ is the set of trees returned by the RNNG word-synchronised beam search procedure (Fried et al., 2017, §2.4.4) with fast-tracking (Stern et al., 2017). We use word and action beam sizes of $k = 50$ and $k \times 10 = 500$, respectively.

Training dataset. Under the ideal setup, we would train the RNNG on the entire Gulordava et al. (2018) training set, although it is infeasible to do so in practice due to the difficulty of scaling RNNGs to large datasets (§4.2).[16] Hence, we train the RNNG model on a smaller, 20% subset of the training set, corresponding to roughly 600,000 training sentences and 16 million words. As the Gulordava et al. (2018) dataset is already preprocessed, we select this subset such that all word types occur at least once on this smaller training set.

Hyper-parameters. In order to speed up training time, we use a smaller RNNG model size (2 hidden layers with 256 units per layer) compared to the LSTM model in §4.4.3, which has 2 hidden layers and 650 units per layer. We optimise the rest of the RNNG hyper-parameters through a similar grid search procedure, and select the hyper-parameters that yield the best validation set perplexity. We found the following optimiser to work best: mini-batch gradient descent with a learning rate of 0.3, which is then decayed exponentially by a factor of 0.92 at every epoch after the tenth. We use a batch size of 10 sentences, and apply variational dropout (Gal and Ghahramani, 2016) with a dropout rate of 0.3.

Discussion. We present the RNNG results in Table 4.1 (sixth column: **"RNNG"**), and compare with two different LSTM models trained on either: (i) the full dataset (fourth column: **"Our LSTM"**), and (ii) the same (smaller) training set as the RNNG (fifth column: **"Small LSTM"**).[17] We now remark on three key observations. First, our findings clearly reaffirm the benefits of *both* syntactic inductive biases and data scale. In terms of syntactic inductive biases, an RNNG model that leverages syntactic annotations and explicit compositions can outperform a comparable small LSTM model on 11 out of 15 constructions, and improves overall accuracy on targeted syntactic evaluations from 79% to 85%, corresponding to a substantial 29% reduction in relative error rates. Remarkably, the RNNG model can *match* the overall performance of the full LSTM model that is trained on 5x as much data: Both models achieve an 85% overall performance across all constructions, which is only 3% short of overall human performance on this benchmark. However, a closer examination of the two models' performance reveals that they have different success and failure modes: Concretely, RNNGs perform better than LSTMs on number agreement constructions, while the opposite holds true for reflexive anaphora and NPI constructions.

[16]Our preliminary experiments indicate that training an RNNG model on the whole Gulordava et al. (2018) training set, using the same number of model parameters as the LSTM model (§4.4.3), would require at least a few weeks of training time on a single V100 GPU, even with the help of automatic dynamic batching.

[17]We control such that the small LSTM baseline has a similar number of parameters as the RNNG.

Second, beyond the benefits of explicit syntactic inductive biases, our findings also affirm the importance of data scale for building syntactically competent language models. We find that the LSTM LM trained on the full training set outperforms the LSTM trained on a smaller training set by a large margin (79% overall performance for the small LSTM, as opposed to 85% for the larger LSTM trained on more data). In particular, the performance difference between the small and full LSTMs sheds light on which constructions are sensitive to variations in the amount of data. For instance, agreement across an object relative clause exhibits a large variation across the two training regimes (77% to 54%), suggesting that LSTM LMs require a large amount of data to do well at this type of construction. Our findings on the importance of data scale for LM training are consistent with the success of recent LM pre-training on large datasets (Peters et al., 2018; Devlin et al., 2019, *inter alia*), in addition to earlier work on noisy channel models for tasks such as machine translation and speech recognition (Jelinek, 1998; Rosenfeld, 2000; Koehn, 2010, *inter alia*).

Third, despite its smaller training set, the RNNG model can perform extremely well at number agreement, substantially outperforming not only the full LSTM that is trained on 5x more data, but also a pre-trained BERT (Devlin et al., 2019, Table 4.3) model that is trained on 150x more data. Despite its impressive performance at number agreement, the RNNG model still lags behind the full LSTM on reflexive anaphora and NPI, suggesting that larger amounts of data are likely to be necessary to master these types of syntactic competence.

4.5 DSA-LSTM Experiments

In §4.4.3 and §4.4.4, we have demonstrated the benefits of *both* syntactic inductive biases and data scale for acquiring a broad range of syntactic competence. Furthermore, we have shown that the benefits of syntactic inductive biases and data scale are, in fact, complementary in nature: A sequential LSTM language model that learns from a larger dataset has different success and failure modes from an RNNG model that leverages syntactic inductive biases, albeit at the expense of data scale. In light of this finding, we are interested in the following primary research question: Can a DSA-LSTM model perform *even better* at targeted syntactic evaluations by combining the best of both data scale and syntactic inductive biases, which is achieved through our proposed knowledge distillation learning procedure (§4.3)?

4.5.1 Experimental Settings

In order to enable a fair comparison, we train the DSA-LSTM model using the same experimental setup as the standard LSTM model in §4.4.3. Concretely, we train the DSA-LSTM on the entirety of the Gulordava et al. (2018) training set; compared to RNNGs, the DSA-LSTM model can be trained in a much shorter amount of time, as it shares the exact same sequential architecture as a conventional LSTM model, which can easily benefit from specialised hardwares like GPUs and TPUs. Note that the DSA-LSTM model differs from conventional LSTM LMs only in its training loss, which has an additional KD term that learns from the next-word predictive distribution of the RNNG teacher (Eq. 4.6).

Computational challenge. We remark that the KD loss that we use to train the DSA-LSTM introduces a computational challenge, as we need to compute the RNNG teacher's entire softmax for predicting the next word (Eq. 4.6); this needs to be done for each of the 80M word positions in the training set. To that end, one option is to compute the RNNG teacher's distributions on-the-fly as the DSA-LSTM training progresses, although in practice this would create a computational bottleneck due to the RNNG's high computational costs. Hence, here we choose to pre-compute and store the RNNG teacher's predictions, although a naïve implementation would result in a prohibitive memory footprint.[18] To circumvent this issue, for every token $x_i^{(n)}$ in the training set, we pre-compute and store the RNNG teacher's hidden state (rather than the full softmax distribution) $\mathbf{h}_i^{(n)} \in \mathbb{R}^H$, where $H \ll |\Sigma|$. Based on the pre-computed hidden states, we construct the teacher distribution on-the-fly through an affine transformation followed by a softmax, which has a negligible overhead.

Hyper-parameters. As previously mentioned, we use the exact same DSA-LSTM architecture as the conventional LSTM LM (§4.4.3), where we use a 2-layer sequential LSTM model with 650 hidden units per layer. One key difference is that we optimise the learning rate independently, as the optimal learning rate for the DSA-LSTM's interpolated KD loss (Eq. 4.7) may be different from the standard cross-entropy language modelling loss. To that end, we conduct a grid search that selects the best learning rate based on validation

[18]Pre-computing the RNNG's predictions necessitates storing the RNNG's entire softmax distribution at each word position in the training set (Eq. 4.6). Unfortunately, this requires storing $N \times |\Sigma|$ floating points, where N is the total number of training tokens and $|\Sigma|$ is the size of the vocabulary. For the Gulordava et al. (2018) training set (with 80M tokens), this corresponds to storing 4 trillion floating points, or 16 terabytes.

set perplexity,[19] based on which we select the following optimiser: a mini-batch gradient descent with an initial learning rate of 0.4, which is decayed exponentially by a factor of 0.9 at every epoch after the tenth. To regularise the model, we apply the same variational dropout (with a dropout probability of 0.2) as the conventional LSTM model in §4.4.3.

Training speed. Since the DSA-LSTM shares the exact same sequential architecture as a conventional LSTM, it is thus amenable to manual batching, and, on a V100 GPU, is about five times faster to train than a comparable RNNG. Despite this significant speed-up, training the DSA-LSTM in our basic implementation is still half as fast as a conventional LSTM LM. We attribute this difference to the additional overhead associated with the KD objective, such as the I/O operations that load the RNNG teacher's pre-computed hidden states, and computing the cross-entropy between the teacher and student models for the entire vocabulary (Eq. 4.6). These operations, however, only apply at training; at test time, there is no additional overhead for inferring $q_{\hat{\boldsymbol{\theta}}-\alpha\text{-int}}(\mathbf{x}^{(n)})$ under the trained DSA-LSTM.

Baselines. All in all, the DSA-LSTM benefits from three key components: (i) a KD objective, which in itself has been shown to be a good regulariser for language modelling, even when the teacher model does not necessarily have syntactic inductive biases (Furlanello et al., 2018); (ii) the scalability of sequential architectures, which enables the DSA-LSTM to learn from larger datasets; and (iii) syntactic inductive biases, which in the case of the DSA-LSTM comes from distilling the predictions of the RNNG teacher (§4.6). We argue that it is important not only to show that the DSA-LSTM model can perform better, but also understand the extent to which these three components contribute to model performance. To that end, we compare the DSA-LSTM with the following comprehensive set of baselines:

- a well-tuned LSTM LM (§4.4.3) that is scalable but lacks syntactic biases (**"Full LSTM"**);
- the RNNG teacher trained on a 20% subset of the training data (§4.4.4), which benefits from syntactic inductive biases, albeit at the expense of model scalability (**"RNNG"**);
- a DSA-LSTM that is trained on the same, smaller 20% subset as the RNNG teacher (**"S-DSA-LSTM"**). Crucially, this baseline isolates the importance of scalability: This

[19] Perplexity still constitutes a primary metric for language model evaluation, in addition to other metrics such as targeted syntactic evaluations accuracy, which are similarly based on the LM's assigned probability to a test utterance $q_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})$. Hence, we use validation set perplexity (as opposed to the interpolated KD loss in Eq. 4.7, which we use as the training objective) as the basis for early stopping and hyper-parameter tuning.

Model	Scalability	Syntactic Bias	KD Objective
Full LSTM	✓	✗	✗
RNNG	✗	✓	✗
BA-LSTM	✓	✗	✓
S-DSA-LSTM	✗	✓	✓
DSA-LSTM	✓	✓	✓

Table 4.2: A summary of the key properties of each model used in our experiment.

model still benefits from the KD objective and syntactive inductive biases (as derived from the RNNG teacher), although it lacks the opportunity to learn from a larger dataset;

- a born-again LSTM (Furlanello et al., 2018, **"BA-LSTM"**) that benefits from the same interpolated KD objective (in addition to model scalability), but lacks syntactic inductive biases due to the use of a sequential LSTM LM as the teacher model, as opposed to an explicitly hierarchical RNNG model. We discuss this model in more detail below.

Born-Again LSTM (BA-LSTM). We remark that the interpolated KD objective in Eq. 4.7 is not specific to the RNNG teacher, and can in fact be applied to *any* pair of teacher and student models. To that end, Furlanello et al. (2018) surprisingly achieved better language modelling perplexity through a *born-again* approach, where they first: (i) trained an LSTM LM on the gold data with cross-entropy (Eq. 4.2), and then (ii) used the resulting trained LSTM as a teacher model for a student model that shares the exact same architecture and model capacity as the teacher. In light of this result, we aim to understand the importance of learning from an explicitly hierarchical RNNG teacher model (which is not the case with BA-LSTM, where the teacher model is sequential); *i.e.,* how much of the DSA-LSTM performance can be attributed to the use of a hierarchical RNNG teacher? Would a born-again LSTM model that learns from the exact same interpolated KD objective, albeit from a sequential teacher instead of RNNGs, achieve the same performance as the DSA-LSTM?

Summary of baselines. We summarise the key properties of each model in Table 4.2.

4.5.2 Empirical Findings

Discussion. We present our findings in Table 4.3. In order to avoid clutter, for each model, we present only the mean performance of 10 identical models that are trained from different

	Small Training Set			Full Training Set				
	Small LSTM†	S-DSA-LSTM†	RNNG†	Full LSTM	BA-LSTM	DSA-LSTM	BERT	Humans
Gulordava et al. (2018) **test ppl.**	94.54	93.95	**92.30**	**53.73**	54.64	56.74	N/A	N/A
NUMBER AGREEMENT								
Simple	0.89	0.96	**0.99**	**1.00**	**1.00**	**1.00**	1.00	0.96
In a sentential complement	0.89	**0.98**	0.93	0.97	**0.98**	**0.98**	0.83	0.93
Short VP coordination	0.90	0.88	**0.96**	0.96	0.95	**0.99**	0.89	0.94
Long VP coordination	0.78	0.74	**0.94**	**0.82**	0.80	0.80	0.98	0.82
Across a prepositional phrase	0.83	0.88	**0.95**	0.89	0.89	**0.91**	0.85	0.85
Across a subject relative clause	0.81	0.87	**0.95**	0.87	0.87	**0.90**	0.84	0.88
Across an object relative clause	0.54	0.69	**0.95**	0.77	0.81	**0.84**	0.89	0.85
Across an object relative clause (no *that*)	0.55	0.61	**0.93**	0.70	0.74	**0.77**	0.86	0.82
In an object relative clause	0.79	0.87	**0.96**	0.90	0.91	**0.92**	0.95	0.78
In an object relative clause (no *that*)	0.72	0.88	**0.96**	0.86	0.83	**0.92**	0.79	0.79
Average of number agreement	0.77	0.84	**0.95**	0.87	0.88	**0.90**	0.89	0.86
REFLEXIVE ANAPHORA								
Simple	**0.93**	0.90	0.83	0.91	**0.92**	0.91	0.94	0.96
In a sentential complement	0.77	**0.78**	0.46	0.81	0.81	**0.82**	0.89	0.91
Across a relative clause	0.63	0.67	**0.82**	0.64	0.64	**0.67**	0.80	0.87
Average of reflexive anaphora	**0.78**	**0.78**	0.70	0.79	0.79	**0.80**	0.88	0.91
NEGATIVE POLARITY ITEMS								
Simple	0.93	0.84	0.28	0.96	**0.98**	0.94	N/A	0.98
Across a relative clause	0.82	0.73	0.78	0.75	0.70	**0.91**	N/A	0.81
Average of negative polarity items	**0.88**	0.79	0.53	0.86	0.84	**0.92**	N/A	0.90
Average of all constructions	0.79	0.82	**0.85**	0.85	0.86	**0.89**	N/A	0.88

Table 4.3: Empirical findings for the **"DSA-LSTM"**. For each column, we report the mean of 10 identical models trained from different random seeds; standard deviation values are reported in Appendix A. **"S-DSA-LSTM"** indicates the DSA-LSTM trained on the smaller RNNG training set, whereas **"BA-LSTM"** is the born-again model where the teacher model is a full LSTM LM. We also compare with the performance of **"BERT"**-Base on the same targeted syntactic evaluations benchmark, which is not strictly comparable as BERT is trained on 30 times more data. † indicates models trained on the smaller training set (§4.4.4). Results in bold denote the best among those trained on the same amounts of data.

random seeds; we include standard deviation information in Appendix A. Based on the findings in Table 4.3, we remark on the following key observations.

- Of the three models trained on the smaller training set, the S-DSA-LSTM model outperforms a comparable small LSTM trained with the standard LM cross-entropy loss (Eq. 4.2), improving overall acccuracy from 0.79 to 0.82 (a 14% relative error reduction), even though both models share the exact same sequential architecture and data size (*i.e.,* only the training objective is different). On number agreement, the S-DSA-LSTM model successfully narrows the gap between the faster LSTM model and the slower RNNG teacher, which benefits from syntactic annotations and inductive biases at the expense of scalability. All in all, these findings confirm our hypothesis that the KD approach constitutes an effective way for injecting syntactic biases into sequential LSTM models.
- Among the models trained on the full training set, the born-again model (BA-LSTM) is able to outperform a conventional LSTM model, albeit only by a small margin. This finding suggests that our proposed interpolated KD objective (Eq. 4.7) helps improve the

syntactic competence of LSTMs, even when the teacher model lacks explicit syntactic inductive biases, and shares the exact same architecture and model size as the student.

- In terms of LM perplexity, we find that both the BA-LSTM and DSA-LSTM models perform slightly worse than a conventional LSTM LM trained on the standard LM cross-entropy loss (*i.e.,* without any KD loss, Eq. 4.6). We attribute this gap to the smoother target distribution when using KD (Fig. 4.2), which discourages the student model from assigning high probabilities to the correct next word $x_i^{(n)}$, unless the teacher model is extremely confident in predicting $x_i^{(n)}$. This observation is consistent with earlier findings on label smoothing in machine translation (Pereyra et al., 2017; Vaswani et al., 2017), which often results in better BLEU scores at the expense of slightly worse likelihoods.

- Despite identical architectures, the DSA-LSTM substantially improves over the full LSTM in terms of overall performance (85% to 89%), constituting a 27% relative error rate reduction and establishing a new state of the art as of mid-2019. Moreover, in terms of overall performance, the DSA-LSTM model is able to achieve parity with human performance at targeted syntactic evaluations. Crucially, our findings suggest that the DSA-LSTM combines the best of both syntactic biases and data scale: On number agreement, the DSA-LSTM improves over the LSTM baseline, and narrows the gap with the RNNG teacher, while at the same time performing well on reflexive anaphora and NPI, on which the RNNG teacher (but not the full LSTM) fails to achieve a good performance. Remarkably, despite the RNNG teacher's poor performance on reflexive anaphora and NPI, the DSA-LSTM is, in fact, *still* able to outperform the LSTM baseline on these constructions, suggesting that the RNNG's predictions still offer an informative learning signal for the student, even for these constructions where the RNNG performs poorly.

- While not directly comparable, the DSA-LSTM can outperform a pre-trained BERT model (Devlin et al., 2019; Goldberg, 2019)[20] at number agreement. This result is even more striking considering that BERT was trained on 30 times as much data as the DSA-LSTM, and further benefits from bidirectionality. This finding suggests that, at least in terms of syntactic competence, syntactic inductive biases continue to be advantageous, even in the case of LSTM LMs that are able to learn from larger amounts of data.

[20]Goldberg (2019) applied an additional preprocessing step that removes sentences where the focus verb does not appear as a single "wordpiece"; hence, their evaluation sentences are slightly different.

4.6 Probing Analysis

Having established the clear advantages of DSA-LSTM at targeted syntactic evaluations, we now attempt to understand: How exactly does the DSA-LSTM's internal representations differ from that of a conventional LSTM? Would the DSA-LSTM's internal representations necessarily encode more syntactic information? To that end, we adopt the method of Blevins et al. (2018), and employ a classifier probe (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Conneau et al., 2018; Hewitt and Manning, 2019, *inter alia*) that predicts the *grandparent constituent*[21] of a word token $x_i^{(n)}$, based on the LSTM's top-layer hidden state representation around that word position. Under this framework, we interpret the accuracy of the probe on a held-out set as a proxy to how well the model's internal representations encode the relevant syntactic information that is required to succeed in this task.

We use a linear classifier as the probe, and obtain the grandparent non-terminal labels using the same Berkeley parser (§4.4.4) that we used to obtain predicted phrase-structure trees on the Gulordava et al. (2018) dataset. For the probing experiment, we randomly select sentences from each respective training, validation, and test split of the Gulordava et al. (2018) dataset to yield ∼300,000 words for training, and ∼10,000 words for each of the validation and test sets. We concatenate the LSTM hidden states at the current and next words as the probe features,[22] *i.e.,* $[\mathbf{h}_i^{(n)}; \mathbf{h}_{i+1}^{(n)}]$, where ; denotes vector concatenations. We remind the reader that the DSA-LSTM operates on word sequences, and has no direct access to the parse information during training; the only additional supervision it has is in the form of the RNNG teacher's next-word distribution (Eq. 4.6). Hence, both the DSA-LSTM and the conventional LSTM LM have to infer the relevant syntactic information from data.

Discussion. We present the probing result in Fig. 4.3, which indeed confirms our hypothesis: The DSA-LSTM has a much higher probing accuracy than the conventional LSTM LM (83% to 74%, corresponding to a 34% relative error reduction). This finding suggests that

[21] As an example of the grandparent constituent prediction task, consider the following simple example "(S (NP *The cat* (PP *in* (NP *the* ***bathroom***))) (VP ***meows***))". The grandparent constituent of "***bathroom***" is thus the prepositional phrase symbol "PP", while the grandparent constituent of "***meows***" is the non-terminal "S".

[22] Our probing feature set thus slightly differs from that of Blevins et al. (2018), who concatenated the hidden states of a left-to-right and a right-to-left LSTM language model. For this reason, we augment our probing features with the hidden state at the next time step $\mathbf{h}_{i+1}^{(n)}$—as opposed to just the hidden state at the current time step $\mathbf{h}_i^{(n)}$—in order to provide a more comparable setup to the work of Blevins et al. (2018) that incorporates the right context $\mathbf{x}_{>i}$, which here is achieved through a one-word lookahead into the right context.

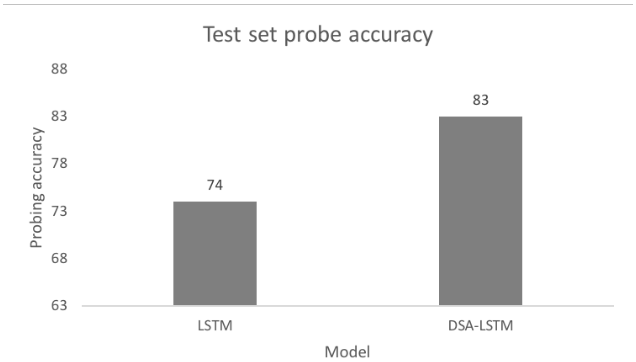


Figure 4.3: Probing accuracy on the test set. We analyse the hidden states of the LSTM and DSA-LSTM to interpret the syntactic information encoded in each model's hidden states.

the means by which the DSA-LSTM achieves a better syntactic competence is by tracking and capturing more syntactic information during the course of sequential processing.

4.7 Related Work

Augmenting language models with syntactic information and inductive biases has been a long-standing area of research. To that end, syntactic language models estimate the joint probability of surface strings and some form of syntactic structures (Jurafsky et al., 1995; Chelba and Jelinek, 2000; Roark, 2001; Henderson, 2004; Emami and Jelinek, 2005; Buys and Blunsom, 2015; Mirowski and Vlachos, 2015; Dyer et al., 2016; Kim et al., 2019, *inter alia*). In contrast, the DSA-LSTM is not a syntactic language model, as it only models the probability of surface strings $q_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})$, although it does benefit from an auxiliary loss that distills the next-word predictive distribution of an RNNG syntactic language model.

Beyond syntactic LMs, a separate line of prior work has explored multi-task learning with syntactic objectives as an auxiliary loss, primarily in the context of language modelling and machine translation (Luong et al., 2016; Eriguchi et al., 2016; Nădejde et al., 2017; Enguehard et al., 2017; Aharoni and Goldberg, 2017; Eriguchi et al., 2017). Our approach of injecting syntactic biases through KD is orthogonal to this line of work: One primary difference is that the DSA-LSTM has no direct access to syntactic annotations; it does, however, have access to a teacher softmax distribution over the next word, which is derived from an RNNG teacher model that leverages syntactic annotations and inductive biases. This approach is related to the earlier work of Smith and Eisner (2012), who trained a generative Hidden Markov Model on a *distribution* over sentences derived from n-gram counts.

Our approach is also closely related to recent work that introduces syntactically-motivated inductive biases into language models. For instance, Chung et al. (2017) segmented the hidden state update of an RNN through a multi-scale hierarchical recurrence, thereby providing a shortcut to the gradient propagation of long-range, hierarchical dependencies. Yogatama et al. (2018) introduced a stack-structured memory to encourage hierarchical modelling in language models, where the resulting model successfully outperforms conventional LSTMs at number agreement evaluation (Linzen et al., 2016). Shen et al. (2019) imposed a hierarchical bias on the LSTM cell-updating mechanism, based on the intuition that larger constituents contain information that changes more slowly across the sequence. Our proposed method is orthogonal, and can be applied on top of these new architectures.

More recently, Noji and Oseki (2021) proposed an ingenious way of manually batching RNNGs, which renders them much more amenable to GPU acceleration and yields a 6x speed-up over the standard implementation. In contrast to our work, their approach constitutes a more direct way of scaling the syntactic inductive biases of RNNGs to larger datasets. Despite this recent progress, we argue that our KD procedure is still relevant for two reasons. First, our proposed KD approach can, in fact, work in conjunction with the approach of Noji and Oseki (2021): As long as there is still a gap in training speed between the manually-batched RNNGs and the current generation of scalable sequence models (*e.g.,* Transformers), we can use the technique of Noji and Oseki (2021) to train the RNNG teacher on as much data as is practically feasible, and then use our KD technique to transfer its knowledge to a Transformer model, which can be trained on even-larger datasets by virtue of its stronger parallelism. Second, our interpolated KD approach can be applied to *any* widely used sequence models as the student, including LSTMs, Transformers, or GRUs (Cho et al., 2014); all it takes to implement our approach is a minor modification of the student's training loss. Hence, whereas the approach of Noji and Oseki (2021) works directly with RNNGs and thus presents a higher barrier for widespread adoption, our approach facilitates the injection of syntactic inductive biases into sequence models that are *already* widely used in many practical systems, without necessitating any major architectural changes.[23]

[23]Note that our proposed interpolated KD loss only applies at training. Afterwards, there is no difference at all between *e.g.,* a conventional LSTM and a DSA-LSTM that benefits from stronger syntactic biases.

4.8 Conclusion

In this chapter, we introduced distilled syntax-aware (DSA) LSTMs, which combine the best of both syntactic inductive biases and data scale. We achieve this by distilling the marginal next-word distributions of an explicitly hierarchical (albeit computationally expensive) RNNG teacher model, which can only be trained on a smaller subset of the data by virtue of its lack of scalability. While we find that conventional LSTMs are able to perform better at targeted syntactic evaluations than previously thought, our proposed DSA-LSTM model improves over this strong baseline, yields a new state of the art,[24] compares favourably to a BERT model trained on more data, and, despite its sequential architecture, encodes latent syntactic information to a large extent. All in all, our findings suggest that stronger syntactic inductive biases continue to be beneficial for improving the syntactic competence of neural LMs, even in the case of LSTM language models that learn from medium-sized corpora.

[24]Our DSA-LSTM model achieved a new state of the art on targeted syntactic evaluations as of mid-2019.

Chapter 5

Syntactic Structure Distillation Pre-training for Bidirectional Encoders

Chapter Abstract

Recently, Transformer-based bidirectional language models—which are pre-trained on large amounts of data—have achieved remarkable success at various NLP tasks (Peters et al., 2018; Devlin et al., 2019), in addition to various benchmarks of syntactic competence (Goldberg, 2019). Hence, given our earlier findings on the benefits of syntactic biases for sequential LSTMs that learn from larger datasets, here we go one step further and ask: Would syntactic inductive biases continue to be relevant, even for Transformer models that can learn from much larger datasets? And where exactly would such syntactic inductive biases be helpful in different types of NLP tasks, above and beyond improving syntactic competence? To answer these questions, we apply a similar KD strategy that enables a scalable BERT model to learn from the predictions of an explicitly hierarchical (albeit harder to scale) RNNG teacher. Since BERT estimates the probability of masked words in bidirectional context, we distill the RNNG teacher's approximate marginal distribution over words in context. Overall, our structure-distilled BERT models reduce relative error by 2-21% on a diverse set of structured prediction tasks, although we obtain mixed results on GLUE. Altogether, our findings reaffirm the benefits of syntactic biases, even for Transformer models that learn from much larger datasets, and contribute to a better understanding of where exactly syntactic biases are most helpful in NLP.

The material in this chapter is adapted from Kuncoro et al. (2020)

5.1 Introduction

In Ch. 4, we have demonstrated how syntactic inductive biases—which are acquired by learning from the predictions of an explicitly hierarchical (albeit harder to scale) RNNG teacher—can substantially improve the syntactic competence of sequential DSA-LSTMs that can learn from larger amounts of data. More recently, however, Transformers (Vaswani et al., 2017) have supplanted LSTMs as the new backbone of current state of the art models in NLP (Devlin et al., 2019; Brown et al., 2020, *inter alia*), partly due to our ability to train them on *even larger* datasets by virtue of their stronger parallelism. In light of this success, in this chapter, we focus on the BERT model (Devlin et al., 2019)—a widely used Transformer-based model that is pre-trained on large amounts of data through a bidirectional masked language modelling loss; the pre-trained BERT model can then be easily fine-tuned to a downstream task of interest. Our choice of BERT is motivated by its remarkable success at various tasks, often outperforming prior approaches by a large margin (Devlin et al., 2019). More intriguingly, BERT has *also* been shown to do well at syntactic grammaticality judgment tasks (Goldberg, 2019), and encode substantial amounts of syntax in its learnt representations (Liu et al., 2019a; Tenney et al., 2019a,b; Hewitt and Manning, 2019; Jawahar et al., 2019), even though the underlying Transformer architecture lacks explicit modelling of syntactic structures, and does not benefit from any form of syntactic supervision.

In light of this success, we build on our findings that highlight the benefits of syntactic inductive biases in Ch. 4, and ask: Can Transformer-based masked language models like BERT—which are pre-trained on much more data than the DSA-LSTM model in Ch. 4, and have *already* achieved wide-ranging success at various NLP benchmarks—*still* benefit from more explicit syntactic biases? And what types of natural language understanding tasks would benefit the most from more explicit syntactic inductive biases, above and beyond the syntactic grammaticality judgment tasks that we explored in Chs. 3 and 4? To answer these questions, we devise a new pre-training strategy that injects syntactic biases into a BERT (Devlin et al., 2019) model that works well at scale. To that end, our approach is based on the RNNG knowledge distillation procedure in Ch. 4, albeit with one key difference: As BERT models a collection of conditionals for words in *bidirectional* context (§5.3), we propose to distill the RNNG's marginal distribution over words in context, denoted as $t_{\psi}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$, for which we develop an efficient yet effective approximation (§5.4).

We evaluate our structure-distilled BERT on a diverse set of 6 structured prediction tasks, where our approach outperforms the standard BERT baseline by varying margins; rather surprisingly, the benefits of syntactic inductive biases *also* extend to structured prediction tasks that are not explicitly syntactic in nature, such as coreference resolution and semantic role labelling. Despite the gains on the structured prediction tasks, we achieve mixed results on GLUE: Our approach improves performance on the corpus of linguistic acceptability (Warstadt et al., 2018, CoLA), but performs slightly worse on the rest of GLUE. This finding suggests that syntactic biases do not benefit all tasks to the same extent, and alludes to a partial dissociation between performance on GLUE and on structured prediction tasks.

All in all, our findings are important in three ways, as they: (i) showcase the benefits of more explicit syntactic biases, even for Transformer-based models like BERT that learn from large amounts of data; (ii) contribute to a better understanding of where exactly syntactic biases are beneficial in NLP, and where they are not; and most importantly, (iii) going forward, make a case for designing scalable models that not only can be trained on large amounts of data, but also integrate stronger notions of syntactic biases at the same time.

5.2 The Case for Augmenting BERT with Syntactic Biases

While we have demonstrated the benefits of syntactic inductive biases for the sequential DSA-LSTM model in Ch. 4, we argue that extending the RNNG knowledge distillation (KD) approach into BERT is an important complement to this work for three reasons.

- While the sequential DSA-LSTM model is already much more scalable than the RNNG teacher, it is nevertheless still impractical to train the DSA-LSTM model on extremely large amounts of data; this is due to the inherent sequential dependencies within LSTM architectures.[1] In practice, such sequential dependencies within LSTMs constitute a barrier to higher parallelism under specialised deep learning hardwares like GPUs and TPUs, and thus render LSTM architectures much less scalable compared to Transformers.

[1]Under the LSTM architecture, the current hidden state $\mathbf{h_i}$ can only be computed once all its preceding hidden states $\mathbf{h_{<i}}$ have been computed. In contrast, each Transformer layer can construct the hidden state for a particular word token entirely in parallel; hence Transformers benefit from a much higher training speed-up under deep learning hardwares. Recent work, however, has proposed alternative RNN-based architectures that are much easier to parallelise, which results in better training speed compared to Transformers (Lei, 2021).

The BERT model, for instance, is trained on a corpus of 3.4 billion words—nearly 40x larger than the DSA-LSTM training set in Ch. 4, whereas other Transformer-based models like RoBERTa (Liu et al., 2019b) and GPT-3 (Brown et al., 2020) are trained on even-larger datasets. Would more explicit syntactic biases continue to be beneficial, even for such Transformer-based models that are pre-trained on very large amounts of data? To that end, our proposed knowledge distillation approach provides an avenue to examine *how* the addition of more explicit syntactic biases changes the performance of BERT on different types of NLP benchmarks, and thus provides an answer to this question.

- In Ch. 4, we have demonstrated the effectiveness of our proposed RNNG knowledge distillation approach for a DSA-LSTM student that models the probability of the next word conditional on the left context, *i.e.,* $q_{\boldsymbol{\theta}}(x_i^{(n)} \mid \mathbf{x}_{<i}^{(n)})$; we denote such models as *unidirectional*, left-to-right language models.[2] In contrast, the current dominant paradigm in textual representation learning is through *bidirectional* models that consider both the left and right context, such as BERT, XLNet (Yang et al., 2019), and Electra (Clark et al., 2020). In this chapter, we investigate whether a similar KD approach is *also* effective for injecting syntactic biases into a BERT model that operates in a bidirectional fashion—above and beyond the unidirectional DSA-LSTM that we explored in Ch. 4—which would provide evidence for the generality and robustness of our proposed approach.

- Thus far, we have established the benefits of more explicit syntactic biases within the context of grammaticality judgment tasks, including both number agreement (Ch. 3) and targeted syntactic evaluations (Ch. 4) benchmarks of syntactic competence. Would such syntactic biases also be beneficial for *other* natural language understanding (NLU) benchmarks, including for benchmarks that—unlike grammaticality judgment tasks—are not explicitly syntactic in nature? Given the widespread use of BERT in NLP (*e.g.,* by simply fine-tuning the pre-trained BERT model), extending our structure distillation approach into BERT pre-training thus provides an avenue to answer this question, and identify *where exactly* syntactic biases are most helpful among a wide range of benchmarks.

[2]A prominent example of such unidirectional, left-to-right language models is the family of Transformer-based GPT models, such as GPT-3 (Brown et al., 2020).

5.3 Background: BERT Pre-training Objective

We begin with a brief overview of the BERT masked language modelling objective, as proposed by Devlin et al. (2019). Concretely, let $\mathbf{x} = [x_1, \cdots, x_T]$ be a sequence of T tokens—where n is typically set to either 128, 256, or 512—that are sampled from the pre-training dataset. The aim of BERT pre-training is to find a set of model parameters $\hat{\boldsymbol{\theta}}_{\mathrm{B}}$ that would maximise the probability of reconstructing parts of the input $\mathbf{x}$, conditional on a corrupted version $c(\mathbf{x}) = c(x_1), \cdots, c(x_T)$, where $c(\cdot)$ denotes the stochastic corruption protocol of Devlin et al. (2019) that is applied to each word $x_i \in \mathbf{x}$. More formally:

$$\hat{\boldsymbol{\theta}}_{\mathrm{B}} = \arg\min_{\boldsymbol{\theta}} - \sum_{i \in M(\mathbf{x})} \log p_{\boldsymbol{\theta}}(x_i \mid c(x_1), \cdots, c(x_T)), \tag{5.1}$$

where $M(\mathbf{x}) \subset \{1, \cdots, T\}$ denotes the indices of *masked tokens* that serve as reconstruction targets. In practice, the corruption protocol $c(\cdot)$ and the reconstruction targets $M(\mathbf{x})$ are intertwined: Here $M(\mathbf{x})$ denotes the indices of tokens in $\mathbf{x}$ (typically $\sim 15\%$) that were altered by the corruption protocol $c(\mathbf{x})$.[3] This masked LM objective is combined with a next-sentence prediction of whether the two segments in $\mathbf{x}$ are contiguous sequences.

Corruption protocol. The BERT corruption protocol $c(x_i)$ would stochastically choose to "mask" the token x_i with a 15% probability, and leave x_i unmasked otherwise. In the event that x_i is chosen as a masked token, it is converted to a special "*[MASK]*" symbol with 80% probability. Nevertheless, the introduction of the "*[MASK]*" symbol creates a discrepancy between the BERT pre-training and fine-tuning stages, where such "*[MASK]*" symbol is unlikely to be encountered when fine-tuning on downstream tasks. To mitigate this, x_i is converted into a randomly sampled vocabulary item with 10% probability, and kept unchanged (albeit still considered as a masked token) with the remaining 10% probability.

Bidirectionality. Unlike the unidirectional, left-to-right language modelling training loss in Eq. 4.2, the BERT pre-training loss in Eq. 5.1 is inherently bidirectional: Here the BERT model predicts the true identity of the masked word x_i, conditional on a slightly corrupted

[3] In practice, this means that BERT pre-training only uses $\sim 15\%$ of the tokens in $\mathbf{x}$ as a source of pre-training supervision. In contrast, the unidirectional LM objective in Eq. 4.2 predicts *all* tokens in $\mathbf{x}$ (albeit in a left-to-right fashion), and thus uses all 100% of tokens as pre-training supervision. Recent work, such as Electra (Clark et al., 2020), has proposed pre-training approaches that both: (i) leverage all of the pre-training tokens as a source of pre-training supervision, which results in more efficient learning; and (ii) incorporate both the left and right context in order to derive a contextual representation of each token $x_i \in \mathbf{x}$.

Figure 5.1: An example of the BERT masked language modelling task. The true identity of the "*[MASK]*" token is the verb "*chase*", and "*window*" is an attractor noun. We suppress syntactic annotations and potential corruptions on the context tokens (§5.3) for clarity.

version of the entire sequence $\mathbf{x}$, which includes tokens that are *both* to the left (*i.e.,* $\mathbf{x}_{<i}$), and also to the right (*i.e.,* $\mathbf{x}_{>i}$) of the current masked token x_i.

5.4 Approach

In this section, we introduce our knowledge distillation approach, which aims to inject syntactic inductive biases into a Transformer-based BERT model that works well at scale. To account for the bidirectional nature of the BERT student (§5.3), this is accomplished by distilling the approximated posterior distribution of an explicitly hierarchical (albeit much harder to scale) RNNG teacher, which operates in a unidirectional, left-to-right fashion.

5.4.1 Motivation

In Ch. 4, we have demonstrated that distilling the predictions of an explicitly hierarchical RNNG can improve the syntactic competence of a sequential DSA-LSTM student model that learns from a larger dataset, even when the RNNG teacher itself is trained on less data. Despite this success, we conjecture that a direct application of the KD technique that we proposed in Ch. 4—which has been shown to work well for a DSA-LSTM student that operates in a unidirectional, left-to-right fashion—would not necessarily work well in the case of a BERT student model. We attribute this to a key discrepancy between the RNNG teacher and the BERT student: Concretely, here the BERT student is *not* a unidirectional language model like the RNNG teacher or the DSA-LSTM, but rather a bidirectional encoder that models a collection of conditionals for words in bidirectional context (Eq. 5.1).

Here we introduce two strategies for dealing with this challenge. The first, naïve approach is to simply *ignore* this discrepancy, and apply the exact same technique that we did in Ch. 4 (specifically, Eq. 4.6). Under this approach, we simply aim to minimise the cross-entropy between the RNNG teacher's marginal next-word distribution based on the left context alone, *i.e.,* $t_{\psi}(w \mid \mathbf{x}_{<i})$ for each $w \in \Sigma$, and the BERT student's distribution for predicting w as the masked word, conditional on the (slightly corrupted) bidirectional context $c(\mathbf{x}_{<i}, x_i, \mathbf{x}_{>i})$.[4] While this approach is surprisingly effective (§5.5.3), we illustrate a key issue with this approach in Fig. 5.1, where we consider a simple example "*The dogs by the window [MASK=chase] the cat*". In this example, the RNNG's strong syntactic biases mean that we can expect $t_{\psi}(w \mid \textit{The dogs by the window})$ to assign high probabilities to plural verbs like "*bark*", "*chase*", "*fight*", and "*run*" that are consistent with the agreement controller "*dogs*"—despite the presence of a singular attractor (Linzen et al., 2016, Ch. 3), "*window*". Nevertheless, some plural verbs that are favoured based on the left context alone, such as "*bark*" and "*run*", are in fact poor alternatives when considering the right context (*e.g.,* the sentence "**The dogs by the window <u>bark/run</u> the cat*" is syntactically illicit).[5] In other words, distilling the RNNG's estimate of $t_{\psi}(w \mid \mathbf{x}_{<i})$ fails to take into account the right context $\mathbf{x}_{>i}$ that is accessible to the BERT student, and runs the risk of encouraging BERT to assign high probabilities for words that fit poorly with the bidirectional context—such as the verbs "*bark*" and "*run*"—which are syntactically illicit for the example in Fig. 5.1.

Hence, our second approach is to learn from teacher distributions that not only: (i) reflect the strong syntactic biases of RNNGs, but also (ii) consider *both* the left and right context for predicting the masked word. Formally, we distill the RNNG's marginal distribution over words in bidirectional context, $t_{\psi}(w \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$, henceforth referred to as the **posterior** probability for generating w as the masked word under *all* available information. We now demonstrate that this quantity can, in fact, be computed from left-to-right LMs like RNNGs.

5.4.2 Posterior Inference

Given a *pre-trained* unidirectional, left-to-right LM that factorises $t_{\psi}(\mathbf{x}) = \prod_{i=1}^{T} t_{\psi}(x_i \mid \mathbf{x}_{<i})$, we discuss how to infer an estimate of $t_{\psi}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$. By definition of conditional proba-

[4]Note that, for each value $i \in \{1, \cdots, |\mathbf{x}|\}$, the sequence $\mathbf{x}$ can be decomposed as $[\mathbf{x}_{<i}, x_i, \mathbf{x}_{>i}]$, where $\mathbf{x}_{<1}$ and $\mathbf{x}_{>|\mathbf{x}|}$ are represented by the empty string.

[5]The verbs "*bark*" and "*run*"—as they occur within the context of the example in Fig. 5.1—are intransitive verbs that do not take any objects.

bilities:[6]

$$
\begin{aligned}
t_{\psi}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i}) &= \frac{t_{\psi}(\mathbf{x}_{<i}, x_i, \mathbf{x}_{>i})}{\sum_{w \in \Sigma} t_{\psi}(\mathbf{x}_{<i}, \tilde{x}_i = w, \mathbf{x}_{>i})}, && \text{(definition of conditional prob.)} \\
&= \frac{t_{\psi}(\mathbf{x}_{<i})\, t_{\psi}(x_i \mid \mathbf{x}_{<i})\, t_{\psi}(\mathbf{x}_{>i} \mid x_i, \mathbf{x}_{<i})}{t_{\psi}(\mathbf{x}_{<i}) \sum_{w \in \Sigma} t_{\psi}(w \mid \mathbf{x}_{<i})\, t_{\psi}(\mathbf{x}_{>i} \mid \tilde{x}_i = w, \mathbf{x}_{<i})}, && \text{(chain rule decomposition)} \quad (5.2) \\
&= \frac{t_{\psi}(x_i \mid \mathbf{x}_{<i}) \prod_{j=i+1}^{T} t_{\psi}(x_j \mid \mathbf{x}_{<j})}{\sum_{w \in \Sigma} t_{\psi}(w \mid \mathbf{x}_{<i}) \prod_{j=i+1}^{T} t_{\psi}(x_j \mid \tilde{\mathbf{x}}_{<j}(w, i))}, && \text{(cancelling common factors)}
\end{aligned}
$$

where $\tilde{\mathbf{x}}_{<j}(w, i) = [\mathbf{x}_{<i}; w; \mathbf{x}_{i+1:j-1}]$ denotes an alternate left context where x_i is replaced by $w \in \Sigma$.

Intuition. After cancelling common factors $t_{\psi}(\mathbf{x}_{<i})$, the posterior computation in Eq. 5.2 is decomposed into two terms: (i) the likelihood of producing x_i given its prefix, *i.e.,* $t_{\psi}(x_i \mid \mathbf{x}_{<i})$; and (ii) conditional on the fact that we have generated x_i and its prefix $\mathbf{x}_{<i}$, the likelihood of producing the observed continuations $\mathbf{x}_{>i}$, *i.e.,* $t_{\psi}(\mathbf{x}_{>i} \mid x_i, \mathbf{x}_{<i})$. In our running example (Fig. 5.1), the posterior would assign low probabilities to plural verbs like "*bark*" that are nevertheless probable under the left context alone (*i.e.,* $t_{\psi}($*bark* | *The dogs by the window*$)$ would likely be high), because they are unlikely to generate the continuations $\mathbf{x}_{>i}$ (*i.e.,* we expect $t_{\psi}($*the cat* | *The dogs by the window bark*$)$ to be low since it is syntactically illicit). In contrast, the posterior would assign high probabilities to plural verbs like "*fight*" and "*chase*" that are consistent with the bidirectional context, since we expect both $t_{\psi}($*fight* | *The dogs by the window*$)$ and $t_{\psi}($*the cat* | *The dogs by the window fight*$)$ to be probable.

Computational cost. While we *can* infer an estimate of the posterior $t_{\psi}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$, even from left-to-right LMs like RNNGs, in practice it is infeasible to exactly compute the RNNG's posterior for every single masked token x_i in the dataset D, which is necessary for the RNNG knowledge distillation procedure (see §5.4.5 for a high-level overview of the proposed KD approach). Concretely, let T denote the maximum length of $\mathbf{x}$. Our KD approach requires computing the posterior distribution (Eq. 5.2) for every masked token x_i in the dataset D, which (excluding marginalisation cost over $\mathbf{y}$) necessitates $O(|\Sigma| * T * |D|)$ operations, where each operation returns the RNNG's estimate of $t_{\psi}(x_j \mid \mathbf{x}_{<j})$. In the BERT setup,[7] this procedure leads to prohibitively many operations ($\sim 5 * 10^{+16}$).

[6] We assume that $\mathbf{x}$ is a fixed-length sequence, and aim to infer the LM's estimate for generating a *single* token x_i, relative to all other words $w \in \Sigma$ (denominator in Eq. 5.2), conditional on the bidirectional context.

[7] In BERT pre-training, $|\Sigma| \approx 29,000$ (vocabulary size of BERT-cased), $|D| \approx 3 * 10^9$, and $T = 512$.

5.4.3 Posterior Approximation

Since exact inference of the posterior is computationally expensive, we introduce an efficient yet effective approximation procedure. First, we conduct the following approximation: $t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid x_i, \mathbf{x}_{<i}) \approx t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid x_i)$; in other words, we assume that the RNNG teacher's predictions for the right context $\mathbf{x}_{>i}$ would only depend on the current masked token x_i, *independently* of the preceding tokens $\mathbf{x}_{<i}$, which would otherwise constitute a part of the full prefix. Substituting this approximation to Eq. 5.2 yields:[8]

$$t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i}) \approx \frac{t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{<i})\, t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid x_i)}{\sum_{w \in \Sigma} t_{\boldsymbol{\psi}}(w \mid \mathbf{x}_{<i})\, t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid w)}. \tag{5.3}$$

While Eq. 5.3 is still expensive to compute (*i.e.,* it still has the same asymptotic complexity as Eq. 5.2), this approximation enables us to apply the Bayes rule to compute $t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid x_i)$:

$$t_{\boldsymbol{\psi}}(\mathbf{x}_{>i} \mid x_i) = \frac{t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{>i})\, t_{\boldsymbol{\psi}}(\mathbf{x}_{>i})}{q(x_i)}, \tag{5.4}$$

where $q(\cdot)$ denotes the unigram distribution.[9] For the sake of efficiency, we replace $t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{>i})$ with a separately trained "reverse", **right-to-left** RNNG, denoted as $r_{\boldsymbol{\omega}}(x_i \mid \mathbf{x}_{>i})$. We now apply Eq. 5.4 and the right-to-left RNNG parameterisation $r_{\boldsymbol{\omega}}(x_i \mid \mathbf{x}_{>i})$ into Eq. 5.3:

$$\begin{aligned} t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{<i}, \mathbf{x}_{>i}) &\approx \frac{\dfrac{t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{<i})\, r_{\boldsymbol{\omega}}(x_i \mid \mathbf{x}_{>i})\, t_{\boldsymbol{\psi}}(\mathbf{x}_{>i})}{q(x_i)}}{\sum_{w \in \Sigma} \dfrac{t_{\boldsymbol{\psi}}(w \mid \mathbf{x}_{<i})\, r_{\boldsymbol{\omega}}(w \mid \mathbf{x}_{>i})\, t_{\boldsymbol{\psi}}(\mathbf{x}_{>i})}{q(w)}}. \\ &= \frac{\dfrac{t_{\boldsymbol{\psi}}(x_i \mid \mathbf{x}_{<i})\, r_{\boldsymbol{\omega}}(x_i \mid \mathbf{x}_{>i})}{q(x_i)}}{\sum_{w \in \Sigma} \dfrac{t_{\boldsymbol{\psi}}(w \mid \mathbf{x}_{<i})\, r_{\boldsymbol{\omega}}(w \mid \mathbf{x}_{>i})}{q(w)}} \qquad \text{(cancelling common factors)} \end{aligned} \tag{5.5}$$

Our approximation in Eq. 5.5 crucially reduces the required number of operations from $O(|\Sigma| * T * |D|)$ to $O(|\Sigma| * |D|)$, although the actual speedup is much more substantial in practice, since Eq. 5.5 involves easily batched operations that considerably benefit from

[8]This approximation preserves the intuition explained in §5.4.2. Concretely, verbs like *bark* would also be assigned low probabilities under this posterior approximation, since we expect $t_{\boldsymbol{\psi}}(\textit{the cat} \mid \textit{bark})$ to be low as it is syntactically illicit, while the alternative "*bark **at** the cat*" would have been syntactically licit.

[9]The unigram distribution is the *non-contextual* probability of generating a particular word in a given corpus. In its simplest form, this distribution can be estimated by counting the number of times a particular word w appears in a training corpus D, and divide it with the total number of word tokens in that same corpus, *i.e.,* $q(w) = \dfrac{\text{count}(w, D)}{\sum_{w' \in \Sigma} \text{count}(w', D)}$.

specialised hardwares like GPUs.[10] We remark that our proposed approach here is a general one: It can approximate the posterior over x_i from *any* left-to-right LM (including from powerful recent left-to-right LMs like GPT-3), which can then be used as a learning signal for BERT through KD, irrespective of the LM's parameterisation. It does, however, necessitate a separately trained right-to-left LM, which doubles the cost of the teacher model pre-training.

Connection to a product of experts. Eq. 5.5 has a similar form to a product of experts (Hinton, 2002, PoE) between the left-to-right and right-to-left RNNGs' next-word distributions, albeit with extra unigram terms $q(w)$. If we replace the unigram distribution with a uniform one, *i.e.,* by assuming $q(w) = 1/|\Sigma| \; \forall w \in \Sigma$, Eq. 5.5 reduces to a standard PoE.

Approximating the marginal. Our proposed approximation in Eq. 5.5 requires obtaining estimates of $t_{\boldsymbol{\psi}}(x_i \,|\, \mathbf{x}_{<i})$ and $r_{\boldsymbol{\omega}}(x_i \,|\, \mathbf{x}_{>i})$ from the left-to-right and right-to-left RNNGs, respectively, which necessitate an intractable marginalisation procedure over all possible tree prefixes $\mathbf{y}_{<i}$ (for the left-to-right RNNG) and $\mathbf{y}_{>i}$ (for the right-to-left RNNG). To circumvent this issue, we apply the same marginal probability approximation as we did in §4.3.6, although here we rely on the one-best predictions of an off-the-shelf, accurate Berkeley phrase-structure parser of Fried et al. (2019). Concretely, let $\hat{\mathbf{y}}^{\text{berk}}(\mathbf{x})$ be the Berkeley parser's one-best predicted phrase-structure tree for an input sentence $\mathbf{x}$. Furthermore, let $\hat{\mathbf{y}}^{\text{berk}}_{<i}(\mathbf{x})$ be all the non-terminal symbols in $\hat{\mathbf{y}}^{\text{berk}}(\mathbf{x})$ that occur before the i-th token x_i. In line with our approach in §4.3.6, we approximate the marginal next-word distribution from the left-to-right RNNG teacher as follows:[11]

$$\begin{aligned} t_{\boldsymbol{\psi}}(x_i \,|\, \mathbf{x}_{<i}) &= \sum_{\mathbf{y}_{<i} \in Y(\mathbf{x}_{<i})} t_{\boldsymbol{\psi}}(x_i, \mathbf{y}_{<i} \,|\, \mathbf{x}_{<i}) \\ &= \sum_{\mathbf{y}_{<i} \in Y(\mathbf{x}_{<i})} t_{\boldsymbol{\psi}}(\mathbf{y}_{<i} \,|\, \mathbf{x}_{<i}) \; t_{\boldsymbol{\psi}}(x_i \,|\, \mathbf{x}_{<i}, \mathbf{y}_{<i}) \quad \text{(chain rule)} \\ &\approx t_{\boldsymbol{\psi}}(x_i \,|\, \mathbf{x}_{<i}, \hat{\mathbf{y}}^{\text{berk}}_{<i}(\mathbf{x})). \qquad \text{(approximation in §4.3.6)} \end{aligned} \tag{5.6}$$

[10]We remark that Eq. 5.5 can be computed in an efficient way, where we simply sum up the **logits** (*i.e.,* the output layer prior to the softmax normalisation, §2.2) for predicting the word at index i based on: (i) the left-to-right RNNG's logits for the next word, and (ii) the right-to-left RNNG's logits for the next word. After summing the two RNNGs' logits vectors, we then substract the log of the unigram count of each word w. We then simply run the resulting vector through a softmax activation, which results in the exact same normalised probability distribution (albeit much faster to compute on GPUs) than directly calculating Eq. 5.5.

[11]The caveats of this approximation, such as the non-incremental nature of the Berkeley parser, are discussed in more detail in §4.3.6. Note that the non-incremental nature of this approximation is even less of an issue here than in Ch. 4, as the BERT student is non-incremental and has access to the full bidirectional context.

We approximate $r_{\boldsymbol{\omega}}(x_i \mid \mathbf{x}_{>i})$ from the right-to-left RNNG teacher in a similar fashion.

Preliminary Experiments. Before proceeding with the KD experiments, we begin by asking: How effective is our proposed posterior approximation technique? In other words, how much speed-up can we obtain with this approach, compared to exact inference in Eq. 5.2? And how large is the performance difference between our proposed posterior approximation and exact inference? To answer these questions, we conduct preliminary language modelling experiments on the Penn Treebank (Marcus et al., 1993, PTB) dataset. We defer a full summary of the preliminary experimental setup to Appendix B, but remark on two key findings. First, we find that our posterior approximation is much faster than exact inference by a factor of more than 50,000, which demonstrates the **efficiency** of the proposed approach. Second, despite this substantial speed-up, our proposed approximation still achieves a competitive average posterior negative log likelihood compared to exact inference (2.68 as opposed to 2.5 for exact inference), as shown in Table B.1 in Appendix B. Hence, above and beyond the significant gains in terms of inference speed, this finding further attests to the **effectiveness** of our proposed posterior approximation procedure.

Differences Between the Models. We now empirically validate our motivating intuition in Fig. 5.1: A model that takes into account the bidirectional context (as is the case for our proposed posterior approximation in Eq. 5.5) *should* make different predictions compared to the unidirectional left-to-right and right-to-left models.[12] To ascertain whether this is truly the case, we compute the mean Kullback-Leibler (KL) divergence between the distributions from the proposed posterior approximation (Eq. 5.5) and the distributions from: (i) the left-to-right model, (ii) the right-to-left model, and (iii) a simple product of experts baseline (*i.e.,* Eq. 5.5, but where $q(w)$ is the uniform distribution). The findings in Table 5.1 suggest that our proposed posterior approximation approach indeed yields quantifiably different distributions from the left-to-right and right-to-left baselines. To a lesser extent, it also differs from a simple product of experts baseline that similarly incorporates *both* the left-to-right and right-to-left models' predictions, albeit where $q(w)$ is the uniform distribution.

[12]We conduct this experiment using the exact same empirical setup as the **Preliminary Experiments**.

Model	KL Divergence with Our Posterior Approximation
Left-to-right LM	2.27±1.84
Right-to-left LM	2.04±1.87
Product of Experts	1.12±1.08

Table 5.1: Preliminary experiments reporting the *mean* ± *stdev.* of the KL divergence (in nats) between the proposed posterior approximation (Eq. 5.5) and: (i) the left-to-right LM, (ii) the right-to-left LM, and (iii) a simple product of experts baseline (Eq. 5.5, albeit where $q(w)$ is the uniform (rather than unigram) distribution).

5.4.4 Objective Function

Overall, the aim of our proposed knowledge distillation pre-training is to find BERT parameters $\hat{\boldsymbol{\theta}}_{\text{KD}}$ that best emulate our approximation of the RNNG's posterior distribution $t_{\boldsymbol{\psi}}(w \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$ through a similar word-level cross-entropy loss (Hinton et al., 2015; Kim and Rush, 2016; Furlanello et al., 2018, *inter alia*) that we used in Ch. 4. Formally:

$$\hat{\boldsymbol{\theta}}_{\text{KD}} = \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D|} \sum_{\mathbf{x} \in D} \underbrace{\sum_{i \in M(\mathbf{x})} \sum_{w \in \Sigma} \tilde{t}_{\boldsymbol{\psi},\boldsymbol{\omega}}(w \mid \mathbf{x}_{<i}, \mathbf{x}_{>i}) \, \log p_{\boldsymbol{\theta}} \left(\tilde{x}_i = w \mid c(x_1), \cdots, c(x_T) \right)}_{\ell_{\text{KD}}(\mathbf{x};\boldsymbol{\theta})}, \tag{5.7}$$

where $\tilde{t}_{\boldsymbol{\psi},\boldsymbol{\omega}}(w \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$ denotes our proposed approximation of the posterior distribution $t_{\boldsymbol{\psi}}(w \mid \mathbf{x}_{<i}, \mathbf{x}_{>i})$, as defined in Eqs. 5.5 and 5.6.

Interpolation. As in Ch. 4, we interpolate the RNNG knowledge distillation loss (which provides a source of syntactic inductive biases for the BERT student) with the standard BERT masked language modelling loss, which enables the student model to fully exploit the rich common-sense and semantics knowledge contained in large text corpora by virtue of predicting the true identity of the masked token x_i.[13] Formally:

$$\hat{\boldsymbol{\theta}}_{\text{B-KD}} = \arg\min_{\boldsymbol{\theta}} -\frac{1}{|D|} \sum_{\mathbf{x} \in D} \left[\alpha \, \ell_{\text{KD}}(\mathbf{x};\boldsymbol{\theta}) \;+\; (1-\alpha) \sum_{i \in M(\mathbf{x})} \log p_{\boldsymbol{\theta}} \left(x_i \mid c(x_1), \cdots, c(x_T) \right), \right] \tag{5.8}$$

omitting the next-sentence prediction for brevity. We henceforth set $\alpha = 0.5$.

[13] Recall that the KD loss $\ell_{\text{KD}}(\mathbf{x};\boldsymbol{\theta})$ does not take into account the true identity of the masked word x_i.

5.4.5 Putting It All Together

We provide a high-level overview of our structure distillation BERT pre-training procedure, which is accomplished by the following three simple steps.

- In the first step, we train the two RNNG teachers on as much data as we feasibly can. To that end, we separately train one RNNG teacher that works in a left-to-right fashion, and another that works in a right-to-left fashion, which is necessary for our posterior approximation procedure in Eq. 5.5. In practice, due to the high computational costs of RNNGs and the enormity of the BERT pre-training corpora, we are only able to train the RNNG teachers on a small 3% subset of the entire BERT pre-training dataset. For simplicity, we train the left-to-right and right-to-left RNNGs on the exact same 3% subset.
- In the second step, once the RNNG teacher has finished training and assuming that the entire BERT pre-training dataset has been parsed with an off-the-shelf parser, we calculate the approximated posterior distribution for each masked word x_i in the whole training set. As shown in Eq. 5.5, we compute this approximation based on: (i) the next-word distribution of the pre-trained, *left-to-right* RNNG teacher, (ii) the next-word distribution of the pre-trained, *right-to-left* RNNG teacher, and (iii) the unigram distribution for generating each word $w \in \Sigma$ independently of the context. The next-word marginal distribution of each unidirectional RNNG teacher is approximated using Eq. 5.6.
- In the third and final step, we train the BERT student on an interpolation of the KD loss that aims to emulate the RNNG teacher's approximated posterior distribution for generating the masked word, in addition to the standard BERT masked language modelling loss. This interpolated objective aims to combine the best of both syntactic biases and data scale.

5.5 Experiments

Here we describe our evaluation setup, and present and discuss our key findings.

5.5.1 Evaluation Tasks and Setup

We conjecture that the improved syntactic competence from our approach would benefit a broad range of tasks that involve structured output spaces, including tasks that are not explicitly syntactic in nature. We thus evaluate our structure-distilled BERTs on six diverse structured prediction tasks that encompass syntactic, semantic, and coreference resolution tasks, in addition to the GLUE benchmark that is largely comprised of classification tasks.

Phrase-structure parsing - PTB. We first evaluate our model on phrase-structure parsing on the WSJ section of the PTB. Following prior work, we use sections 02-21 for training, section 22 for validation, and section 23 for testing. We apply our approach on top of the BERT-augmented in-order (Liu and Zhang, 2017) transition-based parser of Fried et al. (2019), which approaches the current state of the art performance. Since the RNNG teacher that we distill into BERT also makes its predictions through recursive compositions that are based on phrase-structure trees, this setup is related to self-training (Yarowsky, 1995; Charniak, 1997; Zhou and Li, 2005; McClosky et al., 2006; Andor et al., 2016, *inter alia*).

Phrase-structure parsing - OOD. Still in the context of phrase-structure parsing, we evaluate how well our approach generalises to three out-of-domain (OOD) treebanks: Brown (Francis and Kučera, 1979), Genia (Tateisi et al., 2005), and the English Web Treebank (Petrov and McDonald, 2012). Following Fried et al. (2019), we test the PTB-trained parser on the test splits[14] of these OOD treebanks *without* any retraining, in order to simulate the case where no in-domain labelled data are available. We use the same codebase as above.

Dependency parsing - PTB. Our third task is PTB dependency parsing with Stanford Dependencies (De Marneffe and Manning, 2008) v3.3.0. We use the BERT-augmented joint phrase-structure and dependency parser of Zhou and Zhao (2019), which is inspired by head-driven phrase-structure grammar (Pollard and Sag, 1994, HPSG). As the RNNG teacher that we distill into BERT is based on phrase-structure grammar, the dependency parsing task assesses whether the syntactic inductive biases that are derived from phrase-structure grammar would generalise well to a different, dependency grammar formalism.

Semantic role labelling. Our fourth evaluation task is span-based semantic role labelling (SRL) on the English CoNLL 2012 (OntoNotes) dataset (Pradhan et al., 2013). We apply our approach on top of the BERT-augmented model of Shi and Lin (2019), as implemented on AllenNLP (Gardner et al., 2018). As the SRL task goes beyond identifying syntactic

[14]We use the Brown test split of Gildea (2001), the Genia test split of McClosky et al. (2008), and the EWT test split from SANCL 2012 (Petrov and McDonald, 2012).

information, it remains an open question whether—and to what extent—more explicit syntactic inductive biases can help improve performance on this task.

Coreference resolution. Our fifth evaluation task is coreference resolution on the English OntoNotes (Pradhan et al., 2012). We use the BERT-augmented model of Joshi et al. (2019), which extends the higher-order coarse-to-fine model of Lee et al. (2018). Similar to the SRL task, the coreference resolution task is not explicitly syntactic in nature, thus constituting an interesting testbed for investigating whether syntactic biases are also helpful for these tasks.

CCG Supertagging Probe. All our proposed evaluation tasks thus far necessitate either fine-tuning the entire BERT model, or training a task-specific model on top of the BERT embeddings. Hence, it remains unclear how much of the gains are due to better syntactic representations from our new *pre-training* strategy, as opposed to the available supervision at the *fine-tuning* stage. To better understand the gains from our approach, we additionally evaluate on combinatory categorial grammar (Steedman, 2000, CCG) supertagging (Bangalore and Joshi, 1999; Clark and Curran, 2007) through a **classifier probe** (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017, *inter alia*), where no BERT fine-tuning takes place.[15]

CCG supertagging is a compelling probing task because it necessitates an understanding of bidirectional context information; furthermore, the per-word classification setup lends itself well to classifier probes.[16] Nevertheless, it remains unclear how much of the accuracy can be attributed to the information encoded in the representation, as opposed to the classifier probe itself. We thus adopt the **control task** protocol of Hewitt and Liang (2019) that assigns each word type to a random control category,[17] which assesses the memorisation capacity of the classifier. In addition to the probing accuracy, we report the probe *selectivity*,[18] where higher selectivity denotes probes that faithfully rely on the linguistic knowledge encoded in the representation. Going forward, we use linear classifiers to maintain high selectivities.

[15] A similar CCG probe was explored by Liu et al. (2019a), with comparable results relative to our baseline.

[16] Another alternative for conducting probing evaluation on CCG is to use the resulting (frozen) BERT contextualised word representation as features for an off-the-shelf CCG parser. Here we choose to focus on supertagging due to its simplicity and per-word evaluation setup, and leave this extension to future work.

[17] Following prior work, we set the cardinality of this control category to the number of supertags.

[18] A probe's selectivity is defined as the difference between the probing task accuracy and the control task accurary, which controls for the classifier probe's memorisation ability.

Commonality. All our structured prediction experiments are conducted on top of publicly available repositories of BERT-augmented models, with the exception of the CCG supertagging task that we evaluate as a probe. This setup means that obtaining our results is as simple as changing the pre-trained BERT weights in those repositories to our pre-trained structure-distilled BERT, and applying the exact same steps as for fine-tuning the standard BERT model. We summarise the fine-tuning hyper-parameters for each task in Appendix B.

GLUE Benchmark. Above and beyond the 6 structured prediction tasks described above, we evaluate our approach on the classification[19] tasks of the GLUE benchmark, with the exception of the Winograd NLI (Levesque et al., 2012) for consistency with the original BERT paper (Devlin et al., 2019). We use the same BERT GLUE fine-tuning hyper-parameters as the prior work of Joshi et al. (2020), which are summarised in Appendix B.

5.5.2 Experimental Setup and Baselines

Here we describe the key aspects of our empirical setup, and outline the baselines for assessing the effectiveness of our approach.

RNNG extension to subwords. The RNNG model (§2.4)—as proposed by Dyer et al. (2016)—operates directly on top of *word* sequences, potentially rendering them incompatible with the BERT student, which operates at the more granular level of subword units (Sennrich et al., 2016). Hence, to facilitate compatibility with the BERT student, we propose an extension of the RNNG model that operates over subword units. As each word can be split into an arbitrary-length sequence of subwords, we preprocess the trees to include an additional non-terminal symbol that represents a word sequence, as illustrated by the example "(S (NP (WORD *the*) (WORD *d ##og*)) (VP (WORD *ba ##rk ##s*)))", where tokens prefixed by "*##*" denote subword units. While we find this formulation to be effective in practice, one downside of this formulation is that it makes the phrase-structure tree deeper (hence resulting in slower computation). Another alternative is to represent each phrase as a flat sequence of subwords (hence bypassing the need to have a separate "(WORD" non-terminal category),[20] although we find this alternative formulation to yield worse perplexity.

[19] This setup excludes the semantic textual similarity benchmark (STS-B), which is a regression task.

[20] An example of the alternative formulation that bypasses the need for special "(WORD" non-terminals is "(S (NP *the d ##og*) (VP *ba ##rk ##s*)).

RNNG Teacher. We implement the subword-augmented RNNG teachers on DyNet, and obtain "silver-grade" phrase-structure trees for the entire BERT training set using the near-state of the art parser of Fried et al. (2019). These trees are used to train the left-to-right and right-to-left RNNG teachers, and also to approximate the RNNG teachers' marginal next-word distribution at inference (Eq. 5.6). We use the same WordPiece tokenisation and vocabulary as BERT-Cased, and select the following hyper-parameters that achieved the best validation perplexity: 2-layer stack LSTMs (Dyer et al., 2015) with 512 hidden units per layer, optimised by standard mini-batch gradient descent with an initial learning rate of 0.5 that is decayed exponentially by a factor of 0.9 at every epoch after the tenth. We use variational dropout with $p = 0.3$, and a batch size of 10 sentences. We train both the left-to-right and right-to-left RNNGs with the exact same hyper-parameters and dataset, which here is a shared subset of 3.6M sentences ($\sim$3%) that are subsampled from the BERT training set. We train the two RNNG teacher models independently, each with automatic dynamic batching; overall, training each model takes three weeks on a V100 GPU.[21]

BERT Student. We apply our structure distillation pre-training approach to $\text{BERT}_{\text{BASE}}$-Cased, and explore the extension to $\text{BERT}_{\text{LARGE}}$-Cased in §5.5.3. For fair comparison, we use the exact same training dataset, model configuration, WordPiece tokenisation, vocabulary, and hyper-parameters (summarised in Appendix B) as in the standard pre-trained BERT model.[22] The sole exception is that we use a larger initial learning rate of $3e^{-4}$, which performs better on our preliminary experiments;[23] we apply this larger pre-training learning rate to all models (including the no distillation/standard BERT baseline) for fair comparison.

Baselines and comparisons. To assess the effectiveness of our proposed approach, we compare our structure-distilled BERT against the following comprehensive set of baselines:

- A standard $\text{BERT}_{\text{BASE}}$-Cased model that lacks any form of RNNG knowledge distillation loss. This model benefits from scalability at the expense of syntactic biases (**"No-KD"**);

[21]To our best knowledge, our RNNG training setup employs one of the largest RNNG training datasets to date (3.6M sentences and roughly 90M words), alongside the more recent RNNG model of Noji and Oseki (2021) that is trained on a 100M-word text dataset through more effective manual batching techniques (§4.7).

[22]`https://github.com/google-research/bert`.

[23]We find this larger learning to perform better on most of our evaluation tasks. Liu et al. (2019b) has similarly found that tuning BERT's initial learning rate leads to better results.

- Four variants of structure-distilled BERTs that: (i) only distill the left-to-right RNNG (**"L2R-KD"**), (ii) only distill the right-to-left RNNG (**"R2L-KD"**), (iii) distill the RNNG's approximated marginal for generating x_i under the bidirectional context, where $q(w)$ (Eq. 5.5) is the *uniform* distribution (**"UF-KD"**), and lastly (iv) a similar variant as (iii), but where $q(w)$ is the *unigram* distribution (**"UG-KD"**). All these BERT models crucially benefit from the syntactic biases of RNNGs, although only variants (iii) and (iv) learn from teacher distributions that consider the *bidirectional context* for predicting x_i; and
- A $\text{BERT}_{\text{BASE}}$ that distills the approximate posterior for generating x_i under the bidirectional context, but from *sequential* LSTM teachers (**"Seq-KD"**) instead of RNNGs.[24] This baseline crucially isolates the importance of learning from hierarchical teachers, as it employs the same approximation technique and KD loss as the structure-distilled BERTs.

Learning curves. With enough labelled data, BERT can acquire the relevant syntactic information from the fine-tuning (as opposed to pre-training) stage, although we argue that better pre-trained representations should facilitate more *sample-efficient* generalisations (Yogatama et al., 2019). Hence, we also examine the models' fine-tuning learning curves, as a function of varying amounts of training data, on phrase-structure parsing and SRL.

Random seeds. Since fine-tuning the exact same pre-trained BERT from different random seeds can nevertheless lead to varying results, we report the mean performance from three random seeds on the structured prediction tasks, and from five random seeds on GLUE.

Test results. To preserve the integrity of the test sets, we first report all performance on the validation sets, and only report the test set results for: (i) the **No-KD** baseline, and (ii) the best structure-distilled model on the validation set (**"Best-KD"**).

5.5.3 Findings and Discussion

We report the validation and test set results for the structured prediction tasks in Table 5.2. Furthermore, the validation set learning curves for phrase-structure parsing and SRL, which compare the **No-KD** baseline with the **UG-KD** variant, are provided in Fig. 5.2.

[24] For fair comparison, we train the LSTM on the same subset as the RNNG, and with a comparable model size. An alternative here is to use Transformers, although we elect to use LSTMs to facilitate a fair comparison with RNNGs, as RNNGs are built on top of LSTM architectures (albeit with explicit composition operations).

Task		Validation Set: Baselines: No-KD	Seq-KD	Structure-distilled BERTs: L2R-KD	R2L-KD	UF-KD	UG-KD	Test Set: No-KD	Best-KD	Err. Red.
Parsing	Const. PTB - F1	95.38	95.33	95.55	95.55	95.58	**95.59**	95.35	**95.70**	7.6%
	Const. PTB - EM	55.33	55.41	55.92	56.18	56.39	**56.59**	55.25	**57.77**	5.63%
	Const. OOD - F1†	86.76	86.54	87.43	**87.53**	87.23	87.40	89.04	**89.76**	6.55%
	Dep. PTB - UAS	96.48	96.40	**96.70**	96.64	96.60	96.66	96.79	**96.86**	2.18%
	Dep. PTB - LAS	94.65	94.56	**94.90**	94.80	94.79	94.83	95.13	**95.23**	1.99%
SRL - OntoNotes		86.17	86.09	86.34	86.29	86.30	**86.46**	86.08	**86.39**	2.23%
Coref. - OntoNotes		72.53	69.27	73.74	73.49	**73.79**	73.33	72.71	**73.69**	3.58%
CCG supertag. probe		93.69	91.59	93.97	**95.21**	95.13	**95.21**	93.88	**95.2**	21.57%
Probe selectivity		24.79	23.77	23.3	23.57	27.28	**28.3**	23.15	**26.07**	N/A

Table 5.2: Validation and test results for the six diverse structured prediction tasks, where each entry reflects the mean of three random seeds. To preserve the integrity of the test set, we only obtain the test set results for the no distillation baseline and the best structure-distilled BERT on the validation set; **"Err. Red."** reports the test error reductions *relative* to the **No-KD** baseline. We report both F1 and exact match (EM) scores for PTB phrase-structure parsing; for dependency, we report both unlabelled (UAS) and labelled (LAS) attachment scores. The "Const. OOD" (†) row indicates the mean F1 from three out-of-domain corpora: Brown, Genia, and the English Web Treebank (EWT), although the validation results exclude the Brown Treebank that has no corresponding validation set.

General discussion. We summarise several key observations from Table 5.2 and Fig. 5.2.

- All four structure-distilled BERT models consistently outperform the **No-KD** baseline, including the **L2R-KD** and **R2L-KD** variants that only distill the syntactic knowledge of unidirectional RNNGs. Remarkably, this pattern holds true for *all six* structured prediction tasks. In contrast, we observe no such gains for the **Seq-KD** baseline, which largely performs worse than the **No-KD** model. Hence, we conclude that the gains afforded by our approach can be properly attributed to the **syntactic biases** of the RNNG teacher.
- We conjecture that the surprisingly strong performance of the **L2R-KD** and **R2L-KD** models, which distill the knowledge of *unidirectional* RNNGs, can be attributed to the interpolated objective in Eq. 5.8 ($\alpha = 0.5$). This interpolation means that the target distribution assigns a probability mass of at least 0.5 to the true masked word x_i, which is guaranteed to be consistent with the bidirectional context. However, the syntactic knowledge contained in the unidirectional RNNGs' predictions can still provide a syntactically informative learning signal, via the rest of the probability mass, for the BERT student.
- While all four structure-distilled BERT variants are able to outperform the **No-KD** baseline, models that distill our approximation of the RNNG teachers' distribution for words in bidirectional context (**UF-KD** and **UG-KD**) yield the best results on four out

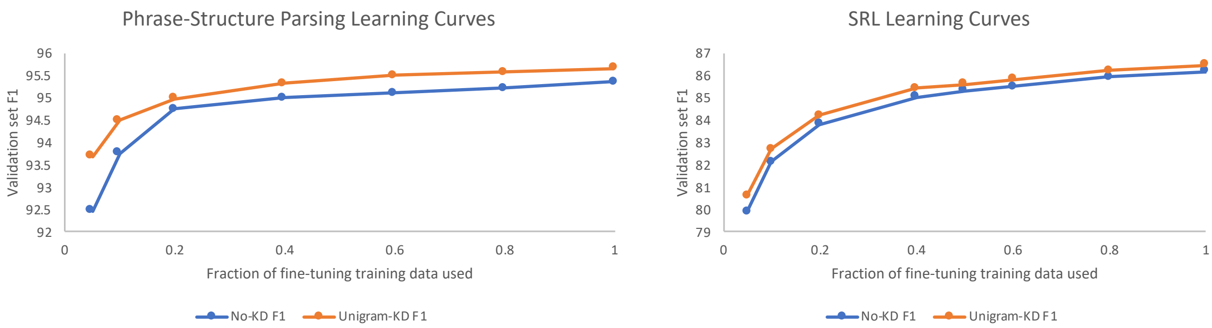


Figure 5.2: The fine-tuning learning curves that examine how the number of fine-tuning instances (from 5% to 100% of the full training sets) affects validation set F1 scores in the case of phrase-structure parsing (explicitly syntactic) and SRL (not explicitly syntactic). We compare the **No-KD**/standard $\text{BERT}_{\text{BASE}}$-Cased and the **UG-KD** structure-distilled BERT.

of the six structured prediction tasks (PTB phrase-structure parsing, SRL, coreference resolution, and the CCG supertagging probe). This finding thus confirms our hypothesis on the importance of teacher distributions that consider the bidirectional context, and further demonstrates the effectiveness of our proposed posterior approximation approach.

- Overall, we observe the largest gains for syntactic tasks, particularly for phrase-structure parsing and CCG supertagging. However, the improvements are *not* at all confined to purely syntactic tasks: We reduce relative error from strong BERT baselines by 2.2% and 3.6% on SRL and coreference resolution, respectively. While the RNNG's syntactic biases are derived from phrase-structure grammar, the strong improvement on CCG supertagging, in addition to the smaller improvement on dependency parsing, suggests that the RNNG's syntactic biases generalise well across different grammar formalisms.

- We observe larger improvements in a low-resource scenario, where the model is exposed to fewer fine-tuning instances (Fig. 5.2), suggesting that syntactic biases are helpful for enabling more **sample-efficient** generalisations. This pattern holds for both tasks that we investigated: phrase-structure parsing (syntactic in nature) and SRL (not explicitly syntactic in nature). With only 5% of the fine-tuning data, the **UG-KD** model improves F1 score from 79.9 to **80.6** for SRL (a 3.5% error reduction relative to the **No-KD** baseline, as opposed to 2.2% on the full data). For phrase-structure parsing, the **UG-KD** model achieves a remarkable **93.68** F1 (a 16% relative error reduction, as opposed to 7.6% on the full data) with only 5% of the PTB. Remarkably, this performance is better than past state of the art parsers trained on the *full* PTB c. 2017 (Kuncoro et al., 2017).

	No-KD	UG-KD
Validation Set (Per-task average / 1-best random seed)		
CoLA	50.7 / 60.2	**54.3 / 60.6**
7-task avg. (excl. CoLA)	**85.4 / 87.8**	84.8 / 86.9
Overall 8-task avg.	**81.1 / 84.4**	81.0 / 83.6
Test set (Per-task 1-best random seed on validation set)		
CoLA	53.1	**55.3**
7-task avg. (excl. CoLA)	**84.2**	83.5
Overall 8-task avg.	**80.3**	80.0

Table 5.3: Summary of the validation and test set results on GLUE. The validation set results are derived from the average of five random seeds for each task, which accounts for variance, and the 1-best random seed, which does not. Meanwhile the test results are derived from the 1-best configuration (including random seed) that performs best on the validation set.

GLUE results and discussion. We report the GLUE validation and test results for $\text{BERT}_{\text{BASE}}$-Cased in Table 5.3. Since we observe a different pattern of results on the Corpus of Linguistic Acceptability (Warstadt et al., 2018, CoLA) than on the rest of GLUE, we henceforth report: (i) the CoLA results, (ii) the 7-task average that excludes CoLA, and (iii) the average score across all 8 tasks; the per-task GLUE breakdown is provided in Appendix B. We select the **UG-KD** model for the GLUE experiments, because it achieves the best validation set 8-task average performance among the four structure-distilled BERTs.

Based on the findings in Table 5.3, the results on GLUE provide an interesting contrast to the consistent improvements we observed on the structured prediction tasks. More concretely, our **UG-KD** model outperforms the baseline model on CoLA, but performs slightly worse on the other GLUE tasks in aggregate, leading to a slightly lower overall test set accuracy (80.0 for the **UG-KD** as opposed to 80.3 for the **No-KD** baseline). Indeed, the improvement on the syntax-sensitive CoLA benchmark provides additional evidence—beyond the consistent improvement on the explicitly syntactic structured prediction tasks in Table 5.2—that our approach indeed yields improved syntactic competence. We conjecture that these improvements do not transfer to the other GLUE tasks because they rely more on lexical and semantic properties, and less on syntactic competence (McCoy et al., 2019).

We defer a more thorough investigation of how much syntactic competence is necessary for solving most of the GLUE tasks to future work, but make two remarks. First, the findings on GLUE are consistent with the hypothesis that our approach yields improved syntactic competence, albeit at the expense of a slightly less rich meaning representation, which we

Sentence Input	**No-KD & L2R-KD Pred.**	**R2L-KD & UG-KD Pred.**
*"Apple II owners , for example , had to **use** their TV sets as screens and stored data on audiocassettes"*	(S[b]\NP)/NP	((S[b]\NP)/PP)/NP

Table 5.4: An example of the CCG supertag predictions for the verb "***use***", from the four different BERTs. The correct answer is "((S[b]\NP)/PP)/NP", which both the **R2L-KD** and the **UG-KD** predict correctly (blue). However, the **No-KD** baseline and the **L2R-KD** model produce (the same) incorrect predictions (red); concretely, both models fail to subcategorise for the prepositional phrase "*as screens*" as a dependent of "***use***". Beyond the supertag for the word "***use***", all four models predict the correct supertags for all other words (not shown).

attribute to the smaller dataset used to train the RNNG teachers. Second, human natural language understanding includes the ability to predict structured outputs, *e.g.,* to decipher "*who did what to whom*" as part of SRL. Crucially, succeeding in these tasks necessitates inference about structured output spaces, which (unlike most of GLUE) cannot be reduced to a single classification decision. Our findings indicate a partial dissociation between model performance on these two types of tasks; hence, supplementing GLUE evaluation with some of these structured prediction tasks can offer a more holistic assessment of progress in NLU.

CCG probe example. The CCG supertagging probe is a particularly compelling test bed, because it clearly assesses the model's ability to use bidirectional context information in making its syntactic predictions—*without* introducing additional confounds from the BERT fine-tuning procedure. We thus provide a representative example of the four different BERT variants' predictions for the CCG supertagging probe in Table 5.4, based on which we discuss two observations. First, the different models make different predictions, where the **No-KD** and **L2R-KD** models produce (coincidentally the same) incorrect predictions, whereas the **R2L-KD** and **UG-KD** models can predict the correct supertag. This finding suggests that different teacher models indeed impose different biases on the BERT students.[25]

Second, the mistakes of the **No-KD** and **L2R-KD** BERT models belong to the broader category of challenging argument-adjunct distinctions (Palmer et al., 2005; Fowlie, 2017). More concretely, both models fail to subcategorise for the prepositional phrase (PP) "*as screens*", which serves as an argument of the verb "***use***", as opposed to the noun phrase "*TV sets*". Indeed, distinguishing between these two potential dependencies naturally requires syntactic information from the right context; hence the **R2L-KD** BERT, which is trained

[25] All four BERTs have access to the full *bidirectional* context at test time, although some are trained to mimic the predictions of *unidirectional* RNNGs at pre-training time (**L2R-KD** and **R2L-KD**).

Pairwise Exact Match Agreement	No-KD	L2R-KD	R2L-KD	UF-KD	UG-KD
No-KD	-	36.20	36.56	38.01	37.05
L2R-KD	36.20	-	42.25	43.58	44.43
R2L-KD	36.56	42.25	-	39.95	41.53
UF-KD	38.01	43.58	39.95	-	44.92
UG-KD	37.05	44.43	41.53	44.92	-

Table 5.5: Pairwise model agreement scores for phrase-structure parsing, as measured by average exact match on trees from the validation set of the PTB, for which some pair of models produce different trees. Self-agreement (diagonals) are 100%. As exact match evaluation is symmetric in nature, the entries in the table are also symmetric.

to emulate the predictions of an RNNG teacher that observes the right context, is able to make the correct prediction. This advantage is crucially retained by the **UG-KD** model that distills the RNNG's approximate distribution over words in bidirectional context (Eq. 5.5), and further confirms the effectiveness of our proposed posterior approximation approach.

Measuring the Models' Differences. Beyond the qualitative example in Table 5.4, we further quantify the extent to which the different BERT models produce different predictions. To that end, we compute the *pairwise model agreement* in terms of phrase-structure parsing exact match, between each pair of five model variants (**No-KD**, **L2R-KD**, **R2L-KD**, **UF-KD**, and **UG-KD**).[26] We compute this pairwise model agreement score on the PTB validation set (§22). To better understand the *differences* between the models, we exclude sentences where all five models produce the exact same trees, leaving 826 out of 1700 sentences; further analysis indicates that the excluded sentences are shorter and less ambiguous.

We summarise the findings in Table 5.5, and discuss two key observations. First, the highest exact match agreement between any pair of different models is fairly low at 44.92%, further supporting our conjecture that different teacher models indeed impose different biases on the BERT student, as evidenced by the different model predictions. Second, all four structure-distilled BERT variants have the lowest pairwise agreement score with the **No-KD** baseline ($< 39\%$ pairwise model agreement), suggesting that all variants of our structure distillation objectives yield quantifiably different outputs compared to the no distillation alternative, which does not learn from the syntactic knowledge of RNNGs.

[26] For instance, comparing the agreement between the **No-KD** and **UG-KD** models, we treat the **UG-KD**'s output as "gold reference", and compute the exact match from the **No-KD**'s output with respect to that.

Task		Test Set - BERT$_{\text{LARGE}}$-Cased			
		No-KD	Best-KD	Error Red.	BERT SoTA
Parsing	Const. PTB - F1	95.80	**95.95**	3.73%	95.84†
	Const. PTB - EM	56.87	**57.74**	2.02%	-
	Const. OOD - F1	89.63	**90.20**	5.48%	89.91‡
	Dep. PTB - UAS	96.91	**97.03**	3.78%	97.0†
	Dep. PTB - LAS	95.33	**95.49**	3.43%	95.43†
	SRL - OntoNotes	87.59	**87.77**	1.45%	86.5$^{\diamond}$
	Coref. - OntoNotes	74.03	**74.69**	2.55%	79.6$^{\blacklozenge}$

Table 5.6: Test set results for the structured prediction tasks with BERT$_{\text{LARGE}}$-Cased; each entry reflects the mean of three random seeds. We compare the no distillation baseline (**"No-KD"**) with the best structure-distilled model, as selected on the validation set (**"Best-KD"**); **"Error Red."** reports the test error reductions *relative* to the **No-KD** baseline. We also report the previous state of the art (c. 2020) among non-ensemble models pre-trained on the original BERT training set ("**BERT SoTA**"). Concretely, † refers to the work of Zhou and Zhao (2019), ‡ refers to the work of Fried et al. (2019), $^{\diamond}$ refers to the work of Shi and Lin (2019), and $^{\blacklozenge}$ refers to the work of Joshi et al. (2020).

BERT$_{\text{LARGE}}$ Results. Having evaluated our structure-distilled BERT$_{\text{BASE}}$-Cased, we now apply our approach on top of BERT$_{\text{LARGE}}$-Cased that has 3x as many parameters, and present the results on the structured prediction tasks in Table 5.6. Overall, we observe a similar pattern of results as we do with BERT$_{\text{BASE}}$: On the structured prediction tasks, our best structure distillation approach reduces error by 1.5%-5.5% relative to the **No-KD** baseline. Furthermore, our structure-distilled BERT$_{\text{LARGE}}$ models establish new state of the art single model results c. mid-2020—among models pre-trained on the original BERT training set[27]—on phrase-structure parsing (PTB and OOD), PTB dependency parsing, and SRL.

5.6 Related Work

Earlier work has proposed a few different ways for introducing notions of syntactic structures into BERT, for instance through designing structurally motivated auxiliary losses (Wang et al., 2020), or including syntactic information in the embedding layers that serve as inputs to the Transformer (Sundararaman et al., 2019). In contrast, we employ a different technique for injecting syntactic biases, which is based on the structure distillation technique that we introduced in Ch. 4, although our approach and experiments in this chapter have two key differences. First, in Ch. 4, we put a sole emphasis on cases where both the teacher

[27]This comparison excludes other models like XLNet and RoBERTa, which are trained on more data.

and student models are unidirectional, left-to-right LMs; here we extend this objective for when the student model is a BERT masked language model that has access to bidirectional context. Second, in Ch. 4, we only evaluated the structure-distilled DSA-LSTMs in terms of perplexity and grammatical judgment, in the form of targeted syntactic evaluations (Marvin and Linzen, 2018). In contrast, here we evaluate our structure-distilled BERT models on 6 diverse structured prediction tasks, in addition to the diverse classification tasks of GLUE. It remains an open question whether, and to what extent, syntactic biases are helpful for a broader range of NLU tasks beyond grammatical judgment; our approach and findings in this chapter represent a step towards answering this important question.

More recently, substantial progress has been made in improving the performance of BERT and other masked LMs (Lan et al., 2020; Liu et al., 2019b; Raffel et al., 2019; Sun et al., 2020, *inter alia*). Our structure distillation technique is orthogonal, and can be applied on top of these approaches. We expect our structure distillation approach to provide similar gains for these more powerful masked language models, although we leave it to future work to firmly establish whether this is the case. Our approach also relates to the concept of *foundation* models (Bommasani et al., 2021): Concretely, our approach only modifies the *pre-training* stage of the foundation model (here a BERT masked language model), without requiring any changes at all at the fine-tuning stage. Furthermore, our approach maintains complete compatibility with the standard BERT pipelines: Our structure-distilled BERT models can simply be loaded as pre-trained BERT weights, which can then be fine-tuned for *any* BERT-compatible task in the exact same fashion. Lastly, our findings on the benefits of syntactic knowledge for structured prediction tasks that are not explicitly syntactic in nature, such as SRL and coreference resolution, are consistent with those of prior work (He et al., 2017; Swayamdipta et al., 2018; He et al., 2018; Strubell et al., 2018, *inter alia*).

5.7 Limitations

We remark on three existing limitations of our approach:

- First, we assume the existence of accurate phrase-structure parsers that enable us to obtain "silver-grade" trees for training the RNNG teachers. While this assumption holds true for English due to the existence of accurate and publicly available phrase-structure parsers, this assumption does not necessarily hold true for low-resource languages. Furthermore, the phrase-structure trees that we use to train the RNNG teachers may not generalise well to other domains, such as social media text, colloquial text, and vernacular English.

- Second, pre-training the BERT student in our naïve implementation is about half as fast on TPUs compared to the baseline BERT model. We attribute this difference to I/O bottleneck in loading the pre-computed RNNG teachers' predictions. This overhead only applies at pre-training, and can be reduced through better engineering and parallelisation.
- Third, the improvements afforded by our proposed structure distillation approach do not benefit all tasks equally; in addition, for some structured prediction tasks like coreference resolution, we observe relatively modest improvements from our approach. We remark, however, that our goal in this work is not to claim that our way of injecting syntactic inductive biases is the best or the most effective way of doing so; indeed, we conjecture that there are likely other, better ways of injecting syntactic biases that can potentially afford even larger improvements. Instead, our goal here is to demonstrate that there are, in fact, still some benefits of injecting syntactic biases for improving model performance, *even for* powerful Transformer-based BERT masked language models that acquire their knowledge from a large corpus of more than 3 billion words. In the end, we consider our findings not as a final answer, but rather as a proof of concept that—above and beyond creating the next large language model that is trained on ever-larger datasets—the quest for endowing models with stronger syntactic biases is in fact *still* a worthy endeavour and a promising research direction, even at the time of exciting BERT and GPT-3 success.

5.8 Conclusion

Given the remarkable success of Transformer-based masked language models like BERT that are trained on large amounts of data, it remains an open question whether or not more explicit syntactic biases can still be advantageous, even for these models that work well at scale. Through our RNNG knowledge distillation approach, we present evidence to the affirmative: Our structure-distilled BERT models outperform the baseline on a diverse set of 6 structured prediction tasks, including for tasks like coreference resolution and semantic role labelling that are not explicitly syntactic in nature. We achieve this through a new pre-training strategy that enables the BERT student to learn from the predictions of an explicitly hierarchical, albeit much less scalable, RNNG teacher model. Since the BERT student is a bidirectional model that estimates the conditional probabilities of masked words in context, we propose to distill an efficient yet surprisingly effective approximation of the RNNG's posterior estimate for generating each masked token conditional on the bidirectional context.

All in all, our findings suggest that syntactic biases are beneficial for a diverse range of structured prediction tasks, including for tasks that are not explicitly syntactic in nature. In addition, these biases are particularly helpful for improving fine-tuning sample efficiency on these tasks. Most importantly, our findings motivate the broader question of how we can design models that integrate stronger syntactic biases—and yet can be easily scalable at the same time—as a promising (if relatively underexplored) direction of future NLP research.

Chapter 6

Conclusion

Throughout this thesis, we have demonstrated the benefits of more explicit syntactic inductive biases for expressive neural language models that are trained at the small ($\sim$3 million words, Ch. 3), medium ($\sim$90 million words, Ch. 4), and large data scales ($\sim$3 billion words, Ch. 5). During the course of our investigation, we explored two approaches for injecting such syntactic biases into the models. In the first approach (Ch. 3), we use existing techniques that design hierarchical, tree-structured neural network models, where the model architecture—and by extension, the specific ordering of the neural network operations—is determined by hierarchical syntax trees (Socher et al., 2011, 2013; Tai et al., 2015; Dyer et al., 2015, 2016, *inter alia*), as opposed to the sequential positions of the input words. Whilst this approach is indeed effective, in practice tree-structured neural models are notoriously hard to scale; we attribute this fact to the difficulty of manually batching operations within this broad family of models, which renders them rather ill-suited for taking advantage of specialised deep learning hardwares that aim to accelerate training, such as GPUs and TPUs. Hence, in the second approach, we devise a novel knowledge distillation strategy that combines the best of both worlds, and design neural language models that not only incorporate stronger notions of syntactic biases, but also can be trained on large amounts of data at the same time. Our proposed knowledge distillation approach improves model performance under two different neural architectures (LSTMs and Transformers)—each with its own set of model inductive biases—and also under two different language modelling objectives (unidirectional, left-to-right language modelling, and masked language modelling). Altogether, these findings provide clear evidence for the effectiveness, generality, and robustness of our approach.

Furthermore, we find that more explicit syntactic inductive biases are not only beneficial for improving model performance on various challenging benchmarks of syntactic competence, but also for improving model performance on a wide variety of structured prediction tasks. Remarkably, this pattern of results extends even to structured prediction tasks like coreference resolution and semantic role labelling, which are not explicitly syntactic in nature. We further observe that the benefits of syntactic biases extend beyond improved overall performance: Language models that are augmented with more explicit syntactic biases are, in fact, more robust to adversarial examples in the form of sequential perturbations, and achieve better fine-tuning sample efficiency on the structured prediction tasks. Despite consistent improvements on various syntactic competence benchmarks and structured prediction tasks, we nevertheless observe mixed results on GLUE (Wang et al., 2019): Whereas the BERT model that is augmented with more explicit syntactic biases is able to outperform the baseline on the corpus of linguistic acceptability (Warstadt et al., 2018), the reverse is true for the rest of GLUE. This finding suggests that more explicit syntactic biases do not necessarily benefit different natural language understanding (NLU) benchmarks to the same extent, and further alludes to a partial dissociation between model performance on GLUE—which consists mostly of classification tasks—and on other syntax-sensitive benchmarks of NLU, including on NLU benchmarks that involve structured output spaces.

Altogether, our findings in this thesis: (i) provide evidence for the ability of neural language models to infer the underlying syntactic structure of natural language directly from data, even when they are not provided with explicit inductive biases or supervision to do so; (ii) highlight the need to control for various surface cues—including cues that humans are unlikely to employ—that a computational model can exploit to succeed at a cognitive task, without necessarily acquiring the underlying competence that the task was designed to evaluate in the first place; (iii) demonstrate that—despite the strong performance of these neural models—more explicit notions of syntactic inductive biases are, in fact, *still* helpful and beneficial for improving the performance of powerful LSTM and Transformer language models at various tasks, even in the case of masked language models like BERT that learn from vast amounts of data; (iv) provide an example of how existing linguistic theories (particularly concerning the syntax of natural language), annotations, and resources can lead to the design of better computational models of natural language: In the case of this work, we use cognitively-inspired diagnostic tools, such as number agreement, to highlight the flaws in generalisation that neural language models make; we then endow the neural models with a hierarchical inductive bias that is inspired by syntactic theories on how smaller linguistic units like words compose to form larger ones, such as phrases, clauses, and eventually sentences; and lastly (v) motivate the question of how we can design neural models that integrate stronger notions of syntactic biases—and yet can be easily scalable at the same time—as a promising (if relatively underexplored) direction of future research.

Lastly, building on the key findings and methods from this work, we conclude this thesis with a brief overview of several potentially promising directions of future research.

Scaling. Despite our findings, recent rapid progress in language modelling and various downstream tasks is not derived from augmenting models with syntactic inductive biases, but rather from training ever-larger Transformer language models—which lack explicit modelling of hierarchical syntactic structures—on increasingly larger amounts of data (Kaplan et al., 2020; Brown et al., 2020). To some extent, the choice to work with Transformer architectures is indeed understandable: Given a finite budget in terms of computing resources, one can spend those computing resources to train the largest possible model on the largest possible data—a tried-and-tested way of improving model performance that thus far has shown few signs of abating—or instead spend that compute budget on training explicitly hierarchical, tree-structured models that, despite their stronger syntactic inductive biases, are nevertheless infeasible to scale to larger models and datasets. The primary contribution of this work is to show that recent progress that is derived from *both* model and data scales can, in fact, be *complementary* to the benefits of stronger syntactic inductive biases. Through our controlled experiments, we demonstrate that—given the same model size and the same amount of data—models that are augmented with stronger syntactic biases can, in fact, outperform those that do not (albeit not necessarily on all types of tasks). We conjecture that this trend shall continue to hold for larger datasets and larger models compared to the ones that we explored in this work. Nevertheless, validating this hypothesis for the current largest models would require substantially more computing resources, scaling and engineering effort, and also research effort into discovering more effective ways for injecting syntactic inductive biases into neural language models, in a way that preserves model scalability.

Alternative scaling approach. Despite the effectiveness of our proposed knowledge distillation approach for combining the best of both data scale and syntactic biases, our approach still has two key drawbacks. First, it requires a more complex two-stage training process: Concretely, we need to first train the computationally expensive teacher model, store its predictions in a process that consumes a large amount of storage space, and only afterwards can we finally train the student model. Second, even though the tree-structured teacher model is an expert on syntax, it nevertheless lacks the opportunity to develop the relevant semantics and common-sense knowledge—both of which are important prerequisites for downstream tasks—from large textual corpora. Hence, it is conceivable that learning from the teacher model's predictions can interfere with the student model's ability to develop a more sophisticated representation of semantics and common-sense knowledge, although in this work, we mitigate this issue by interpolating the knowledge distillation loss with the standard language modelling loss. Hence, we conjecture that other approaches that *more directly* inject the syntactic biases into scalable neural architectures—thus eliminating the need for a separate, hard-to-scale teacher model—are the more promising ways forward.

Extension to other domains. In reality, natural language is not the only domain that exhibits a latent hierarchical structure. Other domains, such as code, database queries, symbolic mathematics, and even action spaces—*e.g.,* we want a robot that knows how to "*turn left*" and "*turn right*" to also be able to "*turn left then turn right*", whether or not these actions are specified in natural language—also exhibit underlying structures that are hierarchical in nature. Going forward, we conjecture that more explicit modelling of hierarchical structures—when conducted in a way that scales well to large datasets—can also be advantageous for these domains. In fact, the hierarchical structure within some of these domains can be extracted in a deterministic fashion (*e.g.,* abstract syntax trees for code), hence eliminating the need for either a statistical parser that predicts the hierarchical structure, or unsupervised approaches that treat the structure as a latent variable. Indeed, it is conceivable that the same hierarchical neural architecture can work well for all these different domains, although we leave it to future work to determine whether this is the case.

Extension to sequence-to-sequence models. Throughout this thesis, we have incorporated more explicit syntactic inductive biases within the context of neural language models; such models are often referred to as *decoder-only* models. Nevertheless, other interesting use cases that require an understanding of the underlying hierarchical structures—beyond the ones that we have explored in this thesis—are more appropriately characterised as *sequence-to-sequence* mapping problems. Under this approach, the source sequence is encoded by an encoder model, which the decoder model can then condition on, along with its past predictions. Hence, how we can design scalable neural models that can map from one form of hierarchical structure to another (*i.e.,* tree-to-tree) is an exciting open research question in the field, with important applications such as machine translation (where the source and target languages have different syntactic structures), writing programs based on natural language specifications, and semantic parsing that converts natural language to a formal meaning representation (*e.g.,* converting natural language to a database query).

Unsupervised learning & extension to other languages. Throughout this thesis, we have used supervised syntactic parsers to obtain a predicted, "silver-grade" phrase-structure tree for each training sentence, which we then use to train the explicitly hierarchical RNNG model. In practice, this reliance on supervised syntactic parsers is troublesome for two reasons. First, the optimal hierarchical tree structure may in fact depend on the particular task or domain of interest; for instance, the syntactic annotation conventions that were developed for English news data from the 1990s (with Penn Treebank as a prime example) may not necessarily transfer well to other domains, such as English social media text or vernacular English. Second, not all languages benefit from the existence of accurate and publicly available syntactic parsers. In particular, an important prerequisite for developing accurate phrase-structure parsers is the existence of a large treebank, which is expensive and time-consuming to develop. Given the need to develop NLP models that are *accessible* to everyone in the world—and not just those of us who happen to speak certain languages that benefit from large amounts of linguistic resources—an important direction for future work is to extend our approach to other languages. To that end, unsupervised learning of hierarchical syntactic structures—for instance, by treating the syntax tree as a latent variable, and optimising for the marginal log likelihood of the observed data—remains an exciting (albeit challenging) open research question in the field. Unfortunately, the constituency grammar formalism that we explored in this work is not necessarily well-suited to model languages with free word orders. We conjecture that further progress in the multi-lingual direction shall be done within the context of other grammar formalisms, such as dependency grammar, which is well-suited to model languages with free word orders, and additionally benefits from existing multi-lingual treebanks in a large number of languages.

Annex A

Appendix for Syntactic Structure Distillation Training for Left-to-Right Language Models

In Table A.1, we present the empirical results that include standard deviation values for the targeted syntactic evaluations experiment (Marvin and Linzen, 2018, Ch. 4), where we propose the DSA-LSTM, and compare its performance against multiple baselines (§4.5.1).

	Small Training Set		**Full Training Set**				
	S-DSA-LSTM†	**RNNG**†	**Full LSTM**	**BA-LSTM**	**DSA-LSTM**	**BERT**	**Humans**
Gulordava et al. (2018) **test ppl.**	93.95 (±0.18)	**92.30** (±0.27)	**53.73** (±0.16)	54.64 (±0.25)	56.74 (±0.26)	N/A	N/A
NUMBER AGREEMENT							
Simple	0.96 (±0.03)	**0.99** (±0.01)	**1.00** (±0.00)	**1.00** (±0.00)	**1.00** (±0.00)	1.00	0.96
In a sentential complement	**0.98** (±0.02)	0.93 (±0.02)	0.97 (±0.02)	**0.98** (±0.02)	**0.98** (±0.02)	0.83	0.93
Short VP coordination	0.88 (±0.04)	**0.96** (±0.02)	0.96 (±0.02)	0.95 (±0.02)	**0.99** (±0.02)	0.89	0.94
Long VP coordination	0.74 (±0.03)	**0.94** (±0.03)	**0.82** (±0.05)	0.80 (±0.04)	0.80 (±0.02)	0.98	0.82
Across a prepositional phrase	0.88 (±0.02)	**0.95** (±0.01)	0.89 (±0.02)	0.89 (±0.03)	**0.91** (±0.03)	0.85	0.85
Across a subject relative clause	0.87 (±0.02)	**0.95** (±0.03)	0.87 (±0.02)	0.87 (±0.01)	**0.90** (±0.02)	0.84	0.88
Across an object relative clause	0.69 (±0.06)	**0.95** (±0.03)	0.77 (±0.11)	0.81 (±0.05)	**0.84** (±0.03)	0.89	0.85
Across an object relative clause (no *that*)	0.61 (±0.05)	**0.93** (±0.02)	0.70 (±0.05)	0.74 (±0.03)	**0.77** (±0.02)	0.86	0.82
In an object relative clause	0.87 (±0.05)	**0.96** (±0.01)	0.90 (±0.03)	0.91 (±0.03)	**0.92** (±0.04)	0.95	0.78
In an object relative clause (no *that*)	0.88 (±0.03)	**0.96** (±0.02)	0.86 (±0.05)	0.83 (±0.02)	**0.92** (±0.02)	0.79	0.79
Average of number agreement	0.84 (±0.02)	**0.95** (±0.01)	0.87 (±0.02)	0.88 (±0.01)	**0.90** (±0.01)	0.89	0.86
REFLEXIVE ANAPHORA							
Simple	**0.90** (±0.01)	0.83 (±0.02)	0.91 (±0.01)	**0.92** (±0.03)	0.91 (±0.04)	0.94	0.96
In a sentential complement	**0.78** (±0.01)	0.46 (±0.05)	0.81 (±0.02)	0.81 (±0.02)	**0.82** (±0.03)	0.89	0.91
Across a relative clause	0.67 (±0.03)	**0.82** (±0.02)	0.64 (±0.02)	0.64 (±0.02)	**0.67** (±0.03)	0.80	0.87
Average of reflexive anaphora	**0.78** (±0.01)	0.70 (±0.02)	0.79 (±0.01)	0.79 (±0.02)	**0.80** (±0.03)	0.88	0.91
NEGATIVE POLARITY ITEMS							
Simple	**0.84** (±0.05)	0.28 (±0.05)	0.96 (±0.04)	**0.98** (±0.02)	0.94 (±0.04)	N/A	0.98
Across a relative clause	0.73 (±0.07)	**0.78** (±0.06)	0.75 (±0.12)	0.70 (±0.10)	**0.91** (±0.07)	N/A	0.81
Average of negative polarity items	**0.79** (±0.05)	0.53 (±0.04)	0.86 (±0.06)	0.84 (±0.05)	**0.92** (±0.05)	N/A	0.90
Average of all constructions	0.82 (±0.02)	**0.85** (±0.02)	0.85 (±0.02)	0.86 (±0.01)	**0.89** (±0.01)	N/A	0.88

Table A.1: Empirical findings for the **“DSA-LSTM”**. For each column, we report the mean (± standard deviation) values of 10 identical models trained from different random seeds. **“S-DSA-LSTM”** indicates a DSA-LSTM model trained on the smaller RNNG training set (§4.4.4), while **“BA-LSTM”** is the born-again model where the teacher model is a full LSTM LM. We also compare the models with a **“BERT”**-Base model on the same benchmark, which is not strictly comparable as BERT is trained on 30 times more data and benefits from bidirectionality; all other reported results (including the proposed DSA-LSTM) are based on unidirectional LMs. † indicates models trained on the smaller training set (§4.4.4). Results in bold denote the best among those trained with the same amounts of data.

Annex B

Appendix for Syntactic Structure Distillation Pre-training for Bidirectional Encoders

B.1 Preliminary Experiments

We discuss the preliminary experiments that aim to assess the quality (in terms of posterior negative log likelihood) and computational efficiency (in terms of speed-up over exact inference) of our proposed posterior approximation procedure (§5.4.3). Recall that this approximation procedure only applies at *inference*, whereas the underlying language model itself is still *trained* in a typical unidirectional, left-to-right fashion.

Model. Since exactly computing the RNNG's marginal next-word distributions $t_{\psi}(x_i \mid \mathbf{x}_{<i})$ requires an intractable marginalisation procedure over all possible tree prefixes $\mathbf{y}_{<i}$, we run our experiments within the context of sequential LSTM language models, where $t_{\text{LSTM}}(x_i \mid \mathbf{x}_{<i})$ can be computed exactly. This setup crucially enables us to isolate the impact of approximating exact inference of the posterior distribution over x_i under the bidirectional context (Eq. 5.2) with our proposed approximation (Eq. 5.5), without introducing further confounds stemming from the RNNG's marginal approximation procedure (Eq. 5.6).

Dataset and preprocessing. We train the LSTM LM on an *open-vocabulary* version of the PTB,[1] in order to simulate the main experimental setup where both the RNNG teacher and BERT student are also open-vocabulary by virtue of byte-pair encoding (BPE) preprocessing.

[1]Our open-vocabulary setup means that our results are not directly comparable to prior work on PTB language modelling (Mikolov et al., 2010, *inter alia*), which mostly employed a special "UNK" symbol for representing infrequent or unknown words.

To that end, we preprocess the dataset with a SentencePiece (Kudo and Richardson, 2018) BPE tokenisation, where $|\Sigma| = 8,000$; we preserve all case information. Following the empirical setup of the phrase-structure parsing (§5.5.1) experiments, we use Sections 02-21 for training, Section 22 for validation, and Section 23 for testing.

Model hyper-parameters. We train the language model with 2 LSTM layers, 250 hidden units per layer, and a variational dropout (Gal and Ghahramani, 2016) rate of 0.2. We optimse the model parameters with mini-batch gradient descent, with an initial learning rate of 0.25 that is decayed exponentially by a factor of 0.92 for every epoch after the tenth. Since our approximation relies on a separately trained right-to-left LM (Eq. 5.5), we train this variant with the exact same hyper-parameters and dataset split as the left-to-right model.

Evaluation and baselines. We evaluate the models in terms of the average posterior negative log likelihood (NLL) and perplexity.[2] Since exact inference of the posterior is expensive, we evaluate the model only on the first 400 sentences of the test set. We compare the following model variants:

- a mixture of experts baseline that simply mixes ($\alpha = 0.5$) the probabilities from the left-to-right and right-to-left LMs in an *additive* fashion, as opposed to *multiplicative* as in the case of our PoE-like approximation in Eq. 5.5 (**"MoE"**);
- our approximation of the posterior over x_i (Eq. 5.5), where $q(w)$ is the *uniform* distribution (**"Uniform Approx."**);
- our approximation of the posterior over x_i (Eq. 5.5), but where $q(w)$ is the *unigram* distribution (**"Unigram Approx."**); and
- exact inference of the posterior as computed from the left-to-right model, as defined in Eq. 5.2 (**"Exact Inference"**).

Discussion. We summarise the findings in Table B.1, based on which we remark on two key observations. First, the posterior NLL of our approximation procedure that makes use of the unigram distribution (**Unigram Approx.**; third row) is not much worse than that of exact inference, in exchange for a more than 50,000 times speedup[3] in inference speed.

[2] In practice, this perplexity is derived from simply exponentiating the average posterior negative log likelihood.

[3] All three approximations in Table B.1 have similar runtimes.

Model	Posterior NLL	Posterior Ppl.
MoE	3.28	26.58
Uniform Approx.	3.18	24.17
Unigram Approx.	**2.68**	**14.68**
Exact Inference	2.50	12.25

Table B.1: Findings from preliminary experiments (posterior negative log likelihood, **lower is better**) that assess the quality of our proposed posterior approximation procedure. We compare the three approximations against exact inference (bottom row; Eq. 5.2), which is computed from the left-to-right model.

Nevertheless, using the uniform distribution (second row) on $q(w)$ in place of the unigram one (Eq. 5.5) results in a much worse posterior NLL, even though this approach features a similar multiplicative interaction between the left-to-right and right-to-left LMs. Second, combining the left-to-right and right-to-left LMs using a mixture of experts—a baseline that is not well-motivated by our theoretical analysis—yields the worst result.

B.2 BERT Hyper-Parameters

Here we outline the hyper-parameters of the BERT student in terms of pre-training data creation, masked LM pre-training, and fine-tuning.

Pre-training data creation. We use the same codebase[4] and pre-training data as Devlin et al. (2019), which are derived from a mixture of Wikipedia and Books text corpora. To train our structure-distilled BERTs, we sample a masking from these corpora following the same hyper-parameters used to train the original $\text{BERT}_{\text{BASE}}$-Cased model: A maximum sequence length of 512, a per-word masking probability of 0.15 (up to a maximum of 76 masked tokens in a 512-length sequence), and a dupe factor of 10. We apply a random seed of 12345. We tokenise the raw dataset using NLTK, and then apply the same (BPE-augmented) vocabulary and WordPiece tokenisation as in the original BERT model. All other hyper-parameters are set to the same values as in the publicly released, original BERT.

Masked LM pre-training. We train all model variants (including the no distillation/standard BERT baseline for fair comparison) using a batch size of 256 sequences. We use an initial Adam learning rate of $3e^{-4}$ for the $\text{BERT}_{\text{BASE}}$ models (as opposed to $1e^{-4}$ in the original

[4] `https://github.com/google-research/bert`.

	Model	CoLA	SST-2	MRPC	QQP	MNLI (M/MM)	QNLI	RTE	GLUE AVG
DEV	No-KD	60.2	**92.2**	**90.0**	**89.4**	**90.3/90.9**	**90.7**	**71.1**	**84.4**
	UG-KD	**60.6**	92.0	88.9	89.3	89.6/90.0	89.9	68.6	83.6
TEST	No-KD	53.1	**92.5**	**88.0**	**88.8**	**82.8/81.8**	**89.9**	**65.4**	**80.3**
	UG-KD	**55.3**	91.2	87.6	88.7	81.9/80.8	89.5	65.0	80.0

Table B.2: Summary of the full results on GLUE (**higher is better**), comparing the **No-KD** baseline with the **UG-KD** structure-distilled BERT (§5.5.2). All results are based on a single random seed: we select the 1-best fine-tuning hyper-parameter (including random seed) on the validation set, which we then evaluate on the test set. Bold denotes the best results for each task within a particular validation/test split.

BERT model) and $1e^{-4}$ for the $\text{BERT}_{\text{LARGE}}$ models. Following Devlin et al. (2019), we pre-train our models for 1M steps. All other hyper-parameters are set to their default values.

GLUE fine-tuning. For each GLUE task, we fine-tune the BERT model by running a grid search over a set of potential learning rates of $\{5e^{-6}, 1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5}\}$ and batch sizes of $\{16, 32\}$, with 5 random seeds. This setup leads to 50 fine-tuning configurations for each GLUE task. Following the BERT fine-tuning setup of Joshi et al. (2020), we train each fine-tuning configuration for 10 epochs, except for CoLA, where we train for 4 epochs.

Structured prediction fine-tuning. For the structured prediction tasks, we use the following hyper-parameters for learning rate and batch size. These hyper-parameters are either the default for a given codebase, or lightly tuned on the No-KD models. We use the same hyper-parameters across all models of a given size (BASE or LARGE) on a given task.

- In-order phrase-structure parser: a BERT learning rate of $2e^{-5}$, a batch size of 32, and a warmup period of 160 updates.
- HPSG dependency parser: a BERT learning rate of $5e^{-5}$, a batch size of 150, and a warmup period of 160 updates.
- Coreference resolution: for the $\text{BERT}_{\text{BASE}}$ models: a learning rate of $1e^{-5}$ and a maximum segment length of 128 word pieces. For the $\text{BERT}_{\text{LARGE}}$ models: a learning rate of $5e^{-6}$ and a maximum segment length of 512 word pieces. Both sizes use a learning rate warmup period of 2 epochs and a batch size of 1 document.
- Semantic role labelling: for the $\text{BERT}_{\text{BASE}}$ models: a learning rate of $5e^{-5}$. For the $\text{BERT}_{\text{LARGE}}$ models: a learning rate of $1e^{-5}$. Both sizes use a batch size of 32.

B.3 Full GLUE Results

We provide the full GLUE results for the **No-KD** and the **UG-KD** BERTs in Table B.2.

Bibliography

Steven Abney. Statistical methods and linguistics. In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press, 1996.

Steven Abney and Mark Johnson. Memory requirements and local ambiguities for parsing strategies. *Journal of Psycholinguistic Research*, 1991.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proc. of ICLR*, 2017.

Roee Aharoni and Yoav Goldberg. Towards string-to-tree neural machine translation. In *Proc. of ACL*, 2017. doi: 10.18653/v1/P17-2021.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proc. of ACL*, 2016.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proc. of NAACL*, 2016a. doi: 10.18653/v1/N16-1181.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proc. of CVPR*, 2016b. doi: 10.1109/CVPR.2016.12.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proc. of ICML*, 2016.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proc. of NeurIPS*, 2014.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Emmon Bach. An extension of classical transformational grammar. In *Problems of Linguistic Metatheory (Proceedings of the 1976 Conference)*. Michigan State University, 1976.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*, 2015.

C. Lee Baker. Double negatives. *Linguistic inquiry*, 1970.

James K. Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 1979.

Srinivas Bangalore and Aravind K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 1999.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? In *Proc. of ACL*, 2017. doi: 10.18653/v1/P17-1080.

Yoshua Bengio, Paolo Frasconi, and Patrice Y. Simard. The problem of learning long-term dependencies in recurrent networks. In *Proc. of ICNN*, 1993. doi: 10.1109/ICNN.1993.298725.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research.*, 2003.

Jean Berko. The child's learning of english morphology. *WORD*, 14(2-3), 1958. doi: 10.1080/00437956.1958.11659661.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. Deep RNNs encode soft hierarchical syntax. In *Proc. of ACL*, pages 14–19, 2018. doi: 10.18653/v1/P18-2003.

Phil Blunsom and Miles Osborne. Probabilistic inference for machine translation. In *Proc. of EMNLP*, 2008.

Kathryn Bock and Carol A Miller. Broken agreement. *Cognitive Psychology*, 23(1):45–93, 1991.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proc. of NeurIPS*, 2020.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proc. of KDD*, 2006.

Jan Buys and Phil Blunsom. A Bayesian model for generative transition-based dependency parsing. *CoRR*, abs/1506.04334, 2015.

Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI*, 1997.

Eugene Charniak. Top-down nearly-context-sensitive parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 674–683, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D10-1066`.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005.

Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech and Language*, 14(4), 2000.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proc. of ACL*, 2018a. doi: 10.18653/v1/P18-1008.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proc. of ACL*, 2018b.

David Chiang, Yuval Marton, and Philip Resnik. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*, 2008.

David Chiang, Kevin Knight, and Wei Wang. 11,001 new features for statistical machine translation. In *Proc. of NAACL-HLT*, 2009.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, 2014. doi: 10.3115/v1/D14-1179.

Do Kook Choe and Eugene Charniak. Parsing as language modeling. In *Proc. of EMNLP*, 2016.

Yejin Choi and Claire Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proc. of EMNLP*, 2008.

Noam Chomsky. *Syntactic Structures*. Mouton, The Hague/Paris, 1957.

Noam Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, 1965.

Noam Chomsky. *Problems of knowledge and freedom*. Pantheon, New York, 1971.

Noam Chomsky. Rules and representations. *Behavioral and Brain Sciences*, 3(1):1–15, 1980. doi: 10.1017/S0140525X00001515.

Noam Chomsky. *Lectures on Government and Binding*. Foris Publications, 1981.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *Proc. of ICLR*, 2017.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. of CVPR*, 2012.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proc. of ICLR*, 2020.

Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4), 2007.

Michael Collins. *Head-driven statistical models for natural language processing*. PhD thesis, University of Pennsylvania, 1999.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 2005. doi: 10.1162/0891201053630273.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proc. of ACL*, 2018. doi: 10.18653/v1/P18-1198.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proc. of ACL*, 2019.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proc. of ICML*, 2017.

Marie-Catherine De Marneffe and Christopher D. Manning. Stanford typed dependencies manual, 2008.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019. doi: 10.18653/v1/N19-1423.

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *Handbook of Nonlinear Filtering*. Oxford, 2011.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*, 2015.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

Ahmad Emami and Frederick Jelinek. A neural syntactic language model. *Machine Learning*, 60:195–227, 2005.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proc. of CoNLL*, 2017. doi: 10.18653/v1/K17-1003.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-sequence attentional neural machine translation. In *Proc. of ACL*, 2016. doi: 10.18653/v1/P16-1078.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to parse and translate improves neural machine translation. In *Proc. of ACL*, 2017. doi: 10.18653/v1/P17-2012.

Fernanda Ferreira and Charles Clifton. The independence of syntactic processing. *Journal of Memory and Language*, 1986. doi: https://doi.org/10.1016/0749-596X(86)90006-9.

Meaghan Fowlie. *Slaying the Great Green Dragon: Learning and modelling iterable ordered optional adjuncts*. PhD thesis, UCLA, 2017.

Winthrop Nelson Francis and Henry Kučera. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Linguistics, 1979.

Julie Franck, Glenda Lassi, Ulrich H. Frauenfelder, and Luigi Rizzi. Agreement and movement: A syntactic analysis of attraction. *Cognition*, 101(1), 2006. doi: https://doi.org/10.1016/j.cognition.2005.10.003.

Daniel Fried, Mitchell Stern, and Dan Klein. Improving neural parsing by disentangling model combination and reranking effects. In *Proc. of ACL*, 2017. doi: 10.18653/v1/P17-2025.

Daniel Fried, Nikita Kitaev, and Dan Klein. Cross-domain generalization of neural constituency parsers. In *Proc. of ACL*, 2019.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *Proc. of ICML*, 2018.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proc. of NAACL*, 2019.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. of NeurIPS*, 2016.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018.

Anastasia Giannakidou et al. *Polarity sensitivity as (non) veridical dependency*. John Benjamins Publishing Company Amsterdam, 1998.

Daniel Gildea. Corpus variation and parser performance. In *Proc. of EMNLP*, 2001.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 2002. doi: 10.1162/089120102760275983.

Yoav Goldberg. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of ICLR*, 2015.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. In *Proc. of NAACL*, 2018.

Kadri Hacioglu. Semantic role labeling using dependency trees. In *Proc. of COLING*, 2004.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. Finding syntax in human encephalography with beam search. In *Proc. of ACL*, 2018. doi: 10.18653/v1/P18-1254.

Betty Hart and Todd R. Risley. *Meaningful differences in the everyday experience of young American children*. Paul H. Brookes Publishing Company, Baltimore, MD, USA, 1995.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016. doi: 10.1109/CVPR.2016.90.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*, 2017.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*, 2018.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*, 1992.

James Henderson. Inducing history representations for broad coverage statistical parsing. In *Proc. of NAACL*, 2003.

James Henderson. Discriminative training of a neural network statistical parser. In *Proc. of ACL*, 2004.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proc. of AAAI*, 2018.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proc. of EMNLP*, 2019.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proc. of NAACL*, 2019. doi: 10.18653/v1/N19-1419.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL `http://arxiv.org/abs/1503.02531`.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998. doi: 10.1142/S0218488598000094.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger P. Levy. A systematic assessment of syntactic generalization in neural language models. In *Proc. of ACL*, 2020.

Marcus Hutter. The human knowledge compression contest. 2012.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proc. of ACL*, 2019.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262100665.

Frederick Jelinek and John D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 1991.

Lifeng Jia, Clement Yu, and Weiyi Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proc. of CIKM*, 2009.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proc. of EMNLP*, 2017.

Philip N. Johnson-Laird. *Mental Models*. Harvard University Press, 1983.

Karen Sparck Jones. Natural language processing: a historical review. In *Current Issues in Computational Linguistics: in Honour of Don Walker (Ed Zampolli, Calzolari and*. Kluwer, 1994.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 1999.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. BERT for coreference resolution: Baselines and analysis. In *Proc. of EMNLP*, 2019.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *TACL*, 2020.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling, 2016.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary N. Tajchman, and Nelson Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Proc. of ICASSP*, 1995. doi: 10.1109/ICASSP.1995.479396.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to discover, ground and use words with segmental neural language models. In *Proc. of ACL*, 2019. doi: 10.18653/v1/P19-1645.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proc. of EMNLP*, 2016. doi: 10.18653/v1/D16-1139.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. Unsupervised recurrent neural network grammars. In *Proc. of NAACL*, 2019. doi: 10.18653/v1/N19-1114.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010. ISBN 0521874157.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of NeurIPS*, 2012.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proc. of EMNLP System Demonstrations*, 2018.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*, 2017.

Adhiguna Kuncoro, Chris Dyer, John Hale, and Phil Blunsom. The perils of natural behavioral tests for unnatural models: The case of number agreement. *Poster presented at the Learning Language in Humans and in Machines conference, online at: https://osf.io/view/L2HM/*, 2018a.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proc. of ACL*, 2018b. doi: 10.18653/v1/P18-1132.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. Scalable syntax-aware language modelling with knowledge distillation. In *Proc. of ACL*, 2019.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 2020.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proc. of ICML*, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proc. of ICLR*, 2020.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015.

Gaël Le Godais, Tal Linzen, and Emmanuel Dupoux. Comparing character-level neural language models using a lexical decision task. In *Proc. of EACL*, 2017.

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. In *Proc. of NAACL*, 2018.

Tao Lei. When attention meets fast recurrence: Training language models with reduced compute. In *Proc. of EMNLP*, 2021.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proc. of KR*, 2012.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), University of Helsinki*, pages 6–7, 1970.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 2016.

Jiangming Liu and Yue Zhang. In-order transition-based constituent parsing. *TACL*, 2017.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL*, 2019a.

Yijia Liu, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. Distilling knowledge for search-based structured prediction. In *Proc. of ACL*, 2018. doi: 10.18653/v1/P18-1129.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. In *Proc. of ICLR*, 2016.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48), 2020. doi: 10.1073/pnas.1907367117.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 1993.

William Marslen-Wilson. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–523, 1973.

Yuval Marton and Philip Resnik. Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. of ACL-HlT*, 2008.

Rebecca Marvin and Tal Linzen. Targeted syntactic evaluation of language models. In *Proc. of EMNLP*, 2018.

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proc. of NAACL*, 2006.

David McClosky, Eugene Charniak, and Mark Johnson. When is self-training effective for parsing? In *Proc. of COLING*, 2008.

R. Thomas McCoy, Robert Frank, and Tal Linzen. Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proc. of CogSci*, 2018.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proc. of ACL*, 2019.

Igor Mel'čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, 1988.

Gabor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *Proc. of ICLR*, 2018.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of ICLR*, 2013.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. Interspeech*, 2010.

George A. Miller and Stephen Isard. Free recall of self-embedded english sentences. *Information and Control*, 7(3):292–303, 1964. doi: https://doi.org/10.1016/S0019-9958(64)90310-9.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*, 2009.

Piotr Mirowski and Andreas Vlachos. Dependency recurrent neural language models for sentence completion. In *Proc. of ACL-IJCNLP*, 2015. doi: 10.3115/v1/P15-2084.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proc. of ICML*, 2007. doi: 10.1145/1273496.1273577.

Karo Moilanen and Stephen Pulman. Sentiment composition. In *Proc. of RANLP*, 2007.

Maria Nădejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. Predicting target language CCG supertags improves neural machine translation. In *Proc. of the Second Conference on Machine Translation*, 2017. doi: 10.18653/v1/W17-4707.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proc. of EMNLP*, 2018.

Matthew J. Nelson, Imen El Karoui, Kristof Giber, Xiaofang Yang, Laurent Cohen, Hilda Koopman, Sydney S. Cash, Lionel Naccache, John T. Hale, Christophe Pallier, and Stanislas Dehaene. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences of the United States of America*, 2017.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017a.

Graham Neubig, Yoav Goldberg, and Chris Dyer. On-the-fly operation batching in dynamic computation graphs. In *Proc. of NeurIPS*, 2017b.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual treebank collection. In *Proc. of LREC*, 2016.

Hiroshi Noji and Yohei Oseki. Effective batching for recurrent neural network grammars. In *Findings of the ACL-IJCNLP*, 2021. doi: 10.18653/v1/2021.findings-acl.380.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 2005.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proc. of ICML*, 2013.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *Proc. of ICLR*, 2018.

Fernando Pereira. Formal grammar and information theory: Together again? *PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY*, 2000.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548, 2017.

Amy Perfors, Joshua B. Tenenbaum, and Terry Regier. The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338, 2011. doi: https://doi.org/10.1016/j.cognition.2010.11.001.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018. doi: 10.18653/v1/N18-1202.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proc. of NAACL*, 2007.

Slav Petrov and Ryan McDonald. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 2012.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*, 2006.

Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proc. of CoNLL*, 2012.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In *Proc. of CoNLL*, 2013.

Geoffrey K. Pullum. Learnability, hyperlearning, and the poverty of the stimulus. *Proceedings of the Annual Meeting of the Berkeley Linguistics Society*, 22, 1996.

Geoffrey K Pullum and Barbara C Scholz. Empirical assessment of stimulus poverty arguments:. *The Linguistic Review*, 19(1-2):9–50, 2002. doi: doi:10.1515/tlir.19.1-2.9.

Stephen Pulman. Grammars, parsers, and memory limitations. *Language and Cognitive Processes*, 1986.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proc. of ICML*, 2009. doi: 10.1145/1553374.1553486.

Philip Resnik. Left-corner parsing and psychological plausibility. In *Proc. of COLING*, 1992.

Marco Tulio Ribeiro, Tongshuang (Sherry) Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proc. of ACL*, 2020.

Laura Rimell, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *Proc. of EMNLP*, 2009.

Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2), 2001.

Ronald Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(8), 2000. doi: 10.1109/5.880083.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 1986. doi: 10.1038/323533a0.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. Broad-coverage parsing using human-like memory constraints. 36(1):1–30, 2010.

Abigail See, Peter Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL*, 2017.

Rico Sennrich. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proc. of EACL*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proc. of ICLR*, 2019.

Peng Shi and Jimmy Lin. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255, 2019.

Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural MT learn source syntax? In *Proc. of EMNLP*, 2016.

Jason Smith and Jason Eisner. Unsupervised learning on an approximate corpus. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 131–141, Montréal, Canada, June 2012. Association for Computational Linguistics. URL `https://aclanthology.org/N12-1014`.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proc. of ICML*, 2011.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, 2013.

Mark Steedman. *The Syntactic Process*. MIT Press, 2000.

Mitchell Stern, Daniel Fried, and Dan Klein. Effective inference for generative neural parsing. In *Proc. of EMNLP*, 2017. doi: 10.18653/v1/D17-1178.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*, 2018.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proc. of AAAI*, 2020.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. Syntax-infused transformer and bert models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156*, 2019.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Proc. of INTERSPEECH*, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. of ICLR*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. of CVPR*, 2016. doi: 10.1109/CVPR.2016.308.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL*, 2015.

Michael Tanenhaus, Michael Spivey-Knowlton, Kathleen Eberhard, and Julie Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634, 1995.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. Syntax annotation for the GENIA corpus. In *Proc. of IJCNLP*, 2005.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*, 2019a.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proc. of ICLR*, 2019b.

Lucien Tesnière. *Eléments de Syntaxe Structurale*. Klincksieck, Paris, 1959.

Ke Tran, Arianna Bisazza, and Christof Monz. The importance of being recurrent for modeling hierarchical structure. In *Proc. of EMNLP*, 2018. doi: 10.18653/v1/D18-1503.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *NIPS*, 2015.

Wilhelm von Humboldt. *Über die Verschiedenheit des menschlichen Sprachbaues und ihren Einfluß auf die geistige Entwicklung des Menschengeschlechts*. Reprint: Humboldt 1972:368-756, 1836.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 2008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of ICLR*, 2019.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. StructBERT: Incorporating language structures into pre-training for deep language understanding. In *Proc. of ICLR*, 2020.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *TACL*, 8, 2020. doi: 10.1162/tacl_a_00321.

Paul J. Werbos. Applications of advances in nonlinear sensitivity analysis. In *System Modeling and Optimization*, 1982. ISBN 978-3-540-39459-4.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. Structural supervision improves learning of non-local grammatical dependencies. In *Proc. of NAACL*, 2019.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 2018. doi: 10.1162/tacl_a_00019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Proc. of NeurIPS*, 2019.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*, 1995.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proc. of COLING*, 2000.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. Memory architectures in recurrent neural network language models. In *Proc. of ICLR*, 2018.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373, 2019.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. In *Proc. ICLR*, 2015.

Junru Zhou and Hai Zhao. Head-driven phrase structure grammar parsing on Penn treebank. In *Proc. of ACL*, 2019.

Zhi-Hua Zhou and Ming Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 2005.