Predicting and Discovering Linguistic Structure with Neural Networks

Ke Tran

Predicting and Discovering Linguistic Structure with Neural Networks

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op vrijdag 30 november 2018, te 12:00 uur

door

Ke Tran

geboren te Nghe An, Vietnam

Promotiecommissie

Promotor:		
	Dr. C. Monz	Universiteit van Amsterdam
Co-promotor:		
	Dr. A. Bisazza	Universiteit Leiden
Overige leden:		
	Dr. K. Cho	New York University
	Dr. E. Gavves	Universiteit van Amsterdam
	Prof. dr. G. van Noord	Rijksuniversiteit Groningen
	Prof. dr. K. Sima'an	Universiteit van Amsterdam
	Prof. dr. M. Welling	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Cover by Ke Tran
Printed by Ridderprint, The Netherlands.

ISBN: 978-94-6375-208-4

To
my parents, who love colourful figures,
and
Arianna Bisazza, who argued all the time.

Acknowledgments

I owe a debt of gratitude to Christof Monz, my advisor, who granted me the infinity stone of freedom in pursuing my research. Christof took me in as a graduate student despite knowing that Machine Translation was not a topic dear to my heart. You allowed me to work on neural networks in the very first months of my PhD. This was beyond my expectation. I am thankful for what you have done as a supervisor, for convincing me MT is an interesting problem, for teaching me to ask practical questions, for your patience, for disagreeing, for supporting, and for many enjoyable conversations during unscheduled short breaks at the balcony.

Arianna Bisazza has been my co-supervisor since the first year of the PhD. You taught me a great deal about languages, and most importantly, to care about solving language problems. Being a PhD candidate, my daily routine consisted of biking to Science Park, making a cup of coffee, bursting into Arianna's office, and arguing about the nature of reality that just occurs to me a few minutes ago. It was fun. Perhaps arguing with you was the highlight of my brief existence at Science Park. You have shaped my scientific thinking. When we were not arguing, which was often the case, you offered me moral support. Thank you for being a role model that I will always look up to.

I am honoured to have Cho, Efstratios, Gertjan, Khalil, and Max serving on my committee. Thank you for your time reading this thesis.

I am fortunate to have Kristina Toutanova as my mentor during the summer internship at Microsoft Research Redmond in 2015. Kristina is a rare combination of sharpness and humility that has a profound impact on who I want to become as a scientist. I also thank Jaime Teevan, Chris Brockett, Saleema Amershi, and other members of the MSR Microtasking team for a fabulous summer.

My interest in unsupervised learning was kindled during the second internship at ISI/USC. I would like to thank Kevin Knight and Daniel Marcu for a joyful summer working on neuralizing unsupervised models. Ashish Vaswani and

Yonatan Bisk always schemed to move our discussions form Marina del Rey to Venice Grind coffee shop where we had most of the technical details figured out. Thank you both for a super fun and productive summer. Yonatan later became my collaborator for a grammar induction paper. Thank you for sharing many frustrations working on syntax and being a wonderful collaborator.

The MT group is small but it makes up a big part during my journey. Thank you Katya, Marlies, Marzieh, Hamid, Praveen, and Ivan. Although Maartje is not in the MT group, you have been an amiable companion in the Tardis C3.241 during the time this thesis was written. It is hard to stay sane without sharing the hardship and laughter with all of you.

In retrospect, I have to admit that I underestimated the importance of beer time, coffee breaks, and social activities on the well-being of a graduate student. Having said that, ILPS is a great place for raising a PhD student. Thank you Maarten for managing such an amazing group. I would like to thank past and current members of ILPS for leaving many fond memories of a long and windy journey. A special thank you goes to Petra whose magic dispelled countless shadows of bureaucracy that lurked and roamed in the park of science.

I have learned a lot from students who I supervised or TA'ed. Thank you for making a complete experience of doing a PhD.

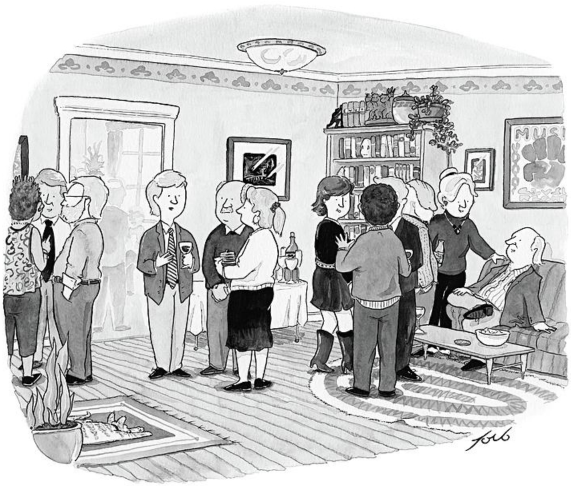
I am grateful for many friends I have met during my long journey. Abdo generously and hilariously offers chaos and randomness into the pandemonium of other people's life including mine. I am pleased to have you as a colleague, a roommate, a paranymph, and a friend. Evgeny, the Spock to my Kirk, logicifies every aspect of human emotions. I am thankful to have this warm-hearted fella standing next to me in my defense. Artem, a Beluga soul filled with an undying love for Karl Marx, tirelessly challenges the definition of social norms and never fails to expand my knowledge. After many months nodding each other's existence at the 2nd floor balcony, Vittorio and I found that we share a passion for ramen, espresso, and jazz as well as the disappointment of the Dutch weather. Everyday, Orhan and I quested for the best Asian restaurant during my three months internship in Paris. Jo kept me updated with German politics and shared our struggle to make work-life balance even though we both failed miserably. Milos with his unique sense of humor introduced me to the world of absurdism. If I have to say anything about the correlation coefficient between the absurdity and the meaningfulness of life, I'm confident that it's positive. Raquel is definitely on my side with that statement although Phong might disagree. Hoang Hung, my friend since undergraduate days, always reminds me that I actualy went to college. I should mention that I like to hangout with Margo. Thank you alI for being a part of my PhD and my life.

Finally, my PhD time would have been impossible without the unconditional love and support from my parents. My dad took me to the library when I was a kid and when I ran out of books to read, my mom bought me a computer.

Contents

Chapter 1

Introduction



"Behold, as I guide our conversation to my narrow area of expertise."

Current natural language processing (NLP) systems based on neural networks often rely on an enormous amount of data, if available, to learn the underlying patterns that are useful to solve the tasks of interest. Despite the recent success of neural network approaches in NLP, it has been shown that modern NLP systems are quite brittle in the presence of infrequent but probable linguistic phenomena. Belinkov and Bisk (2018) provide an example where Facebook's character-based neural machine translation (NMT) system mistook the Arabic word يصبحهم (*ySbHhm*), meaning a blessing for Good Morning, for يذبحهم (*y*bHhm*), meaning to hurt or slaughter someone.

In the first Workshop on Building Linguistically Generalizable Natural Language Processing Systems, findings of the *Build It Break It, The Language Edi-*

tion[1] shared task (Ettinger et al., 2017) corroborate the need for having linguistic knowledge in NLP systems. The results of the sentiment classification shared task show that all the deep neural network models break when they are confronted with linguistically motivated attacks. For instance, with a minimal substitution of the comparative phrase *"little too willing"* by *"just willing enough"*, all the participated systems misclassify the sentence *"A bizarre (and sometimes repulsive) exercise that's just willing enough to swoon in its own weird embrace."* as negative. This shows the limitations of all the participated systems when handling the sentiment conveyed by the phrase *"just willing enough"* in context.

Taking a step further without manually crafting adversarial examples, Jia and Liang (2017) evaluate reading comprehension systems using a simple method to automatically generate adversarial sentences and insert them into reading passages without changing the correct answers. Their method utilizes linguistic tools such as WordNet (Miller, 1995), Glove word vectors (Pennington et al., 2014), and Stanford CoreNLP (Manning et al., 2014) in addition to crowdsourcing to generate sentences that are similar to the question. They show that the accuracy of sixteen published (neural network) models for the Stanford Question Answering Dataset dramatically drops from 76% F1 score to 36% on average. This finding shows the need for developing new models that understand language more precisely.

We believe that linguistic knowledge is crucial for advancing natural language processing. Apart from leveraging known linguistic structure to develop accurate NLP systems, linguistic knowledge can be used to diagnose and probe the robustness of NLP systems as well as to understand their limitations and to shed light on possible research directions in the quest of building intelligent machines (Lake et al., 2017).

1.1 Research Questions

This thesis focuses on modeling hierarchical linguistic structure in NLP systems using recent advances in neural network models (known as Deep Learning). Although linguistic structure prediction has been studied exhaustively in the NLP literature (Smith, 2011), what we hope to contribute to in this thesis are the possibilities that neural network approaches can provide and their current capacity of understanding human languages. We explore the expressive representational powers of neural networks for *predicting* as well as *discovering* linguistic structure in both supervised and unsupervised NLP tasks. Furthermore, while achieving state-of-the-art results for many NLP tasks, neural network models have many intriguing properties that we have not yet fully understood. From a computational linguistics perspective, we examine to which extent neural networks are capable of capturing linguistic phenomena implicitly and whether patterns captured by neural networks are identifiable within linguistics.

[1] https://bibinlp.umiacs.umd.edu/

Concretely, we set out to answer the five following research questions in this thesis:

Research question 1: *Do neural networks offer modeling advantages for linguistic structure prediction in comparison to non-neural methods?*

In this research question, we choose machine translation (MT) as the task of interest. Machine translation is a task of building statistical models to translate a source sentence in one language to a target sentence in another language. MT is not only a successful example of commercial NLP products on a global scale, it also poses many interesting challenges for research. To produce meaningful translations, an MT system must capture semantics and syntax to a reasonable degree. Particularly, we tackle the problem of translating from English into morphologically rich languages (MRLs) using additional features produced by neural networks. We divide this research question into two sub-questions:

RQ1.1 *Can neural networks effectively exploit the source-side context to make accurate predictions of target language morphology?*

Morphologically rich languages such as Czech and Russian present a difficult challenge for machine translation due to lexical sparsity and complex morphological agreements. A common strategy to deal with sparsity is to break words into smaller units called *morphemes*. In Chapter 2, we propose a neural network model that learn to predict these morphemes conditioned on a large window of source words.

RQ1.2 *Can neural network models leverage morphological labels to make context-sensitive predictions about morphology?*

Many low-resource languages do not have any morphological analyzer. Could we still build a better translation system that translate into those languages? Here, we make a reasonable assumption that there exists a dictionary that tells us about out of context morphological analyses. We propose a method that exploits this information to predict context sensitive morphology in Chapter 3.

Having shown that neural networks are a powerful tool for NLP tasks, we investigate their learned representations from a linguistic perspective with the next research question.

Research question 2: *From a linguistic perspective, what makes recurrent neural networks work so well for language modeling?*

Recurrent neural networks (RNNs), in particular Long Short-Term Memory (LSTM) networks, are often a favored choice for language modeling due to their excellent performance on a wide range of NLP tasks. Recently, Melis et al. (2018) showed that under a controlled setup that eliminates the source of hyperparameter variation, properly regularized LSTMs outperform many recent architectures. We investigate what linguistic phenomena LSTMs capture in their hidden states enabling them to make accurate predictions. We divide our research question into two sub-questions:

RQ2.1 *Can recurrent neural networks (RNNs) be made more interpretable?*

In order to perform a precise linguistic analysis, we need to create an interpretable interface with LSTMs' internal representations. This interface should give us a meaningful insight beyond the activation values of their hidden units. To achieve this goal, we propose Recurrent Memory Networks (RMNs), a set of models that allow us to inspect neural network computations as well as probing their capacity to learn structure as a by-product of the language modeling objective.

RQ2.2 *What linguistic phenomena captured by RNNs make them so successful in modeling language?*

To answer this question, we perform in-depth analyses of our models with respect to known linguistic phenomena. We find that the networks learn semantic collocations and weakly capture some syntactic dependencies.

RQ2.3 *Do RMNs offer any extra modeling power in addition to their interpretability?*

We show that RMNs do not sacrifice any predictive performance for their interpretability. In contrast, with a simple modification of RMNs, we obtain state-of-the-art performance on Sentence Completion Challenge (Zweig and Burges, 2012).

Recently, a new class of non-recurrent neural networks (Vaswani et al., 2017; Gehring et al., 2017) has demonstrated competitive performances on language understanding. One advantage of non-recurrent architectures is that their computations are highly parallelizable, thus they are faster to train in comparison to recurrent based architectures. However their ability to capture hierarchical structure, which is crucial in language, has not yet been assessed. We address this with the following research question.

Research question 3: *Do non-recurrent neural networks have the same ability to exploit hierarchical structures implicitly in comparison to their recurrent counterpart?*

We are interested in a particular non-recurrent architecture, namely the Transformer (Vaswani et al., 2017). Similar to RMNs, it offers the benefit of interpretability thanks to a self-attention mechanism. Because the main building block of Transformer is the attention mechanism, we refer to Transformer as Fully Attention Network (FANs) to emphasize this characteristic. To address **research question 3**, we ask two sub-questions:

RQ3.1 *Do FANs have the same ability as LSTMs to exploit hierarchical structure in natural language data?*

We assess whether FANs capture non-trivial syntactic dependencies and implement syntactic structure faithfully using a subject-verb agreement task proposed by Linzen et al. (2016). The task was originally proposed to probe the recurrent neural networks' ability to learn hierarchical structure in natural language. In order to solve this task, FANs must infer the dependency between subject and verb without being provided with syntactic annotations.

RQ3.2 *Do FANs have the same ability as LSTMs to encode artificial tree-structured data?*

We assess whether FANs can exploit the given tree structure in logical statements to reason about the relationship between two statements. The task was originally proposed by Bowman et al. (2015b) to evaluate the ability to learn tree-structured composition of recurrent neural networks.

In the previous research questions, we have studied neural networks in supervised settings. With supervised objectives, neural networks can extract useful features and discover shallow linguistic structure to solve the task of interest. However, supervision requires high-quality labeled data which is often hard to obtain. Therefore we also examine the potential of neural networks in an unsupervised setting. Therefore, we ask:

Research question 4: *Can neural networks be used to induce linguistic structure in a completely unsupervised manner?*

While the success of neural networks in supervised learning linguistic structure is undeniable, it is well worth to ask whether neural networks offer the same expressive power in a fully unsupervised learning setting. Unsupervised linguistic structure prediction has played an important role in many NLP applications. For instance, in phrase-based statistical machine translation (PSMT), word alignment is purely unsupervised and it is the key component in building translation models, which often are regarded as the heart of PSMT. Another example of a successful application of unsupervised learning is topic modeling (Blei et al., 2003), which has been widely used for instance in information retrieval systems.

We investigate this question by studying a simple directed graphical model, namely the Hidden Markov Model.

RQ4.1 *Can an unsupervised Hidden Markov Model be fully expressed as a neural network model?*

We present an unsupervised Neural Hidden Markov Models (NHMMs) that are fully parametrized by neural networks. Our NHMMs can be trained end-to-end using stochastic gradient descent.

RQ4.2 *What are the advantages of the neural methods in comparison to traditional Bayesian methods?*

We demonstrate how easy it is to integrate additional features such as character-level information into our NHMMs for the purpose of inducing syntactic categories. Using character-level representations, our models exploit useful syntactic clues and obtain state-of-the-art results in part-of-speech induction within a generative framework.

RQ4.3 *Are neural models sensitive to parameter initialization in the same way as non-neural models are?*

We provide an ablation study of unsupervised HMMs with respect to weight initializations and neural network architectures.

Having demonstrated the advantage of neural network parametrization for an unsupervised latent variable in recovering the underlying linguistic structure, we come to the final research question in this thesis. Namely, we investigate what structure a neural network model would discover to maximize the performance of an NLP task. Here, we ask:

Research question 5: *What type of model can learn to induce linguistic structure and utilize those learnt structures to improve task-specific performance?*

The field of NLP has longed prioritized structured model representations under the assumption that hierarchical structure is a fundamental property of language and therefore often a prerequisite for NLP tasks. While recent research (Shi et al., 2016; Linzen et al., 2016; Belinkov et al., 2017) has shown that some aspects of language are captured implicitly by recurrent neural networks, our goal here is to build a model that operates explicitly over a *latent* tree structure of the input. Specifically, we investigate whether the success and apparent necessity of attention mechanisms are related to their ability to capture these structural/hierarchical properties of language. We study this question in the context of neural machine translation. We present a structured attention NMT model (SA-NMT) in Chapter 7 with two aims: First we hope that our model will discover useful latent structure that improves machine translation. Second, through careful design, our model simultaneously allows us to investigate its representations and learned structure from a linguistic perspective using widely accepted linguistic theories. With SA-NMT we answer three important questions:

RQ5.1 *Can we design a translation model that learns to induce a dependency tree-like representation of the source sentence?*

We present a structured attention augmented encoder for NMT. To bias our model towards using dependency syntax of the source side, we design a gating control mechanism together with a shared attention mechanism in the decoder. Our model learns to translate while simultaneously inducing the source side dependency trees.

RQ5.2 *Are the dependency trees learnt with the translation objective recognizable from a linguistic perspective?*

We show that our model does not only improve the translation quality but also learns a sensible dependency grammar compared to simple left-branching baselines. We evaluate grammar induction on Universal Dependency datasets and show an intriguing effect of the choice of target languages on grammar induction.

RQ5.3 *Is there any correlation between the interpretability of the learned structure and the translation quality?*

We show that there is no correlation between the quality of translation and the quality of the learned dependency structure when using our SA-NMT.

The findings of this thesis contribute to our understanding of the expressive power of neural networks in NLP. Neural network approaches remove the need for feature engineering in the challenging task of translating into morphologically rich languages. Recurrent neural networks implicitly capture linguistic structure that is beneficial for language modeling. In comparison to non-recurrent counterpart, recurrent neural networks are more effective at exploiting hierarchical structures implicitly. The neural network parametrization enables the integration of linguistic feature seamlessly in part-of-speech induction. Finally, neural networks can be used to induce linguistic structure that improves machine translation and provide a more interpretable interface for testing the usefulness of linguistically motivated features.

1.2 Main Contributions

Here we summarize the main contributions of this thesis to the field of natural language processing.

1.2.1 Algorithmic Contributions

We develop novel neural network models and learning algorithms for predicting and discovering linguistic structure.

1. We propose neural network models for predicting word translation in context. Our approach does not assume the availability of supervised morphological segmentation tools (Chapter 2).

2. We introduce a soft-tag representation and a neural network architecture for predicting target language morphology in machine translation (Chapter 3). Our approach successfully circumvents the problem of ambiguous word analyses and makes it possible to improve translation into MRLs where morphological lexicons but no manually disambiguated corpora exist.

3. We propose Recurrent Memory Networks, a class of memory augmented recurrent neural networks, that provide a more interpretable interface for linguistic analysis of the model's predictions (Chapter 4).

4. We introduce a general framework for unsupervised linguistic prediction using neural network parametrization (Chapter 6). We demonstrate our framework by developing an unsupervised neural HMM. Our framework allows seamless integration of linguistic features into the model.

5. We present a structured attention augmented NMT model that simultaneously translates while inducing dependency trees (Chapter 7). Our model leverages the benefit of structural information while providing an interface to investigate the kind of structures that are needed to maximize translation performance.

1.2.2 Empirical Contributions

We evaluate our proposed models on large scale datasets as well as carring out control experiments to validate our hypotheses. We provide empirical results for each research question asked in this thesis. More specifically:

1. We evaluate our bilingual neural network models on word translation prediction task from English into three morphologically rich languages: Bulgarian, Russian, and Czech. We integrate our models into a phrase-based MT system and evaluate translation performance on a large scale dataset from English to Russian (Chapter 2).

2. We evaluate our proposed models based on soft-tag representations for a morphological re-inflection task and machine translation for two language pairs: English→Italian and English→Russian (Chapter 3).

3. We evaluate the language model performance of recurrent memory networks on three large scale datasets: English, Italian, and German. We perform an analysis along various linguistic dimensions captured by our models. Finally, we evaluate our models on the sentence completion challenge dataset (Chapter 4).

4. We carry out subject-verb agreement and logical inference experiments to compare the ability to exploit hierarchical structures implicitly between recurrent and non-recurrent neural networks (Chapter 5).

5. We conduct an empirical evaluation of our unsupervised neural Hidden Markov Model in the context of part-of-speech induction. We study the effect of hyper-parameters and weight initialization in unsupervised learning (Chapter 6).

6. We evaluate translation quality of our structured attention augmented NMT on three language pairs with different degrees of morphological complexity. We measure the quality of the latent trees induced by our models on Universal Dependency datasets (Chapter 7).

1.3 Thesis Overview

After this chapter, the main content of this thesis is divided into six research chapters. Bellow, we provide the high-level abstract of each chapter.

- Chapter 2 - Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks. In this chapter, we present an approach for predicting target word morphology without relying on any addtional linguistic tools. We integrate our models into a phrase-based MT system and evaluate translation performance. Our results provide an answer to **research question 1.1**.

- Chapter 3 - Neural Inflection Model. In this chapter, we propose a soft-tag representation and a neural network architecture for leveraging the existing morphological analyzers. Our empirical results offer an answer to **research question 1.2**.

- Chapter 4 - Recurrent Memory Networks for Language Modeling. In this chapter, we present a novel class of recurrent neural networks with increased interpretability. We perform a linguistic analysis of our models to answer **research question 2**.

- Chapter 5 - The Importance of Being Recurrent for Modeling Hierarchical Structure. In this chapter, we present subject-verb agreement and logical inference tasks that probe the ability to capture hierarchical structures of non-recurrent neural networks. We provide an empirical answer to **research question 3**.

- Chapter 6 - Unsupervised Neural Hidden Markov Model. In this chapter, we propose a neural network parametrization of an unsupervised HMM. We provide expectation-maximization training algorithm of our models. We evaluate our models in the context of part-of-speech induction and provide an answer to **research question 4**.

- Chapter 7 - Inducing Grammars with and for Neural Machine Translation. In this chapter, we introduce a structured attention augmented NMT model that learns source side dependency trees and utilizes this structural information to improve the translation. The proposed models allow us to extract the dependency tree of the source sentence in a principle manner. Our translation results and grammar induction evaluations provide an answer to **research question 5**.

Finally, Chapter 8 - Conclusions summarizes the main findings in this thesis and suggests possible research directions for future work.

1.4 Origins

The research presented in Chapters 2-4 and Chapter 6 is based on four peer-reviewed publications, while Chapter 5 and Chapter 7 are currently under submission. In the following, we mention the role of each co-author in each publication.

- Chapter 2 is based on Tran, Bisazza, and Monz (2014), *Word Translation Prediction for Morphologically Rich Languages with Bilingual Neural Networks*, published in EMNLP 2014. Bisazza suggested the topic of modeling target morphology in machine translation. Tran developed bilingual neural network models and carried out the experiments. Bisazza and Monz contributed to the text and discussion.

- Chapter 3 is based on Tran, Bisazza, and Monz (2015), *A Distributed Inflection Model for Translating into Morphologically Rich Languages*, published in MT-Summit 2015. Tran proposed the neural network model based on soft-tag representation and carried out the experiments. Bisazza and Monz contributed to the text and discussion.

- Chapter 4 is based on Tran, Bisazza, and Monz (2016a), *Recurrent Memory Networks for Language Modeling*, published in NAACL 2016. Tran proposed RMN, carried out the experiments and wrote most of the paper. Bisazza proposed evaluating on the sentence completion task and performed the analysis. Bisazza and Monz contributed to the discussion and editing.

- Chapter 5 is based on Tran, Bisazza, and Monz (2018), *The Importance of Being Recurrent for Modeling Hierarchical Structure*, published in EMNLP 2018. Tran carried out all the experiments for the tasks suggested by Bisazza and wrote most of the paper. Bisazza and Monz contributed to the text.

- Chapter 6 is based on Tran, Bisk, Vaswani, Marcu, and Knight (2016b), *Unsupervised Neural Hidden Markov Model*, published in Workshop on Structured Prediction for NLP, EMNLP 2016. Tran proposed the unsupervised framework and carried out the experiments. Bisk and Vaswani contributed to the discussion. Tran, Bisk, and Vaswani contributed equally to the text. Marcu and Knight proposed unsupervised structure prediction as a summer internship project and contributed to subsequent discussion.

- Chapter 7 is based on Tran and Bisk (2018), *Inducing Grammars with and for Neural Machine Translation*, published in Workshop on Neural Machine Translation and Generation 2018. Tran proposed the models and performed grammar induction and machine translation experiments. Bisk performed analysis of the induced trees. Both authors contributed equally to the text.

This thesis also indirectly benefits from Toutanova, Brockett, Tran, and Amershi (2016), *A Dataset and Evaluation Metrics for Abstractive Compression of Sentences* *and Short Paragraphs*, published at EMNLP 2016.

Chapter 2

Morphology-aware Bilingual Neural Networks

2.1 Introduction and Research Questions

The ability to make context-sensitive translation decisions is one of the major strengths of phrase-based SMT (PSMT). However, the way PSMT exploits source-language context has several limitations as pointed out, for instance, by Quirk and Menezes (2006) and Durrani et al. (2013). First, the amount of context used to translate a given input word depends on the phrase segmentation, with hypotheses resulting from different segmentations competing with one another. Another issue is that, given a phrase segmentation, each source phrase is translated independently from the others, which can be problematic especially for short phrases. As a result, the predictive translation of a source phrase does not access useful linguistic clues in the source sentence that are outside of the scope of the phrase.

Lexical weighting (Koehn et al., 2003) tackles the problem of unreliable phrase probabilities, typically associated with long phrases, but does not alleviate the problem of context segmentation. An important share of the translation selection task is then left to the language model (LM), which is certainly very effective but can only leverage *target* language context. Moreover, decisions that are taken at early decoding stages—such as the common practice of retaining only the top *n* translation options for each source span—depend only on the translation models and on the target context available in the phrase.

Source context based translation models (Gimpel and Smith, 2008; Mauser et al., 2009; Jeong et al., 2010; Haque et al., 2011) naturally address these limitations. These models can exploit a boundless context of the input text, but they assume that target words can be predicted independently from each other, which makes them easy to integrate into state-of-the-art PSMT systems. Even though the independence assumption is made on the target side, these models have shown the benefits of utilizing source context, especially in translating into morphologically rich languages. One drawback of previous research on this topic, though, is that it relied on rich sets of manually designed features, which in turn required the availability of linguistic annotation tools like POS taggers and syntactic parsers.

In this chapter, we specifically focus on improving the prediction accuracy for word translations. Achieving high levels of word translation accuracy is particularly challenging for language pairs where the source language is morphologically poor, such as English, and the target language is morphologically rich, such as Russian, *i.e.*, language pairs with a high degree of surface realization ambiguity (Minkov et al., 2007).

Unlike previous approaches that require linguistic annotations (Minkov et al., 2007; Kholy and Habash, 2012; Chahuneau et al., 2013) or feature engineering (Gimpel and Smith, 2008), we tackle the challenge using neural networks that learn to extract useful features directly from raw text.

Neural networks will be the main workhorse of this thesis and it is important to justify the choice of modeling frameworks. In this chapter and Chapter 3, we provide empirical answers to the **research question 1:** *Do neural networks offer modeling advantages for linguistic structure prediction in comparison to non-neural methods?* Concretely, in this chapter we ask:

Research question 1.1: *Can neural networks effectively exploit the source-side context to make accurate predictions of target language morphology?*

In the quest to answer this research question, we break it down into a couple of sub-questions:

RQ1.1.a *Can translation be improved by a more accurate selection of the translation options already existing in the SMT models, as opposed to generating new options?*

There are two possible approaches to handle sparsity in the target language: (1) to build a model that can provide accurate prediction for all observed word forms, and (2) to build a model that can generate potentially unseen word forms. In Section §2.2, we show that the first approach should be treated as priority.

RQ1.1.b *How to model target morphology using neural networks?*

Our end goal is to improve PSMT, therefore it is important to design a model that can be integrated seamlessly into PSMT systems. We provide a probabilistic view of our model (§2.3) and the neural network implementations (§2.4). Besides directly predicting fully inflected forms as Jeong et al. (2010), our model can also predict stems and suffixes explicitly. Prediction accuracy is evaluated with respect to three morphologically rich target languages (Bulgarian, Czech, and Russian) showing that our approach consistently yields substantial improvements over a competitive baseline. We also show that these improvements in prediction accuracy can be beneficial in an end-to-end MT scenario by integrating it into a large-scale English-Russian PSMT system. Finally, a detailed analysis shows that our approach induces a positive bias on phrase translation probabilities leading to a better ranking of the translation options employed by the decoder.

The rest of this chapter is organized as follow: We perform an analysis of lexical coverage to answer **RQ1.1.a** in Section §2.2. We then present a general framework of predicting word translation in Section §2.3. Next, we answer **RQ1.1.b** by introducing bilingual neural network (BNN) models (§2.4) and evaluating their performance on word translation prediction (§2.5) as well as machine translation (§2.6). We summarize related word in Section §2.7. In Section §2.8 we note the limitations of our approach. Finally, we conclude this chapter in Section §2.9.

2.2 Lexical Coverage of SMT Models

The first question we ask is whether translation can be improved by a more accurate selection of the translation options already existing in the SMT models, as opposed to generating new options. To answer this question we measure the lexical coverage of a baseline PSMT system trained on English-Russian.[1] We choose this language pair because of the morphological richness on the target side: Russian is characterized by highly inflectional morphology with particularly complex nominal declensions (six core cases, three genders and two number categories).

As suggested by Green and DeNero (2012), we compute the recall of reference tokens in the set of target tokens that the decoder could produce in a translation of the source sentence, that is the target tokens of all phrase pairs that matched the input sentence and that were actually used for decoding. This corresponds to the top 30 phrases sorted by weighted phrase, lexical and LM probabilities, for each source span. Koehn (2004a) and our own experience suggest that using more phrases has little or no impact on MT quality. We call this the decoder's *lexical search space*.

Next, we compare the reference/space recall against the reference/MT-output recall: that is, the percentage of reference tokens that also appeared in the 1-best translation output by the SMT system.

Results for the WMT12 benchmark are presented in Table 2.1. From the first two rows, we see that only a rather small part of the correct target tokens available to the decoder are actually produced in the 1-best MT output (50% against 86%).

[1] Training data and SMT setup are described in Section 2.6.

Token recall	
reference / MT-search-space	86.0%
reference / MT-output	50.0%
stem-only reference / MT-output	12.3%
of which reachable	11.2%

Table 2.1: Lexical coverage analysis of the baseline SMT system English-Russian WMT12.

Although our word-level analysis does not directly estimate phrase-level coverage, these numbers suggest that a large potential for translation improvement lies in better lexical selection during decoding.

To quantify the importance of morphology, we count how many reference tokens matched the MT output only at the stem level[2] and for how many of those the correct surface form existed in the search space (reachable matches). These two numbers represent the upper bound of the improvement achievable by a model only predicting suffixes given the target stems.

As shown in Table 2.1, such a model could potentially increase the recall ratio reference/MT-output by 12.3% with generation of new inflected forms, and by 11.2% without. Thus, also when it comes to morphology, generation seems to be of secondary importance compared to better selection in our experimental setup.

2.3 Predicting Word Translations in Context

It is standard practice in PSMT to use word-to-word translation probabilities as an additional phrase score (Koehn et al., 2003). More specifically, state-of-the-art PSMT systems employ maximum likelihood estimates of the context-independent probability of a target word given its aligned source word $P(t_j \mid s_i)$ for each word alignment link a_{ij}.

The main goal of our work is to improve the estimation of such probabilities by exploiting the context of s_i, which in turn we expect will result in better phrase translation selection. Figure 1 illustrates this idea: the translation of *law* in this example has the wrong case—nominative instead of genitive. Due to the rare word *Indiana*, the target LM must backoff to the bigram history and does not penalize this choice sufficiently. However, a model that has access to the word *of* in the near source context could bias the translation of *law* towards the correct case.

We then model $p(t_j \mid c_i)$ with source context c_i defined as a fixed-length word sequence centered around s_i: $c_i = [s_{i-k}, \cdots, s_{i+k}]$. Our definition of context is

[2] Word segmentation for this analysis is performed by the Russian Snowball stemmer, see also Section §2.5.3.

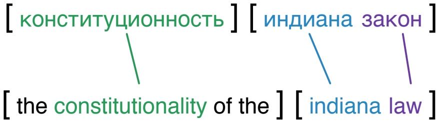

[конституционность] [индиана закон]
[the constitutionality of the] [indiana law]

Figure 2.1: Fragment of English sentence and its incorrect Russian translation produced by the baseline SMT system. Square brackets indicate phrase boundaries. word translations are marked by the same color, word alignment links are provided. Best viewed in color.

similar to the $n-1$ word history used in n-gram LMs. Similarly to previous work in source context-sensitive translation modeling (Jeong et al., 2010; Chahuneau et al., 2013), target words are predicted independently from each other, which allows for an efficient decoding integration. We are particularly interested in translating into morphologically rich languages where source context can provide useful information for the prediction of target translation. For example, the gender of the subject in a source sentence constrains the morphology of the translation of the source verb. Therefore, we integrate the notions of stem and suffix directly into the model. We assume the availability of a word segmentation function g that takes a target word t as input and returns its stem and suffix: $g(t) = (\sigma, \mu)$. Then, the conditional probability $p(t_j \mid c_i)$ can be decomposed into the stem probability and the suffix probability:

$$p(t_j \mid c_i) = p(\sigma_j \mid c_i)\, p(\mu_j \mid c_i, \sigma_j) \tag{2.1}$$

These two probabilities can be estimated separately, which yields the two subtasks:

1. predict target stem σ given source context c;
2. predict target suffix μ given source context c and target stem σ.

Based on the results of our analysis, we focus on the selection of existing translation candidates. We then restrict our prediction on a set of possible target candidates depending on the task instead of considering all target words in the vocabulary. More specifically, for each source word s_i, our candidate generation function returns the set of target words $T_s = \{t_1, \ldots, t_m\}$ that were aligned to s_i in the parallel training corpus, which in turn corresponds to the set of target words that the SMT system can produce for a given source word. In practice, we use a pruned version of T_s to speed up training and reduce noise (see details in Section 2.5).

As for the morphological models, given T_s and g, we can obtain $L_s = \{\sigma_1, \ldots, \sigma_k\}$, the set of possible target stem translations of s, and $M_\sigma = \{\mu_1, \ldots, \mu_l\}$, the set of possible suffixes for a target stem σ. The use of L_s, and M_σ is similar to stemming and inflection operations in (Toutanova et al., 2008) while the set T_s is similar to the GEN function in (Jeong et al., 2010). Note that our suffix generation function M_σ is restricted to the forms observed in the target monolingual data, but not to those aligned to a source word s, which opens the possibility of generating inflected forms that are missing from the translation models. We leave this possibility to the future.

Our approach differs from previous work (Chahuneau et al., 2013; Minkov et al., 2007) in that it does not require linguistic features such as part-of-speech tags and syntactic trees on the source side. The proposed models automatically learn features that are relevant for each of the modeled tasks directly from word-aligned data. To make the approach completely language independent, the word segmentation function g can be trained with an unsupervised segmentation tool. The effects of using different word segmentation techniques are discussed in Section 2.5.

2.4 Bilingual Neural Networks for Translation Prediction

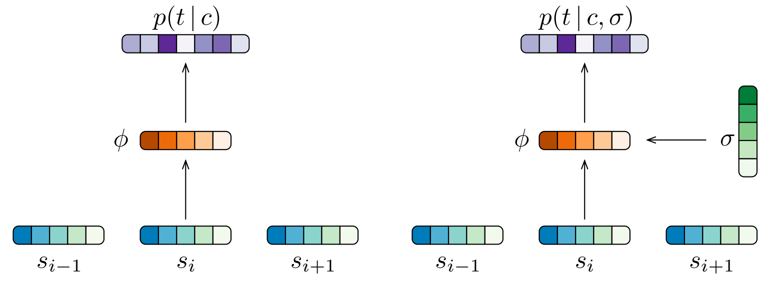
Probabilistic neural networks (NNs), or continuous space language models have received increasing attention over the last few years and have been applied to several natural language processing tasks (Bengio et al., 2003; Collobert and Weston, 2008; Socher et al., 2011, 2012). Within statistical machine translation, they have been used for monolingual target language modeling (Schwenk et al., 2006; Le et al., 2011; Duh et al., 2013; Vaswani et al., 2013), n-gram translation modeling (Le et al., 2012), phrase translation modeling (Schwenk, 2012; Zou et al., 2013; Gao et al., 2014) and minimal translation modeling (Hu et al., 2014). The recurrent neural network LMs of Auli et al. (2013) are primarily trained to predict target word sequences. However, they also experiment with an additional input layer representing source side context.

Our models differ from most previous work in neural language modeling in that we predict a target translation given a source context while previous models predict the next word given a target word history. Unlike previous work in phrase translation modeling with NNs, our models have the advantage of accessing source context that can fall outside the phrase boundaries.

We now describe our models in a general setting, predicting target translations given a source context, where target translations can be either words, stems or suffixes. The source code of our models is available at https://bitbucket.org/ketran/morphbinn.

2.4.1 Neural Network Architecture

Following a common approach in deep learning for NLP (Bengio et al., 2003; Collobert and Weston, 2008), we represent each source word s_i by a column vector $\boldsymbol{r}_{s_i} \in \mathbb{R}^d$ where d is the dimensionality of word embedding vectors. Given a source context $c_i = s_{i-k}, \cdots, s_{i+k}$ of k words on the left and k words on the right of s_i, the context representation $\boldsymbol{r}_{c_i} \in \mathbb{R}^{(2k+1)\times d}$ is obtained by concatenat-



(a) BNN for word prediction.

(b) BNN for suffix prediction.

Figure 2.2: Feed-forward BNN architectures for predicting target translations: (a) word model (similar to stem model), and (b) suffix model with an additional set of vector representations $\boldsymbol{r}_\sigma$ for target stems σ.

ing the vector representations of all words in c_i:

$$\boldsymbol{r}_{c_i} = [\boldsymbol{r}_{s_{i-k}}; \cdots ; \boldsymbol{r}_{s_{i+k}}] \tag{2.2}$$

Our main BNN architecture for word or stem prediction (Figure 2.2a) is a feed-forward neural network (FFNN) with one hidden layer, a weight matrix $\boldsymbol{W}_1 \in \mathbb{R}^{n\times(2k+1)d}$ connecting the input layer to the hidden layer, a matrix $\boldsymbol{W}_2 \in \mathbb{R}^{|V_t|\times n}$ connecting the hidden layer to the output layer, a bias vector $\boldsymbol{b}_1 \in \mathbb{R}^n$, and a bias vector $\boldsymbol{b}_2 \in \mathbb{R}^{|V_t|}$ where $|V_t|$ is the size of target translations vocabulary and n is the number of hidden units. The target translation distribution $p_{\text{BNN}}(t \mid c_i; \boldsymbol{\theta})$ for a given source context c_i is computed by a forward pass:

$$\text{softmax}\left(\boldsymbol{W}_2\, \phi(\boldsymbol{W}_1 \boldsymbol{r}_{c_i} + \boldsymbol{b}_1) + \boldsymbol{b}_2\right) \tag{2.3}$$

where ϕ is a nonlinearity (tanh, sigmoid or rectified linear units). The parameters of the neural network are $\boldsymbol{\theta} = \{\boldsymbol{r}_{s_i}, \boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2\}$.

The suffix prediction BNN is obtained by adding the target stem representation $\boldsymbol{r}_\sigma$ to the input layer (see Figure 2.2b).

We encounter two major issues with FFNNs:

1. They do not provide a natural mechanism to compute word surface conditional probability $p(t \mid c)$ given individual stem probability $p(\sigma \mid c)$ and suffix probability $p(\mu \mid c, \sigma)$;

2. FFNNs do not provide the flexibility to capture long dependencies among words if they lie outside the source context window.

Hence, we consider two BNN variants: a log-bilinear model (LBL) and a convolutional neural network model (ConvNet). LBL could potentially address the first issue by factorizing target representations into target stem and suffix representations whereas ConvNets offer the advantage of modeling variable input lengths (Kalchbrenner et al., 2014), addressing the second issue.

2.4.2 Variants

Log-bilinear model. The FFNN models stem and suffix probabilities separately. A log-bilinear model instead could directly model word prediction through a factored representation of target words, similarly to Botha and Blunsom (2014). Thus, no probability mass would be wasted over stem-suffix combinations that are not in the candidate generation function. The LBL model specifies the conditional distribution for the word translation $t_j \in T_{s_i}$ given a source context c_i:

$$p_{\boldsymbol{\theta}}(t_j \mid c_i) = \frac{\exp(s_{\boldsymbol{\theta}}(t_j, c_i))}{\sum\limits_{t \in T_{s_i}} \exp(s_{\boldsymbol{\theta}}(t, c_i))} \tag{2.4}$$

We use an additional set of word representations $\boldsymbol{q}_t \in \mathbb{R}^n$ for target translations t. The LBL model computes a predictive representation $\boldsymbol{h}$ of a source context c_i by taking a linear combination of the source word representations $\boldsymbol{r}_{s_{i+m}}$ with the position-dependent weight matrices $\boldsymbol{W}_m \in \mathbb{R}^{n \times d}$:

$$\boldsymbol{h} = \sum_{m=-k}^{k} \boldsymbol{W}_m \boldsymbol{r}_{s_{i+m}} \tag{2.5}$$

The score function $s_{\boldsymbol{\theta}}(t_j, c_i)$ measures the similarity between the predictive representation $\boldsymbol{h}$ and the target representation $\boldsymbol{q}_{t_j}$:

$$s_{\boldsymbol{\theta}}(t_j, c_i) = \boldsymbol{h}^\top \boldsymbol{q}_{t_j} + b_{t_j} \tag{2.6}$$

Here b_{t_j} is the bias term associated with target word t_j. $s_{\boldsymbol{\theta}}(t_j, c_i)$ can be seen as the negative energy function of the target translation t_j and its context c_i. The parameters of the model thus are $\boldsymbol{\theta} = \{\boldsymbol{r}_s, \boldsymbol{W}_m, \boldsymbol{q}_t, b_t\}$. Our log-bilinear model is a modification of the log-bilinear model proposed for n-gram language modeling in (Mnih and Hinton, 2007).

Convolutional neural network model. This model (Figure 2.3) computes the predictive representation $\boldsymbol{h}$ by applying a sequence of $2k$ convolutional layers $\{\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{2k}\}$. The source context c_i is represented as a matrix $\boldsymbol{m}_{c_i} \in \mathbb{R}^{d \times (2k+1)}$:

$$\boldsymbol{m}_{c_i} = \left[\boldsymbol{r}_{s_{i-k}}; \ldots; \boldsymbol{r}_{s_{i+k}}\right] \tag{2.7}$$

Each convolutional layer $\boldsymbol{L}_i$ consists of a one-dimensional filter $\boldsymbol{m}_i \in \mathbb{R}^{d \times 2}$. Each row of $\boldsymbol{m}_i$ is convolved with the corresponding row in the previous layer resulting in a weight matrix whose number of columns decreases by one. Thus after $2k$ convolutional layers, the network transforms the source context matrix $\boldsymbol{m}_{c_i}$ to a feature vector $\boldsymbol{h} \in \mathbb{R}^d$. A fully connected layer with weight matrix $\boldsymbol{W}$ followed by a softmax layer are placed after the last convolutional layer $\boldsymbol{L}_{2k}$ to perform classification. The parameters of the convolutional neural network model are $\boldsymbol{\theta} = \{\boldsymbol{r}_s, \boldsymbol{m}_j, \boldsymbol{W}\}$. Here, we focus on a fixed length input, however convolutional neural networks may be used to model variable length input (Kalchbrenner et al., 2014; Kalchbrenner and Blunsom, 2013).

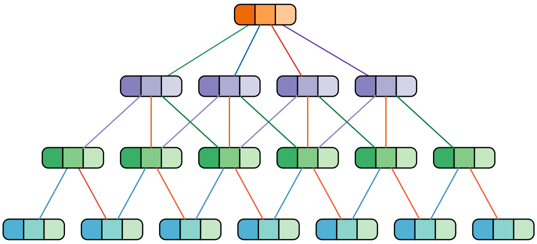


Figure 2.3: Convolutional neural network model. Edges with the same color indicate the same kernel weight matrix.

2.4.3 Training

In training, for each example (t, c), we maximize the conditional probability $P_{\boldsymbol{\theta}}(t \mid c)$ of a correct target label t. The contribution of the training example (t, c) to the gradient of the log conditional probability is given by:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(t \mid c) = \frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(t \mid c) - \sum_{t' \in T_s} p_{\boldsymbol{\theta}}(t' \mid c) \frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(t', c) \tag{2.8}$$

Note that in the gradient, we do not sum over all target translations T but a set of possible candidates T_s of a source word s. In practice $|T_s| \leq 200$ with our pruning settings (see Section 2.5.1), thus training time for one example does not depend on the vocabulary size.

Our training criterion can be seen as a form of contrastive estimation (Smith and Eisner, 2005), however we explicitly move the probability mass from *competing* candidates to the correct translation candidate, thus obtaining more reliable estimates of the conditional probabilities.

The BNN parameters are initialized randomly according to a zero-mean Gaussian. We regularize the models with L_2 norm. As an alternative to the L_2 regularizer, we also experiment with dropout (Srivastava et al., 2014), where the neurons are randomly zeroed out with dropout rate p. This technique is known to be useful in computer vision tasks but has been rarely used in NLP tasks. In FFNN, we use dropout after the hidden layer, while in ConvNet, dropout is applied after the last convolutional layer. The dropout rate p is set to 0.3 in our experiments. We use rectified nonlinearities in FFNN and after each convolutional layer in ConvNet as in our preliminary experiments we find that using rectified linear units gives better results than sigmoid or tanh. We train our BNN models with the standard stochastic gradient descent.

2.5 Evaluating Word Translation Prediction

In this section, we assess the ability of our BNN models to predict the correct translation of a word in context. In addition to English-Russian, we also consider translation prediction for Czech and Bulgarian. As members of the Slavic language family, Czech and Bulgarian are also characterized by highly inflectional morphology. Czech, like Russian, displays a very rich nominal inflection with as many as 14 declension paradigms. Bulgarian, unlike Russian, is not affected by case distinctions but is characterized by a definiteness suffix.

2.5.1 Experimental Setup

The following parallel corpora are used to train the BNN models:

- English-Russian: WMT13 data (News Commentary and Yandex corpora);
- English-Czech: CzEng 1.0 corpus (Bojar et al., 2012) (Web Pages and News sections);
- English-Bulgarian: a mix of crawled news data, TED talks and Europarl proceedings.

Detailed corpus statistics are given in Table 2.2. For each language pair, accuracies are measured on a held-out set of 10K parallel sentences.

	En-Ru	**En-Cs**	**En-Bg**
Sentences	1M	1M	0.8M
Src. tokens	26.5M	19.2M	19.3M
Trg. tokens	24.7M	16.7M	18.9M
Src. T/T	.0109	.0105	.0051
Trg. T/T	.0247	.0163	.0104

Table 2.2: BNN training corpora statistics: number of sentences, tokens, and type/token ratio (T/T). The T/T ratio indicates a degree of lexical variation. A high T/T ratio reflects morphological richness of a language.

To prepare the candidate generation function, each dataset is first word-aligned with GIZA++, then a bilingual lexicon with maximum-likelihood probabilities (P_{mle}) is built from the symmetrized alignment. Alignments are symmetrized using the *grow-diag-final-and* technique (Koehn, 2010). After some frequency and significance pruning, the top 200 translations sorted by $P_{\text{mle}}(t \mid s) \cdot P_{\text{mle}}(s \mid t)$ are kept as candidate word translations for each source word in the vocabulary. Here each lexicon is pruned with minimum word frequency 5, minimum source-target word pair frequency 2, minimum log odds ratio 10. Word alignments are also used to train the BNN models: each alignment link constitutes a training sample, with no special treatment of unaligned words and 1-to-many alignments.

The context window size k is set to 3 (corresponding to 7-gram) and the dimensionality of source word representations to 100 in all experiments. The number of hidden units in our feed-forward neural networks and the target translation embedding size in LBL models are set to 200. All models are trained for 10 iterations with learning rate set to 0.001.

2.5.2 Word, Stem and Suffix Prediction Accuracy

We measure accuracy at top-n, *i.e.*, the number of times the correct translation was in the top n candidates sorted by a model. For each subtask—word, stem and suffix prediction—the BNN model is compared to the context-independent maximum-likelihood baseline $P_{\text{mle}}(t \mid s)$ on which the PSMT lexical weighting score is based. Note that this is a more realistic baseline than the uniform models sometimes reported in the literature. The oracle corresponds to the percentage of aligned source-target word pairs in the held-out set that are covered by the candidate generation function. Out of the missing links, about 4% is due to lexicon pruning. Results for all three language pairs are presented in Table 2.3. In this series of experiments, the morphological BNNs utilize unsupervised segmentation models trained on each target language following (Lee et al., 2011).[3]

Model	**En-Ru**	**En-Cs**	**En-Bg**
	Word prediction (%)		
Baseline	33.0 / 50.1	42.0 / 59.9	47.9 / 66.0
Word BNN	39.4 / 56.6 +6.4 / +6.5	66.6 / 81.4 +24.6/+21.5	56.9 / 74.0 +9.0 / +8.0
Oracle	79.5	90.2	86.9
	Stem prediction (%)		
Baseline	40.7 / 58.2	46.1 / 64.3	51.9 / 70.1
Stem BNN	45.1 / 62.5 +4.4 / +4.3	66.1 / 81.6 +20.0/+17.3	56.7 / 74.4 +4.8 / +4.3
	Suffix prediction (%)		
Baseline	71.2 / 85.6	78.8 / 93.2	81.5 / 92.4
Suffix BNN	77.0 / 89.7 +5.8 / +4.1	91.9 / 97.4 +13.1 /+4.2	87.7 / 94.9 +6.2 / +2.5

Table 2.3: BNN prediction accuracy (top-1/top-3) compared to a context-independent maximum-likelihood baseline. The morphological models in this table make use of unsupervised segmentation.

As shown in Table 2.3, the BNN models outperform the baseline by a large margin for all tasks and languages. In particular, word prediction accuracy at top-1 increases by +6.4%, +24.6% and +9.0% absolute for English-Russian,

[3] We use the implementation at http://groups.csail.mit.edu/rbg/code/morphsyn

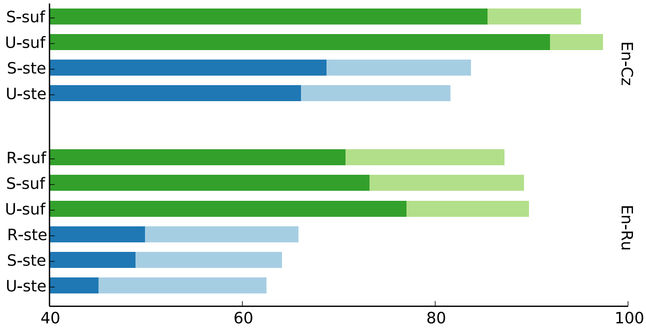


Figure 2.4: Effect of different word segmentation techniques (U: unsupervised, S: supervised, R: rule-based stemmer) on stem and suffix prediction accuracy. The dark part of each bar stands for top-1, the light one for top-3 accuracy.

English-Czech and English-Bulgarian respectively, without the use of any features based on linguistic annotation. While the baseline and oracle differences among languages can be explained by different levels of overlap between training and held-out set, we cannot easily explain why the Czech BNN performance is so much higher. When comparing the three prediction subtasks, we find that word prediction is the hardest task as expected. Stem prediction accuracies are considerably higher than word prediction accuracies for Russian, but almost equal for the other two languages. Finally, baseline accuracies for suffix prediction are by far the highest, ranging between 71.2% and 81.5%, which is primarily explained by a smaller number of candidates to choose from. Also on this task, the BNN model achieves considerable gains of +5.8%, +13.1% and +6.2% at top-1, without the need of manual feature engineering.

From these figures, it is hard to predict whether word BNNs or morphological BNNs will have a better effect on SMT performance. On one hand, the word-level BNN achieves the highest gain over the MLE baseline. On the other hand, the stem- and suffix-level BNNs provide two separate scoring functions, whose weights can be directly tuned for translation quality. A preliminary answer to this question is given by the SMT experiments presented in Section 2.6.

2.5.3 Effect of Word Segmentation

This section analyzes the effect of using different segmentation techniques. We consider two supervised tagging methods that produce the lemma and inflection tag for each token in a context-sensitive manner: TreeTagger (Sharoff et al., 2008) for Russian and the Morce tagger (Spoustová et al., 2007) for Czech.[4] Finally, we employ the Russian Snowball rule-based stemmer as a light-weight context-insensitive segmentation technique.[5]

[4] Annotation included in the CzEng 1.0 corpus release.

[5] http://snowball.tartarus.org/algorithms/russian/stemmer.html

As shown in Figure 2.4, accuracies for both stem and suffix prediction vary noticeably with the segmentation used. However, higher stem accuracies correspond to lower suffix accuracies and vice versa, which can be mainly due to a general preference of a tool to segment more or less than another. In summary, the unsupervised segmentation methods and the light-weight stemmer appear to perform comparably to the supervised methods.

2.5.4 Effect of Training Data Size

We examine the predictive power of our models with respect to the size of training data. Table 2.4 shows the accuracies of stem and suffix models trained on 200K and 1M English-Russian sentence pairs with unsupervised word segmentation. Surprisingly, we observe only a minor loss when we decrease the training data size, which suggests that our models are robust even on a small data set.

# Train sent.	Stem Acc.		Suffix Acc.	
1M	45.1	62.5	77.0	89.7
200K	44.6	61.8	75.7	88.6

Table 2.4: Accuracy at top-1/top-3 (%) of stem and suffix BNNs with different training data sizes.

2.5.5 Fine-grained Evaluation

We evaluate the suffix BNN model at the part-of-speech (POS) level. Table 2.5 provides suffix prediction accuracy per POS for En-Ru. For this analysis, Russian data is segmented by TreeTagger. Additionally, we report the average number of suffixes per stem given the part-of-speech.

Our results are consistent with the findings of (Chahuneau et al., 2013):[6] the prediction of adjectives is more difficult than that of other POS while Russian verb prediction is relatively easier in spite of the higher number of suffixes per stem. These differences reflect the importance of source versus target context features in the prediction of the target inflection: For instance, adjectives agree in gender with the nouns they modify, but this may be only inferred from the target context.

2.5.6 Neural Network Variants

Table 2.6 shows the stem and suffix accuracies of BNN variants on English-Czech.

[6] Chahuneau et al. (2013) report an average accuracy of 63.1% for the prediction of A, V, N, M suffixes. When we train our model on the same dataset (news-commentary) we obtain a comparable result (64.7% vs 63.1%).

POS	A	V	N	M	P
Acc. (%)	49.6	61.9	62.8	84.5	64.4
$\lvert M_\sigma \rvert$	18.2	18.4	9.2	7.1	13.3

Table 2.5: Suffix prediction accuracy at top-1 (%), breakdown by category (A: adjectives, V: verbs, N: nouns, M: numerals and P: pronouns). $\lvert M_\sigma \rvert$ denotes the average number of suffixes per stem.

Although none of the variants outperform our main FFNN architecture, we observe similar performances by the LBL on stem prediction, and by the ConvNet on suffix prediction. This suggests that future work could exploit their additional flexibilities (see Section 2.4.2) to improve the BNN predictive power.

As for the low suffix accuracy by the LBL, it can be explained by the absence of nonlinear transformations. Nonlinearity is important for the suffix model where the prediction of target suffix μ_j often does not depend linearly on s_i and σ_j. The predictive representation of target stem in the LBL stem model, however, mainly depends on the source representation $\boldsymbol{r}_{s_i}$ through a position dependent weight matrix $\boldsymbol{W}_0$. Thus, we observe a smaller drop in accuracy for the stem model than for the suffix model. Conversely, the ConvNet performs poorly on stem prediction because it captures the meaning of the whole source context instead of emphasizing the importance of the source word s_i as the main predictor of the target translation t_j.

Surprisingly, no improvements are obtained by the use of dropout regularization.

Model	**Stem Acc**	**Suffix Acc**
FFNN	66.1 / 81.6	91.9 / 97.4
FFNN + dropout	64.6 / 81.1	91.5 / 97.5
LBL	63.6 / 79.6	86.4 / 96.4
ConvNet + dropout	58.6 / 75.6	90.3 / 96.9

Table 2.6: Accuracies at top-1/top-3 (%) of stem and suffix models. +dropout indicates dropout instead of L_2 regularizer. FFNN is our main architecture.

2.6 SMT Experiments

While the main objective of this paper is to improve prediction accuracy of word translations, see Section 2.5, we are also interested in knowing to which extent these improvements carry over within an end-to-end machine translation task. To this end, we integrate our translation prediction models described in Section 2.4 into our existing English-Russian SMT system.

For each phrase pair matching the input, the phrase BNN score $P_{\text{bnn-p}}$ is computed as follows:

$$p_{\text{bnn-p}}(\bar{s}, \bar{t}, a) = \prod_{i=1}^{|\bar{s}|} \begin{cases} \frac{1}{|\{a_i\}|} \sum_{j \in \{a_i\}} p_{\text{bnn}}(t_j \mid c_i) & \text{if } |\{a_i\}| > 0 \\ p_{\text{mle}}(\text{NULL} \mid s_i) & \text{otherwise} \end{cases} \tag{2.9}$$

where a is the word-level alignment of the phrase pair $(\bar{s}, \bar{t})$ and $\{a_i\}$ is the set of target positions aligned to s_i. Note that the equation above is similar to lexical weighting equation in standard PSMT system. The advantage here is that the probabilities used in our equation come from neural networks, which is known for producing better estimation.

If a source-target link cannot be scored by the BNN model, we give it a p_{bnn} probability of 1 and increment a separate count feature ε. Note that the same phrase pair can get different BNN scores if used in different source side contexts.

Our baseline is an in-house phrase-based (Koehn et al., 2003) statistical machine translation system very similar to Moses (Koehn et al., 2007). All system runs use hierarchical lexicalized reordering (Galley and Manning, 2008; Cherry et al., 2012), distinguishing between monotone, swap, and discontinuous reordering, all with respect to left-to-right and right-to-left decoding. Other features include linear distortion, bidirectional lexical weighting (Koehn et al., 2003), word and phrase penalties, and finally a word-level 5-gram target LM trained on all available monolingual data with modified Kneser-Ney smoothing (Chen and Goodman, 1999). The distortion limit is set to 6 and for each source phrase the top-30 translation candidates are considered. When translating into a morphologically rich language, data sparsity issues in the target language become particularly apparent. To compensate for this we also experiment with a 5-gram suffix-based LM in addition to the surface-based LM (Müller et al., 2012; Bisazza and Monz, 2014).

Recall that PSMT is a log-linear model and integrating our BNN models is straightforward as we treat the predictions of BNN models as additional log-probability ($\log p_{\text{bnn-p}}$) feature functions: one feature for the word prediction model or two features for the stem and suffix models respectively, plus the penalty feature ε. The integration is clean and efficient for training and decoding on CPUs. We *precompute* (Devlin et al., 2014) the hidden units of BNN for the model that does not require target stem and up to the nonlinear transformation for models that require target stem. For each phrase-pair translation candidate, we can trace back to the index of each aligned source word and retrieve the precomputed hidden units. This precomputation trick and the normalization over translation candidates trick allow us to train our PSMT as fast as the baseline PSMT.

Table 2.7 provides some statistics for the data used to train our English-Russian SMT system. The feature weights for all approaches were tuned by using pairwise ranking optimization (PRO) (Hopkins and May, 2011) on the WMT12 benchmark (Callison-Burch et al., 2012). During tuning, 14 PRO parameter estima-

Corpus	**Language**	**#Sentences**	**#Tokens**
paral.train	EN	1.9M	48.9M
	RU		45.9M
Wiki dict.	EN/RU	508K	–
mono.train	RU	21.0M	390M
WMT2012	EN	3K	64K
WMT2013		3K	56K

Table 2.7: SMT training and test data statistics. All numbers refer to tokenized, lowercased data.

SMT system	dev	test
Baseline	24.7	18.9
+ stem/suff. BNN	25.1	19.3▲
Base+suffLM	24.5	19.2
+ word BNN	24.5	19.3
+ stem/suff. BNN	24.7	19.6▲

Table 2.8: Effect of our BNN models on English-Russian translation quality (BLEU[%]).

tion runs are performed in parallel on different samples of the n-best list after each decoder iteration. The weights of the individual PRO runs are then averaged and passed on to the next decoding iteration. Performing weight estimation independently for a number of samples corrects for some of the instability that can be caused by individual samples. The WMT13 set (Bojar et al., 2013) was used for testing. We use approximate randomization (Noreen, 1989) to test for statistically significant differences between runs (Riezler and Maxwell, 2005).

Translation quality is measured with case-insensitive BLEU[%] using one reference translation. As shown in Table 2.8, statistically significant improvements over the respective baseline (Baseline and Base+suffLM) are marked ▲ at the $p < .01$ level.

Integrating our bilingual neural network approach into our SMT system yields small but statistically significant improvements of 0.4 BLEU over a competitive baseline. We can also see that it is beneficial to add a suffix-based language model to the baseline system. The biggest improvements were obtained by combining the suffix-based language model and our BNN approach, yielding 0.7 BLEU over a competitive, state-of-the-art baseline, of which 0.4 BLEU is due to our BNNs. Finally, one can see that the BNNs modeling stems and suffixes separately perform better than a BNN directly predicting fully inflected forms. This is not surprising because the factorization of a word into stem and suffix enable us to deal with lexical sparsity effectively.

To better understand the BNN effect on the SMT system, we analyze the set of phrase pairs that are employed by the decoder to translate each sentence. This set is ranked by the weighted combination of phrase translation and lexical weighting scores, target language model score and, if available, phrase BNN scores. As shown in Table 2.9, the morphological BNN models have a positive effect on the decoder's lexical search space, increasing the recall of reference tokens among the top 1 and 3 phrase translation candidates. The mean reciprocal rank (MRR) also improves from 0.655 to 0.662. Looking at the 1-best SMT output, we observe a slight increase of reference/output recall (50.0% to 50.7%), which is less than the increase we observe for the top 1 translation candidates (57.6% to 59.0%). One possible explanation is that the new, more accurate translation distributions are overruled by other SMT model scores, like the target LM, that are based on traditional maximum-likelihood estimates. While the suffix-based LMs proved beneficial in our experiments, we speculate that higher gains could be obtained by coupling our approach with a morphology-aware neural LM like the one recently presented by Botha and Blunsom (2014).

2.7 Related Work

While the most relevant literature has been discussed in earlier sections, the following approaches are particularly related to ours: Minkov et al. (2007) and Toutanova et al. (2008) address target inflection prediction with a log-linear model based on rich morphological and syntactic features. Their model exploits target context and is applied to inflect the output of a stem-based SMT system, whereas our models predict target words (or pairs of stem-suffix) independently and are integrated into decoding. Chahuneau et al. (2013) address the same problem with another feature-rich discriminative model that can be integrated in decoding, like ours, but they also use it to inflect on-the-fly stemmed phrases. It is not clear what part of their SMT improvements is due to the generation of new phrases or to better scoring. Jeong et al. (2010) predict surface word forms in context, similarly to our word BNN, and integrate the scores into the SMT system. Unlike us, they rely on linguistic feature-rich log-linear models to do that. Gimpel and Smith (2008) propose a similar approach to directly

Token recall (WMT12):	Baseline	+BNN
reference/MT-search-space [top-1]	57.6%	59.0%
reference/MT-search-space [top-3]	70.7%	70.9%
reference/MT-search-space [top-30]	86.0%	85.0%
reference/MT-search-space [MRR]	0.655	0.662
reference/MT-output	50.0%	50.7%
stem-only reference/MT-output	12.3%	11.5%
of which reachable	11.2%	10.3%

Table 2.9: Target word coverage analysis of the English-Russian SMT system before and after adding the morphological BNN models.

predict phrases in context, instead of words.

All those approaches employed features that capture the global structure of source sentences, like dependency relations. By contrast, our models access only local context in the source sentence but they achieve accuracy gains comparably to models that also use global sentence structure.

2.8 Limitations

While the methods presented in this chapter have clear advantages over previous work, there are some limitations in this work.

First, the accuracy of the target inflection depends on the amount of target context the models can access. Our models make a simplifying assumption that we can predict target inflection solely based on the source context (and a target stem). This assumption does not account for many complicated morphological agreements on the target side. While in principle we can model partially generated target context words, integrating those models into PSMT is non-trivial due to the nature of hypothesis recombination assumption in PSTM.

Second, our methods are designed to work for fusional languages as demonstrated in this chapter. However, there are many morphologically rich languages that are not fusional. Turkish, for instance, is agglutinative. Translating into agglutinative languages may require a different treatment, which is beyond the scope of this chapter.

2.9 Conclusions

In this chapter, we have provided the answer to **Research Question 1.1**. Specifically we have addressed the following question in the context of machine translation:

RQ1.1 *Can neural networks effectively exploit the source-side context to make accurate predictions of target language morphology?*

In order to answer this research question, we divide it into two sub-questions. In the following, we revisit the two sub-questions posed at the beginning of the chapter and highlight the main contributions and findings in addressing each individual sub-questions.

RQ1.1a *Can translation be improved by a more accurate selection of the translation options already existing in the SMT models, as opposed to generating new options?*

We have analyzed the lexical coverage in machine translation's search space and shown that there is a large potential for improving translation quality by having a better selection of the available translation options.

RQ1.1b *How to model target morphology using neural networks?*

We have proposed a general approach to predict word translations in context using bilingual neural network architectures. Unlike previous NN approaches, we model word, stem, and suffix distributions in the target language given the context in the source language. Instead of relying on manually engineered features, our models automatically learn abstract word representations and features that are relevant for the word prediction task. Our preliminary results with LBL and ConvNet architectures suggest that potential improvements may be achieved by factorizing target representations or by dynamically modeling source context size. Evaluated on three morphologically rich languages, our approach achieves considerable gains in word, stem and suffix accuracy over a context-independent maximum-likelihood baseline.

Finally, we have shown that the proposed BNN models can be tightly integrated into a phrase-based SMT system. Our analysis shows that the number of correct target words occurring in highly scored phrase translation candidates increases after integrating the morphological BNNs. However, only few of these end up in the 1-best translation output. Nevertheless, we report a small but statistically significant BLEU improvement over a competitive, large-scale English-Russian baseline

Chapter 3

Neural Inflection Model

3.1 Introduction and Research Questions

In morphologically rich languages (MRLs), words can have many different surface forms depending on the grammatical context. When translating into MRLs, standard statistical machine translation (SMT) models such as phrase translation models and n-gram language models (LMs) often fail to select the right surface form due to the sparsity of observed word sequences (Minkov et al., 2007; Green and DeNero, 2012).

While neural language models (Bengio et al., 2003) address lexical sparsity to a certain degree by projecting word sequences to distributed vector representations, they still suffer from the problem of rare words which is particularly exacerbated in MRLs (Botha and Blunsom, 2014; Jean et al., 2015; Luong et al., 2015b).

A potential solution to overcome data sparsity in MRLs, is to use word representations that separate the grammatical aspects of a word, *i.e.*, inflection, from the lexical ones. Such word representations already exist for many languages in the form of morphological analyzers or lexicons. However, using these resources for statistical language modeling is far from trivial due to the issue of ambiguous word analyses.

Table 3.1 illustrates this problem in Italian, for which a fine-grained morphological lexicon but no sizable disambiguated corpus exists. These morphological analyses[1] clearly contain information that is useful to encourage grammatical agreement and, in this case, detect the highlighted error. Unfortunately, though, the needed information is difficult to access because each word can have multiple analyses. Performing contextual disambiguation during translation is an ill-posed problem because the SMT decoder produces large numbers of ungrammatical word sequences but gold tagged training data is naturally

[1] In this chapter we use the terms *analysis* and *tag* interchangeably to denote fine-grained word annotations provided by a morphological analyzer or lexicon.

SMT	**idee**	**ribelli**	**che**	**circola**
Gloss	*ideas*	*rebellious*	*that*	*circulate*
Analyses	**noun-f:p**	noun-f:p noun-m:p **adj:pos+f+p** adj:pos+m+p …	con **pro-wh** det-wh:f+p det-wh:f+s …	ver:impr+pres+2+s **ver:ind+pres+3+s**

Table 3.1: Example of morphological error in Italian SMT output: the verb form should be plural (*circolano*) and not singular (*circola*) to agree in number with the subject. Most of the words have multiple analyses according to our morphological lexicon of reference (Zanchetta and Baroni, 2005). The correct one in context is highlighted.

composed of grammatical sentences. This issue has also been shown to affect syntactic parsing of SMT output in (Post and Gildea, 2008). Moreover, searching for the optimal tag sequence introduces spurious ambiguity into the SMT decoder. Finally, training a disambiguator requires manually disambiguated data, which is not available in many languages and costly to produce.

In chapter 2, we have shown the potential of improving translation quality by predicting accurately the target inflection. In this chapter, we addresss a similar problem but with a different approach. We will provide the last piece of the answer to our first research question, namely "*Do neural networks offer modeling advantages for linguistic structure prediction in comparison to non-neural methods?* ". Concretely, from above motivation, we ask:

Research question 1.2 *Can neural network models leverage morphological labels to make context-sensitive predictions about morphology?*

To answer this question, we need to answer two sub-questions:

RQ1.2.a *Do we need to resolve morphological ambiguity if we are interested in making prediction of the surface forms of words?*

We argue that morphological ambiguity does not need to be resolved for SMT. Instead, we map words to a space where *all possible* morphological attributes of a word are retained (§3.3.1). Rather than enforcing hard tagging decisions, we let the model operate on *soft* morphological representations (or *soft* tags). The resulting tag set is larger than the original one, but still effective at reducing the lexical sparsity of purely word-based LM.

RQ1.2.b *How can we model target inflection effectively using these soft morphological representations?*

Here we argue that using distributed representations for soft morphological tags can help share statistical strength among overlapping tags, *i.e.*, tags that have some attributes in common. To this end, we propose a neural network model that predicts sequences of soft tags conditioned on rich contextual features (§3.3.2).

To answer **RQ1.2.a**, we emprically show that our soft representation model achieves higher accuracies in re-inflecting translations than a model performing contextual disambiguation. To answer **RQ1.2.b**, we show that our model significantly improves translation quality on two different target MRLs.

The chapter is organized as follows: after reviewing the previous work (Section §3.2), we present our distributed inflection model based on soft morphological representations (Section §3.3). In Section §3.4 we introduce the general experimental setup, followed by a detailed description of the re-inflection experiments (Section §3.5) and the end-to-end SMT experiments (Section §3.6). We conclude with a discussion of SMT output examples and an outlook of future work.

3.2 Previous Work

Previous work on inflection modeling for translation into MRLs has mostly relied on the availability of morphologically disambiguated data to choose the most probable analysis of each word in either a context-independent (Minkov et al., 2007) or context-dependent (Green and DeNero, 2012; Koehn and Hoang, 2007; Subotin, 2011) way. While the former irrevocably discards potentially useful attributes of the words, the latter tasks the inflection model with disambiguating the word sequence under construction, which is difficult given the ill-formedness of SMT output and a cause of spurious ambiguity.

Considerably less work has focused on MRLs where disambiguated data does not exist, with few exceptions where ambiguity is solved by randomly selecting one analysis per word type (Minkov et al., 2007; Toutanova et al., 2008; Jeong et al., 2010).

As for how inflection models are integrated into the SMT system, different strategies have been proposed. Minkov et al. (2007); Toutanova et al. (2008); Fraser et al. (2012) treat inflection as a post-processing task: the SMT model is trained to produce lemmatized target sentences (possibly enhanced with some form of morphological annotation) and afterwards the best surface form for each lemma is chosen by separate inflection models. Some work has focused on the generation of new inflected phrases given the input sentence (Chahuneau et al., 2013) or given the bilingual context during decoding (Koehn and Hoang, 2007; Subotin, 2011). Other inflection models have been integrated to SMT as additional feature functions: e.g. as an additional lexical translation score (Jeong et al., 2010; Tran et al., 2014) or as an additional target language model score (Green and DeNero, 2012). We follow this last strategy, rather than generating new inflections, motivated by previous observations that, when translating into MRLs, a large number of reference inflections are already available in the SMT models but are not selected for Viterbi translation (Green and DeNero, 2012; Tran et al., 2014).

More in general, our work is related to class-based language modeling (Brown et al., 1992) with the major difference that we also condition on source-side context and that we use explicit morphological representations instead of data-driven word clusters (Uszkoreit and Brants, 2008), word suffixes (Müller et al., 2012; Bisazza and Monz, 2014) or coarse-grained part-of-speech tags (Koehn et al., 2008).

Modeling morphology using neural networks has recently shown promising results: in the context of monolingual neural language modeling, Luong et al. (2013); Botha and Blunsom (2014) obtain the vectorial representation of a word by composing the representations of its morphemes. Tran et al. (2014) model translation stem and suffix selection in SMT with a bilingual neural network. Soricut and Och (2015) discover morphological transformation rules from word embeddings learned by a shallow network. As the time this chapter is written, we were not aware of work that leveraged fine-grained morphological tags for neural language or translation modeling.

3.3 A Distributed Inflection Model

In MRLs, the surface form of a word is heavily determined by its grammatical features, such as number, case, tense etc. Choosing the right target word form during translation is a complex problem since some of these features depend on the source context while others depend on the target context (agreement phenomena). These, in turn, can either belong to the word (e.g. the gender of a noun in Italian) or depend on the context in which the word appears (*e.g.*, the gender of an adjective).

We model target language inflection by a Markov process generating a sequence of abstract word representations based on source and target context. This complements previous work focusing on either the former (Avramidis and Koehn, 2008; Chahuneau et al., 2013; Tran et al., 2014) or the latter (Green and DeNero, 2012; Fraser et al., 2012; Botha and Blunsom, 2014; Bisazza and Monz, 2014).

3.3.1 Soft Morphological Representations

As previously stated, it is common for words in MRLs to admit multiple morphological analyses out of context. Rather than trying to disambiguate the analyses in context using for instance conditional random fields (Green and DeNero, 2012; Fraser et al., 2012), we modify the tagging scheme so that each word corresponds to only one tag. To also avoid the loss of useful information incurred when arbitrarily selecting one analysis per word type (Minkov et al., 2007; Jeong et al., 2010), we introduce soft morphological representations, or simply *soft tags*.

Assume that a morphological analysis μ is a set of morphological attributes $a(\mu)$ such as *masculine* or *plural*.

Given a word w, a morphological analyzer or lexicon LEX returns a list of possible analyses of that word $\mathcal{A}_w = \{\mu : (w, \mu) \in \text{LEX}\}$. Then, we can map word w to a unique soft tag r_w by simply taking the union of all its possible morphological attributes, that is:

$$r_w \triangleq \bigcup_{\mu_k \in \mathcal{A}_w} a(\mu_k) \tag{3.1}$$

Table 3.2 illustrates soft tag of the Italian word *"ribelle"*.

Word	ribelle
Analyses	adj:pos+f+s, adj:pos+m+s, noun-f:s, noun-m:s
Soft tag	adj:pos|adj:f|adj:s|adj:m|noun-f:s|noun-m:s

Table 3.2: Example of soft tag obtained from four analyses of the Italian word *"ribelle"*.

Hence, soft tags maintain *all* morphological attributes of a word to denote its grammatical dimension while ignoring the lexical content. This new representation scheme compromises between sparsity and ambiguity, and allows for an efficient integration of our model directly into the decoder as no additional cost is incurred for the local tagging search.

Soft tags can also be seen as the marginalization of μ when predicting a surface word w_i given a lemma ℓ_i and its context c_i (*i.e.*, variables that influence w_i, such as w_{i-1}):

$$\begin{aligned} p(w_i \mid \ell_i, c_i) &= \sum_{\mu_k \in \mathcal{A}_{w_i}} p(w_i \mid \mu_k, \ell_i, c_i) p(\mu_k \mid \ell_i, c_i) \\ &\approx \sum_{\mu_k \in \mathcal{A}_{w_i}} p(\mu_k \mid \ell_i, c_i) \end{aligned} \tag{3.2}$$

assuming that any lemma-analysis pair (ℓ, μ) corresponds to at most one inflected form w. Using soft tags, Equation 3.2 can be approximated by $p(r_w \mid \ell_i, c_i)$.

3.3.2 Inflection Neural Network

Our inflection model, Inf-NN, is trained on word-aligned bilingual data to predict sequences of target soft tags given a fixed-size target history *and* the input source sentence (see Figure 3.1). We adopt a neural language model approach as learning distributed representations for the soft tags can help to share statistical information among overlapping tags (*i.e.*, tags that share some morphological attributes). Moreover, compared to Maximum Entropy models that use lexical features, neural networks can better exploit sparse input features such as lexicalized source context and target lemma features, as well as their interactions, in high dimensional spaces.

We learn distributed representations for both source words and target soft tags. The source word representations are initialized from pre-trained embeddings, which has been shown to encode certain morphological regularities (Soricut and Och, 2015), whereas target tag representations are initialized randomly.

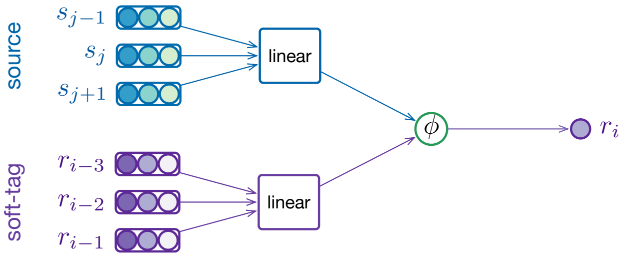


Figure 3.1: Graphical representation of the Inf-NN model: the current target word's soft tag, $\boldsymbol{r}_i$, is predicted based on a fixed-size target tag history and a source side context centered around s_j, the translation of w_i. Each target word w_i can be deterministically mapped to a soft tag $\boldsymbol{r}_i$.

Inf-NN is a feed-forward neural network whose output is a conditional probability distribution over a set of morphological tags given target history and source context. Formally, let $h_i = (r_{i-1}, \ldots, r_{i-n+1})$ be the $n-1$ tag history of the target word w_i, and $c_j = (s_{j-k}, \ldots, s_{j+k})$ the source context centering at the word s_j aligned to w_i by an automatic aligner. We use a simple heuristic similar to the approach by Devlin et al. (2014) to handle null and multiple alignments so that each target word w_i can be mapped to exactly one source word s_j. The heuristic is given as follow:

1. If w_i aligns to multiple source words s_j, then we pick the s_j in the middle.

2. If w_i is unaligned, then we assign the alignment from the closet aligned word to w_i with preference to the right.

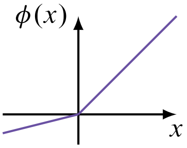
Let $\boldsymbol{s}_j \in \mathbb{R}^d$ and $\boldsymbol{r}_i \in \mathbb{R}^d$ denote the distributed representations of source s_j and target tag r_i respectively. Then, the conditional probability $p_{\text{Inf-NN}}(r_i \mid h_i, c_j)$ is computed at the output layer $\boldsymbol{y}$ of the network as follows:

$$\begin{aligned}
\boldsymbol{z}_i &= \phi(\boldsymbol{U}\boldsymbol{c}_j + \boldsymbol{V}\boldsymbol{h}_i + \boldsymbol{b}_z) \\
\boldsymbol{y} &= \text{softmax}(\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{b}_y) \\
\boldsymbol{c}_j &= [\boldsymbol{s}_{j-k}; \ldots; \boldsymbol{s}_{j+k}] \\
\boldsymbol{h}_i &= [\boldsymbol{r}_{i-1}; \ldots; \boldsymbol{r}_{i-n+1}]
\end{aligned} \tag{3.3}$$

where $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are weight matrices, $[\boldsymbol{v}; \boldsymbol{v}']$ denotes vector concatenation, and ϕ is a non-linear transfer prelu.

As ϕ, we use in all experiments the channel-shared parametric rectified linear unit (PReLU) introduced by (He et al., 2015). PReLU $\phi(x)$ is defined as:

$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$



where a is a parameter learned during training.

Note that the neural network model presented here is just an additional component that will be integreated into a standard PSMT system. Because PSMT systems works on CPUs, the computation in neural network becomes a major bottleneck during decoding. To speed up decoding, we train the Inf-NN model with a self-normalized objective (Devlin et al., 2014; Andreas and Klein, 2015). The self-normalized objective encorages the partition function $Z(x)$ of the output layer of the neural network is close to 1. Thus during decoding, we can avoid computing the partition function. The self-normalized objective proposed by Devlin et al. (2014) is given as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \sum_i \left[\log p\left(r_i \mid h_i, c_j\right) - \alpha \left(\log Z(h_i, c_j) - 0\right)^2 \right] && (3.4) \\ &= \sum_i \log p\left(r_i \mid h_i, c_j\right) - \alpha \log^2 Z(h_i, c_j) && (3.5) \end{aligned}$$

To train a self-normalized model, Andreas and Klein (2015) propose a modification of self-normalized objective proposed by (Devlin et al., 2014):

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_i \boldsymbol{W}_{[r_i]}^\top \boldsymbol{z} - \frac{\alpha}{\gamma} \sum_{(h_i, c_j) \in \mathcal{H}} \log^2 Z(h_i, c_j) \qquad (3.6)$$

where $\mathcal{H}$ is a set of random samples on which self-normalization is performed, $\boldsymbol{\theta} = \left\{\{\boldsymbol{s}_j\}, \{\boldsymbol{r}_i\}, \boldsymbol{W}^c, \boldsymbol{W}^h, \boldsymbol{W}^m, \boldsymbol{b}_z, a\right\}$ are the parameters of the networks, and $Z(h_i, c_j)$ is the partition function of the input (h_i, c_j). In practice, we obtain $\mathcal{H}$ by sampling from a Bernoulli distribution $\mathrm{Bern}(\gamma)$. This is equivalent to applying dropout (Srivastava et al., 2014) on the loss gradient of self-normalization term, where m is the size of a mini-batch. We regularize the networks with L_2 norm.

In our initial experiment, we find that self-normalized objective in equation 3.6 works slighly better, therefore we use this in all subsequent experiments.

3.4 Experimental Setup

We evaluate our approach on two related tasks: re-inflecting reference translations and end-to-end translation from English into MRLs. With the first task, we test the effectiveness of soft morphological representations against

(a) a model that randomly assigns one tag per word type (among its possible tags);

(b) a model that admits multiple tags per word and requires a pre-disambiguated corpus to be trained.

With the second task, we measure translation quality when our inflection model is integrated into a state-of-the-art phrase-based SMT decoder, showing its applicability to languages where no disambiguated data exists.

3.4.1 Data

As target languages, we choose two MRLs belonging to different language families and displaying different inflectional patterns: Russian has very rich nominal, adjectival and verbal inflection, while Italian has moderate nominal and adjectival inflection, but extremely rich verbal inflection. Experiments are performed on the following tasks:

- English-Russian WMT (Bojar et al., 2013): translation of news commentaries with large-scale training data;
- English-Italian IWSLT (Cettolo et al., 2014): translation of speeches with either small-scale training data (TED talks only) or large-scale training data (TED talks and European proceedings).

SMT training data statistics are reported in Table 3.3. The Russian Inf-NN model is trained on a 1M-sentence subset of the bilingual data, while the Italian one is trained on all the data available in each setting. For each data set, we create automatic word alignments using GIZA++ (Och and Ney, 2003).

		En-Ru	**En-It**	
		large	small	large
Bilingual	#sentences	2.4M	180K	2.0M
	src/trg #tokens	49.2M/47.2M	3.6M/3.4M	57.4M/57.0M
	src/trg dict.size	774K/1100K	55K/80K	139K/195K
Monoling.	#sentences	21.0M	2.1M	
	trg #tokens	390M	58.4M	
	src/trg dict.size	2.7M	199K	

Table 3.3: Training corpora statistics.

The ambiguous morphological analyses are obtained from the Russian OpenCorpora lexicon[2] (Bocharov et al., 2013) and from the Italian Morph-it![3] lexicon (Zanchetta and Baroni, 2005). Table 3.4 shows the number of tags and soft tags

[2] opencorpora.org

[3] sslmitdev-online.sslmit.unibo.it/linguistics/morph-it.php

occurring in our training data, as well as the expected counts of analyses per word $\mathbb{E}_w[t]$, words per lemma $\mathbb{E}_l[w]$ and analyses per lemma $\mathbb{E}_l[t]$.

Language	#tags	#soft-tags	$\mathbb{E}_w[t]$	$\mathbb{E}_l[w]$	$\mathbb{E}_l[t]$
Russian	892	4431	3.8	7.2	27.4
Italian	450	901	1.9	12.7	24.3

Table 3.4: Morphological characteristics of the Inf-NN training data: number of tags and soft tags, expected counts of analyses per word $\mathbb{E}_w[t]$, words per lemma $\mathbb{E}_l[w]$ and analyses per lemma $\mathbb{E}_l[t]$.

We find that the Russian tag set and, consequently, the soft tag set are considerably larger than the Italian ones. The average morphological ambiguity is also larger in Russian (3.8 versus 1.9 tags per word). However, somewhat surprisingly, morphological richness is higher in Italian (12.7 versus 7.2 words per lemma). At a closer inspection, we find that most of this richness is due to verbal inflection which goes up to 50 forms for frequently observed verbs.

3.4.2 Neural network Training

The Inf-NN models are trained on a history of 4 target tags and source context of 7 words with the following configuration: Embedding size is set to 200 and the number of hidden units to 768. Target word and soft-tag embeddings are initialized randomly from a Gaussian distribution with mean zero and standard deviation 0.01. Source word embeddings are initialized from pre-trained Glove vectors (Pennington et al., 2014) and rescaled by a factor of 0.1. Weight matrices of linear layers are initialized from a zero-mean Gaussian distribution with standard deviation $\sqrt{2/n_i}$ where n_i is the number of input units (He et al., 2015). We set self-normalization strength $\gamma = 0.02$, Bernoulli parameter $\gamma = 0.1$, and regularization parameter $\eta = 10^{-4}$. All models are trained with a mini-batch size of 128 for 30 epochs. Our stochastic objective functions are optimized using the first-order gradient-based optimizer Adam (Kingma and Ba, 2015). We use the default settings suggested by the authors: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $\lambda = 1 - 10^{-8}$.

3.5 Re-inflection Experiments

The purpose of this experiment is to simulate the behavior of the inflection model during SMT decoding: Given a reference translation and its corresponding source sentence, we re-inflect the former using a simple beam search and count how many times the model recovers the correct surface word form on a 10K-sentence held-out data set. We guide the readers to understand the experimental setup by providing examples of the inputs and gold outputs in Table 3.5.

Table 3.5: Examples of re-inflection task. The model is provided with a source sentence (src) and a target sequence of lemmas (lem), it has to predict the correct surface forms of the target lemmas (tgt).

src	simplicity is the ultimate sophistication
lem	il semplicità essere il ultimo sofisticazione
tgt	la semplicità è l′ ultima sofisticazione
src	a meal without wine is a day without sunshine
lem	un cena senza vino essere come un giorno senza sole
tgt	una cena senza vino è come un giorno senza sole
src	we are born to be real, not to be perfect
lem	essere al mondo per essere vero , non perfetto
tgt	siamo al mondo per essere veri , non perfetti

Since we do not assume the availability of a disambiguator, we also have to deal with lemma ambiguity. While this issue does not affect the definition and training of our Inf-NN, we do need lemmas to determine the set of candidate surface forms $\mathcal{C}_w$ for each word w that is being re-inflected. As a solution, we define $\mathcal{C}_w$ as the union of the surface forms of each possible lemma of w or, more formally, as:

$$\mathcal{C}_w = \{w_i \,|\, \text{lem}(w_i) \cap \text{lem}(w) \neq \varnothing\} \tag{3.7}$$

where $\text{lem}(w)$ denotes the set of lemmas returned by the lexicon for word w. For example, the Italian form *baci* has two possible lemmas: *bacio* (noun: *kiss*) and *baciare* (verb: *to kiss*). Its candidate set $\mathcal{C}_w$ will then include all the forms of the noun *bacio* and all the forms of the verb *baciare*: that is, *bacio, baci, baciamo, baciate, baciano,* etc.

We compare the proposed soft-tag Inf-NN against an Inf-NN trained on randomly assigned tag per type and to another one trained on tag sequences disambiguated by TreeTagger (Schmid, 1994; Sharoff et al., 2008). The latter model must search through a much larger space of morphological tag sequences. Therefore, for a fair comparison, we set a higher beam size when re-inflecting with this model. As another difference from the other models, the TreeTagger-based inflection model relies on the lemmatization performed by TreeTagger to define the candidate set $\mathcal{C}_w$.

To validate the effectiveness of the neural network approach, we also compare Inf-NN to a simpler MaxEnt model trained on a similar configuration. Finally, we evaluate the importance of source-side context features by experimenting with a series of Inf-NN models that are only conditioned on the target tag history.

Since no Italian morphological disambiguator is available to us, we perform this experiment only for Russian.

	MaxEnt	Inf-NN		
	w/ src	w/o src	w/ src	beam
Tree-Tagger: all analyses	56.33	61.19	69.68	200
Random: 1 analysis per word	66.08	72.32	79.92	5
Soft-Reps: 1 soft tag per word	**66.95**	**75.43**	**81.93**	5

Table 3.6: Token-level re-inflection accuracy (%) on a 10K-sentence English-Russian held-out set. The last column indicates the beam size used when searching for the optimal re-inflected sequence.

As shown in Table 3.6, soft tags perform best in all settings and become even more effective when moving from MaxEnt to neural network, demonstrating the importance of learning distributed representations for the soft tags. This result can be explained by the fact that, when fixing one tag per word type either by random assignment or with soft tags, the number of tags per lemma becomes substantially smaller (cf. Table 3.4) and classification easier. On the other hand, the Tree-Tagger based model operating on all word analyses has to deal with spurious ambiguity: that is, a correct sequence of inflected words can correspond to multiple tag sequences that are competing with one another. Solving this problem by marginalizing over the ambiguous analyses (cf. Equation 3.2) can lead to intractable decoding.

The model using soft-tags, which capture all possible morphological attributes of words, performs the best. Even without using source context features, our Inf-NN outperforms the MaxEnt model by 8.5% absolute because of the high dimensional space used to capture complex morphological regularities. By adding source context, we further increase accuracy by 6.5%, leading to an overall gain of 15% over the MaxEnt baseline.

In the next section, we investigate the impact of our most accurate re-inflection model (Soft-Reps Inf-NN) in an end-to-end SMT setting without relying on any disambiguated data.

3.6 End-to-end SMT Experiments

We integrate our Inf-NN model into a phrase-based SMT decoder similar to Moses (Koehn et al., 2007) as an additional log-probability feature function in the log-linear model. Specifically we use $\log p_{\text{Inf-NN}}$ as a new feature.

When a new target phrase $\tilde{w}$ is produced by the decoder, the Inf-NN model returns a probability for each word w_i that composes it, given the previously translated words' soft tags and the source context centered around the source word s_j aligned to w_i. To detect s_j we store phrase-internal word alignments in the phrase table and use simple heuristics to map each target index i to exactly one source index j, as done for the Inf-NN training (Section 3.5). Since every target word corresponds to one soft tag, obtaining the representation of

w_i is trivial (by lookup in a word-tag map) and so is maintaining the target tag history. This crucially differs from previous approaches that distinguish between hypotheses with equal surface forms but different morphological analyses (Koehn et al., 2007), thereby introducing spurious ambiguity into what is already a huge search space.[4] As a result, the integration of our Inf-NN does not affect decoding speed.

3.6.1 Baseline

Our SMT baseline is a competitive phrase-based SMT system including hierarchical lexicalized reordering models (Galley and Manning, 2008) and a 5-gram target LM trained with modified Kneser-Ney smoothing (Chen and Goodman, 1999). Since the large English-Italian data comes from very different sources (TED talks and European proceedings), we construct phrase table and reordering models for this experiment using the fill-up technique (Bisazza et al., 2011). Note that our baseline does not include previously proposed inflection models because the main goal of our experiment is to demonstrate the effectiveness of the proposed approach for languages where no sizable disambiguated data exists, which is indeed the case for Italian.

Feature weights are tuned with pairwise ranking optimization (PRO) (Hopkins and May, 2011) on the union of IWSLT's *dev10* and *test10* in Italian, and on the first 2000 lines of *wmt12* benchmark in Russian (Callison-Burch et al., 2012). During tuning, 14 PRO parameter estimation runs are performed in parallel on different samples of the n-best list after each decoder iteration. The weights of the individual PRO runs are then averaged and passed on to the next decoding iteration. Performing weight estimation independently for a number of samples corrects for some of the instability that can be caused by individual samples.

3.6.2 Results

Translation quality is measured by case-insensitive BLEU (Papineni et al., 2002) on IWSLT's *test12* and *test14* in Italian, and on WMT *newstest2013* and *newstest2014* for Russian, all provided with one reference translation. To see whether the differences between the approaches we compared in our experiments are statistically significant, we apply approximate randomization (Noreen, 1989). Riezler and Maxwell (2005) have shown that approximate randomization is less sensitive to Type-I errors, *i.e.*, less likely to falsely reject the null hypothesis, than bootstrap resampling (Koehn, 2004b) in the context of SMT.

[4] (Green and DeNero, 2012) also tag each target phrase in context as it is produced. However, they avoid the spurious ambiguity problem by only preserving the most probable tag sequence for each phrase (incremental greedy decoding).

	Data	Test	Baseline	Inf-NN
en→ru	large	*newstest2013*	19.0	**19.3**$^{\blacktriangle}$(+0.3)
		newstest2014	26.1	**26.7**$^{\blacktriangle}$(+0.6)
en→it	small	*iwslt12*	24.6	**25.6**$^{\blacktriangle}$(+1.0)
		iwslt14	20.4	**20.9**$^{\blacktriangle}$(+0.5)
	large	*iwslt12*	25.0	**25.8**$^{\blacktriangle}$(+0.8)
		iwslt14	20.9	**21.4**$^{\blacktriangle}$(+0.5)

Table 3.7: Impact on translation quality of the Inf-NN model. $^{\blacktriangle}$ marks significance level $p < .01$.

Results are presented in Table 3.7. Our Inf-NN model consistently leads to significant improvements over a competitive baseline, for both language pairs and all test sets, without affecting decoding speed. By comparing the two data conditions in English-Italian, we see that most of the BLEU gain is preserved even after adding a large amount of parallel training data. This suggests that morphological phenomena are not sufficiently captured by phrases and stresses the importance of specifically modeling word inflection. It is possible that adding even more training data would reduce the impact of our inflection model, but currently we do not have access to other data sets that would be relevant to our translation tasks.

To put these results into perspective, our improvements are comparable to those achieved by previous work (Chahuneau et al., 2013) that generated new phrase inflections using a morphological disambiguator on the same large-scale English-Russian task.

3.6.3 Examples

As previously mentioned, most previous approaches to inflection modeling for SMT are not applicable to languages for which morphologically disambiguated data is not available. It is then particularly interesting to analyze *how* our model affects baseline translations. Table 3.8 presents a number of English-Italian SMT output examples where the use of our soft-tag Inf-NN either resulted in a better inflection choice (1-3) or not (4-5). Out of the 'good' examples, only (1) resulted in a complete match with the reference translation, while in (2) and (3) the system preferred an equally appropriate lexical choice, showing that automatically evaluating inflection models in an SMT setting is far from trivial.

The usefulness of source-side features is demonstrated by example (3): here, the translation of *broken* should agree in gender with the subject *he* but the baseline system chose instead a feminine form (*infranta*). Since the subject pronoun can be dropped in Italian, this error cannot be detected by the target language model and may only be fixed by translating the sequence *'he died broken'* as a

(1)	src	and if you’re wondering about **those other spikes**, those are also fridays
	ref	e se vi state chiedendo cosa sono **questi altri picchi**, sono anche loro dei venerdì
	base	e se vi state chiedendo di **queste altre picchi**, sono anche il venerdì
	infnn	e se vi state chiedendo di **questi altri picchi**, sono anche il venerdì
	Effect:	*Correct number agreement between adjectives and noun*
(2)	src	... a three-hour **version** of this that**’s been viewed** four million times
	ref	... una **versione** di tre ore che **è stata vista** 4 milioni di volte
	base	... una **versione** di tre ore di ciò che **è stato visto** 4 milioni di volte
	infnn	... una **versione** di tre ore di questo che **è stata osservata** quattro milioni di volte
	Effect:	*Correct gender agreement between subject and present perfect*
(3)	src	**he** died **broken** by history
	ref	morì **distrutto** dalla storia
	base	morì **infranta** dalla storia
	infnn	morì **devastato** dalla storia
	Effect:	*Correct gender agreement between subject and adjective*
(4)	src	in one , i was **the classic asian student** ...
	ref	in uno ero **la classica studentessa asiatica** ...
	base	in uno stato **il classico asiatica studente** ...
	infnn	in uno stato **il classico asiatico studente** ...
	Effect:	*Encouraged gender agreement between adjectives and noun, but gender is wrong*
(5)	src	in the other , i was **enmeshed** in lives that were precarious
	ref	nell’altro ero **invischiata** tra esistenze precarie
	base	tra l’altro, sono stato profondamente **impegnati** in vita che erano più precaria
	infnn	nell’altro, ero profondamente **impegnati** in vita che erano più precaria
	Effect:	*Failed to encourage gender agreement because surface form is not in the SMT models*

Table 3.8: Examples of SMT output drawn from IWSLT English-Italian *test12* showing the effect of our inflection model on lexical selection.

single phrase, which was never observed in the training data. By contrast, Inf-NN successfully exploited the source-side context and preferred a masculine form (*devastato*).

Next are two unsuccessful examples: in (4) Inf-NN encouraged the system to translate the whole phrase *'the classic asian student'* as masculine whereas the baseline translation used an incoherent mix of masculine and feminine. Unfortunately, though, the *student* in question, *i.e.*, the speaker, happened to be a woman, but this could not be inferred in any way from this sentence. In (5) Inf-NN failed to fix the agreement between adjective and subject pronoun. By inspecting the parallel data we found that the word *enmeshed* always occurred with plural forms of Italian adjectives. This example shows that improving the scoring of the existing translation options is not always sufficient. While we do not address generation of new inflected forms in this work, this is an interesting direction for future work.

3.6.4 Implementation

We implement Inf-NN model in Torch, a scientific computing framework with wide support for machine learning algorithms. After training, we save model’s parameters in JSON[5] (JavaScript Object Notation) format. We then build the same neural network architecture that only computes the forward pass in C++ because C++ provides a natural interface to integrate Inf-NN model into our in-house PSMT. To achieve efficient computation, we rely on the Eigen library to perform matrix multiplications. Because our PSMT system is written in Perl, we use SWIG[6] (Simplified Wrapper and Interface Generator) to connect the Perl program to the neural network module in C++. Additionally, we pre-compute the hidden units of of the source context before non-linear layer. This trick dramatically reduces the computational cost during tuning PSMT's parameters as shown in Devlin et al. (2014). Our implementation is available at https://bitbucket.org/ketran/soft-tags.

3.7 Limitations

Although our approach can leverage the existing morphological analyzers to build a better translation system into MRLs, at the time this thesis is written, we realize that modeling at character-level using LSTMs (Ling et al., 2015) or CNNs (Kim et al., 2016) can remove the need of morphological analyzer (Lee et al., 2017). Given the current state of the field, it seems that the time for PSMT is coming to an end: the future of machine translation looks brighter for deep neural networks trained end-to-end, without pipeline of independent processing steps.

Nevertheless, we believe that character-level modeling is not likely to be the final answer for machine translation. It seems that using morphological analyses can bring some benefit to the task; for instance, Vania and Lopez (2017) report that neural network language models that access the true morphological analyses outperform character-level models. Recent work of (Song et al., 2018) has shown the benefit of modeling target stems and suffixes in a English→Russian neural machine translation (NMT) in comparison to a full character-based NMT (Lee et al., 2017). Thus, we hope that some aspects of our approach can be transferred to neural machine translation.

3.8 Conclusions

In this chapter, we aimed to improve machine translation where the target language is morphologically rich and its morphological lexicon exists. To this end, we have provided an empirical answer to the following research question:

RQ1.2 *Can neural network models leverage morphological labels to make context-sensitive predictions about morphology?*

[5] https://www.json.org
[6] http://www.swig.org

In order to answer this research question, we divide it into two sub-questions. In the following, we revisit the two sub-questions posed at the beginning of the chapter and highlight the main contributions and findings in addressing each individual sub-question.

RQ1.2a *Do we need to resolve morphological ambiguity if we are interested in making prediction of the surface forms of words?*

We have shown that morphological ambiguity does not need to be resolved if the primary concern is to predict the correct surface form of words. We have proposed soft-tag, a novel morphological representation scheme, that circumvents the problem of ambiguous word analyses and makes it possible to improve translation into MRLs where a morphological lexicon but no manually disambiguated corpora exist. We evaluated soft-representations on a re-inflection task and showed that the proposed soft tags achieve significantly higher accuracy than (i) a model using standard tags and trained on morphologically disambiguated data and (ii) a Maximum Entropy model that does not learn distributed representations for source words and target tags.

RQ1.2b *How can we model target inflection effectively using these soft morphological representations?*

We proposed a neural network model that leverages the soft-tags to predict the target inflection accurately. Using distributed representations, our model exploits the advantage of sharing statistical strength in soft-tags. When integrated into a state-of-the-art SMT decoder, our inflection model significantly improves translation quality in two different language pairs, without having to perform morphological disambiguation during decoding. Our results thus demonstrate the applicability of our approach to languages for which no morphological disambiguator exists.

Chapter 4

Recurrent Memory Networks for Language Modeling

4.1 Introduction and Research Questions

In the previous chapters, we have shown that feed-forward neural networks offer many advantages for linguistic structure prediction. In this chapter, we will study a more powerful class of neural networks, namely recurrent neural networks (Elman, 1990; Mikolov et al., 2010).

Recurrent Neural Networks (RNNs) are remarkably powerful models for sequential data. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), a specific architecture of RNN, has shown to be successful for many natural language processing tasks such as language modeling (Józefowicz et al., 2015), dependency parsing (Dyer et al., 2015), sentence compression (Filippova et al., 2015), and machine translation (Sutskever et al., 2014).

Within the context of natural language processing, a common assumption is that LSTMs are able to capture certain linguistic phenomena. Evidences supporting this assumption mainly come from evaluating LSTMs in downstream applications: Bowman et al. (2015b) carefully design two artificial datasets where sentences have explicit recursive structures. They show empirically that while processing the input linearly, LSTMs can implicitly exploit recursive structures of languages. Filippova et al. (2015) find that using explicit syntactic features within LSTMs in their sentence compression model hurts the performance of overall system. They then hypothesize that a basic LSTM is powerful enough to capture syntactic aspects which are useful for compression.

To understand and explain which linguistic dimensions are captured by an LSTM is non-trivial. This is due to the fact that the sequences of input histories are compressed into several dense vectors by the LSTM's components whose purposes with respect to representing linguistic information is not evident. To the best of our knowledge, the only attempt to better understand the reasons of an LSTM's performance and limitations is the work of Karpathy et al. (2016) by means of visualization experiments and cell activation statistics in the context of character-level language modeling.

Our work is motivated by the difficulty in understanding and interpreting existing RNN architectures from a linguistic point of view. Here we ask:

Research question 2: *From a linguistic perspective, what makes recurrent neural networks work so well for language modeling?*

To answer this research question, we divide it into three sub-questions:

RQ2.1 *Can recurrent neural networks (RNNs) be made more interpretable?*

We propose Recurrent Memory Networks (RMNs), a novel RNN architecture that combines the strengths of both LSTM and Memory Network (Sukhbaatar et al., 2015). In RMNs, the Memory Block component—a variant of a Memory Network—accesses the most recent input words and selectively attends to words that are relevant for predicting the next word given the current LSTM state. We demonstrate that our RMN outperforms competitive LSTM baselines in terms of perplexity on three large German, Italian, and English datasets. Importantly, by looking at the attention distribution over history words, our RMN allows us not only to interpret the results but also to discover underlying dependencies present in the data.

RQ2.2 *What linguistic phenomena captured by RNNs make them so successful in modeling language?*

We perform an analysis along various linguistic dimensions that our model captures. This is possible only because the Memory Block allows us to look into its internal states and its explicit use of additional inputs at each time step. We carry out analysis of the attention weights with respect to lexical position and dependency syntax. Our analysis suggests that RMNs capture lexical co-occurrences regardless of their distance as well as some important syntactic dependencies.

RQ2.3 *Do RMNs offer any extra modeling power in addition to their interpretability?*

We show that, with a simple modification, our RMN can be successfully applied to NLP tasks other than language modeling. On the Sentence Completion Challenge (Zweig and Burges, 2012), our model achieves an impressive 69.2% accuracy, surpassing the previous state of the art 58.9% by a large margin.

The rest of the chapter is organized as follow: We briefly review LSTM networks in Section §4.2. We introduce Recurrent Memory Networks in Section §4.3 to answer **RQ2.1**. We then evaluate RMNs on language model benchmarks in Section §4.4. In Section §4.5 we provide detailed analyses along various linguistic dimensions captured by RMNs as the answer to **RQ2.2**. Next, we answer **RQ2.3** by applying RMNs to the sentence completion task in Section §4.6. We point out the limitation of our models in Section §4.7. Finally, we conclude this chapter in Section §4.8.

4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have shown impressive performances on many sequential modeling tasks due to their ability to encode unbounded input histories. However, training simple RNNs is difficult because of the vanishing and exploding gradient problems (Bengio et al., 1994; Pascanu et al., 2013). A simple and effective solution for exploding gradients is gradient clipping proposed by Pascanu et al. (2013). To address the more challenging problem of vanishing gradients, several variants of RNNs have been proposed. Among them, Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Cho et al., 2014) are widely regarded as the most successful variants. In this work, we focus on LSTMs because they have been shown to outperform GRUs on language modeling tasks (Józefowicz et al., 2015). In the following, we will detail the LSTM architecture used in this work.

Long Short-Term Memory

Notation: Throughout this chapter, we denote matrices, vectors, and scalars using bold uppercase (*e.g.*, $\boldsymbol{W}$), bold lowercase (*e.g.*, $\boldsymbol{b}$) and lowercase (*e.g.*, n) letters, respectively. The LSTM used in this work is specified as follows:

$$\begin{aligned}
\boldsymbol{i}_t &= \mathrm{sigm}(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{j}_t &= \mathrm{sigm}(\boldsymbol{W}_{xj}\boldsymbol{x}_t + \boldsymbol{W}_{hj}\boldsymbol{h}_{t-1} + \boldsymbol{b}_j) \\
\boldsymbol{f}_t &= \mathrm{sigm}(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{o}_t &= \tanh(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{c}_{t-1} \odot \boldsymbol{f}_t + \boldsymbol{i}_t \odot \boldsymbol{j}_t, \\
\boldsymbol{h}_t &= \tanh(\boldsymbol{c}_t) \odot \boldsymbol{o}_t
\end{aligned}$$

where $\boldsymbol{x}_t$ is the input vector at time step t, $\boldsymbol{h}_{t-1}$ is the LSTM hidden state at the previous time step, $\boldsymbol{W}_*$ and $\boldsymbol{b}_*$ are weights and biases. The symbol $\odot$ denotes the Hadamard product or element-wise multiplication.

Despite the popularity of the LSTM in sequential modeling, its design is not straightforward to justify and understanding why it works remains a challenge (Hermans and Schrauwen, 2013; Chung et al., 2014; Greff et al., 2017; Józefowicz et al., 2015; Karpathy et al., 2016). There have been few recent attempts to understand the components of an LSTM from an empirical point of view: Greff et al. (2017) carry out a large-scale experiment of eight LSTM variants. The results from their 5,400 experimental runs suggest that forget gates and output gates are the most critical components of LSTMs. Józefowicz et al. (2015) conduct and evaluate over ten thousand RNN architectures and find that the initialization of the forget gate bias is crucial to the LSTM's performance. While these findings are important to help researchers choose appropriate LSTM architectures, they do not shed light on what information is captured by the hidden states of an LSTM.

4.3 Recurrent Memory Network

It has been demonstrated that RNNs can retain input information over a long period. However, existing RNN architectures make it difficult to analyze what information is exactly retained at their hidden states at each time step, especially when the data has complex underlying structures, which is common in natural language. Motivated by this difficulty, we propose a novel RNN architecture called Recurrent Memory Network (RMN). On linguistic data, the RMN allows us not only to qualify which linguistic information is preserved over time and why this is the case but also to discover dependencies within the data (Section §4.5). Our RMN consists of two components: an LSTM and a *Memory Block* (MB) (Section §4.3.1). The MB takes the hidden state of the LSTM and compares it to the most recent inputs using an attention mechanism (Gregor et al., 2015; Bahdanau et al., 2015; Graves et al., 2014). Thus, analyzing the attention weights of a trained model can give us valuable insight into the information that is retained over time in the LSTM.

In the following, we describe in detail the MB architecture and the combination of the MB and the LSTM to form an RMN.

4.3.1 Memory Block

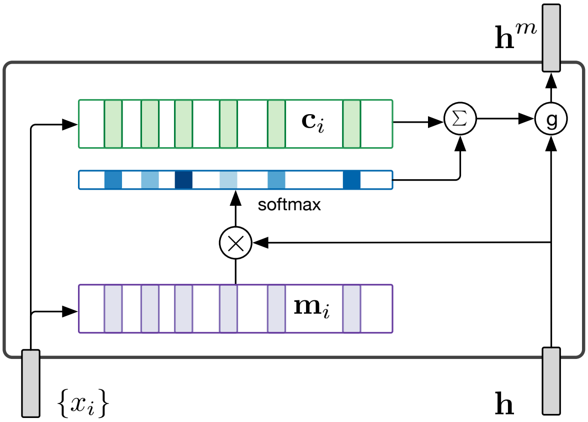


Figure 4.1: A graphical representation of the MB.

The Memory Block (Figure 4.1) is a variant of a Memory Network (Sukhbaatar et al., 2015) with one hop (or a single-layer Memory Network). At time step

t, the MB receives two inputs: the hidden state $\boldsymbol{h}_t$ of the LSTM and a set $\{x_i\}$ of n most recent words including the current word x_t. We refer to n as the memory size. Internally, the MB consists of two lookup tables $\boldsymbol{M}$ and $\boldsymbol{C}$ of size $|V| \times d$, where $|V|$ is the size of the vocabulary and d is the dimensionality of word embeddings. With slight abuse of notation we denote $\boldsymbol{M}_i = \boldsymbol{M}(\{x_i\})$ and $\boldsymbol{C}_i = \boldsymbol{C}(\{x_i\})$ as $n \times d$ matrices where each row corresponds to an input memory embedding $\boldsymbol{M}_i$ and an output memory embedding $\boldsymbol{C}_i$ of each element of the set $\{x_i\}$. We use the matrix $\boldsymbol{M}_i$ to compute an attention distribution over the set $\{x_i\}$:

$$\boldsymbol{p}_t = \operatorname{softmax}(\boldsymbol{M}_i \boldsymbol{h}_t) \tag{4.1}$$

When handling data that exhibits a strong temporal relationship, such as natural language, an additional temporal matrix $\boldsymbol{T} \in \mathbb{R}^{n \times d}$ can be used to bias attention with respect to the position of the data points (Sukhbaatar et al., 2015). In this case, equation 4.1 becomes

$$\boldsymbol{p}_t = \operatorname{softmax}\big((\boldsymbol{M}_i + \boldsymbol{T})\boldsymbol{h}_t\big) \tag{4.2}$$

We then use the attention distribution $\boldsymbol{p}_t$ to compute a context vector representation of $\{x_i\}$:

$$\boldsymbol{s}_t = \boldsymbol{C}_i^\top \boldsymbol{p}_t \tag{4.3}$$

Finally, we combine the context vector $\boldsymbol{s}_t$ and the hidden state $\boldsymbol{h}_t$ by a function $g(\cdot)$ to obtain the output $\boldsymbol{h}_t^m$ of the MB. Instead of using a simple addition function $g(\boldsymbol{s}_t, \boldsymbol{h}_t) = \boldsymbol{s}_t + \boldsymbol{h}_t$ as in Sukhbaatar et al. (2015), we propose to use a gating unit that decides how much it should trust the hidden state $\boldsymbol{h}_t$ and context $\boldsymbol{s}_t$ at time step t. Our gating unit is a form of Gated Recurrent Unit (Cho et al., 2014; Chung et al., 2014):

$$\boldsymbol{z}_t = \operatorname{sigm}(\boldsymbol{W}_{sz}\boldsymbol{s}_t + \boldsymbol{U}_{hz}\boldsymbol{h}_t) \tag{4.4}$$

$$\boldsymbol{r}_t = \operatorname{sigm}(\boldsymbol{W}_{sr}\boldsymbol{s}_t + \boldsymbol{U}_{hr}\boldsymbol{h}_t) \tag{4.5}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}\boldsymbol{s}_t + \boldsymbol{U}(\boldsymbol{r}_t \odot \boldsymbol{h}_t)) \tag{4.6}$$

$$\boldsymbol{h}_t^m = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_t + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{4.7}$$

where $\boldsymbol{z}_t$ is an update gate, $\boldsymbol{r}_t$ is a reset gate.

The choice of the composition function $g(\cdot)$ is crucial for the MB, especially when one of its input comes from the LSTM. The simple addition function might overwrite the information within the LSTM's hidden state and therefore prevent the MB from keeping track of information in the distant past. The gating function, on the other hand, can control the degree of information that flows from the LSTM to the MB's output.

4.3.2 RMN Architectures

As explained above, our proposed MB receives the hidden state of the LSTM as one of its input. This leads to an intuitive combination of the two units by stacking the MB on top of the LSTM. We call this architecture Recurrent-Memory (RM). The RM architecture, however, does not allow interaction between Memory Blocks at different time steps. To enable this interaction we can stack one more LSTM layer on top of the RM (Figure 4.2). We call this architecture Recurrent-Memory-Recurrent (RMR).

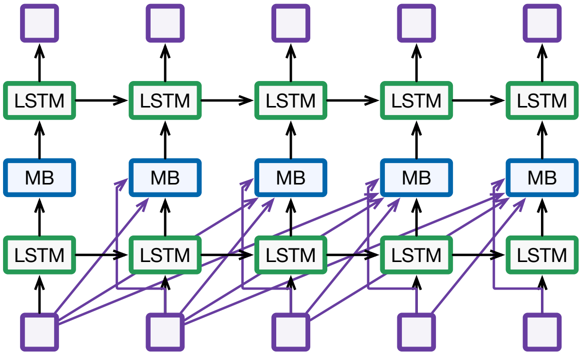


Figure 4.2: A graphical illustration of an unfolded RMR with memory size 4. The MB takes the output of the bottom LSTM layer and the 4-word history as its input. The output of the MB is then passed to the second LSTM layer on top. There is no direct connection between MBs of different time steps. The last LSTM layer carries the MB's outputs recurrently.

4.4 Language Model Experiments

Language models play a crucial role in many NLP applications such as machine translation and speech recognition. Language modeling also serves as a standard test bed for newly proposed models (Sukhbaatar et al., 2015; Kalchbrenner et al., 2016). We conjecture that, by explicitly accessing history words, RMNs will offer better predictive power than the existing recurrent architectures. We therefore evaluate our RMN architectures against state-of-the-art LSTMs in terms of perplexity.

4.4.1 Data

We evaluate our models on three languages: English, German, and Italian. We are especially interested in German and Italian because of their larger vocabularies and complex agreement patterns. Table 4.1 summarizes the data used in our experiments.

The training data consits of approximately 1M sentences in each language. For English, we use all the News Commentary data (8M tokens) and 18M tokens from News Crawl 2014 for training. Development and test data are randomly drawn from the concatenation of the WMT 2009-2014 test sets (Bojar et al., 2015). For German, we use the first 6M tokens from the News Commentary data and

Lang	Train	Dev	Test	$\lvert s \rvert$	$\lvert V \rvert$
En	26M	223K	228K	26	77K
De	22M	202K	203K	22	111K
It	29M	207K	214K	29	104K

Table 4.1: Data statistics. $|s|$ denotes the average sentence length and $|V|$ the vocabulary size.

16M tokens from News Crawl 2014 for training. For development and test data we use the remaining part of the News Commentary data concatenated with the WMT 2009-2014 test sets. Finally, for Italian, we use a selection of 29M tokens from the PAISÀ corpus (Lyding et al., 2014), mainly including Wikipedia pages and, to a minor extent, Wikibooks and Wikinews documents. For development and test we randomly draw documents from the same corpus.

4.4.2 Setup

Our baselines are a 5-gram language model with Kneser-Ney smoothing, a Memory Network (MemN) (Sukhbaatar et al., 2015), a vanilla single-layer LSTM, and two stacked LSTMs with two and three layers respectively. N-gram models have been used intensively in many applications for their excellent performance and fast training. Chen et al. (2016) show that n-gram model outperforms a popular feed-forward language model (Bengio et al., 2003) on a one billion word benchmark (Chelba et al., 2013). While taking longer time to train, RNNs have been proven superior to n-gram models (Mikolov et al., 2010).

We compare these baselines with our two model architectures: RMR and RM. For each of our models, we consider two settings: with or without temporal matrix (+tM or –tM), and linear versus gating composition function. In total, we experiment with eight RMN variants.

For all neural network models, we set the dimension of word embeddings, the LSTM hidden states, its gates, the memory input, and output embeddings to 128. The memory size is set to 15. The bias of the LSTM's forget gate is initialized to 1 (Józefowicz et al., 2015) while all other parameters are initialized uniformly in $(-0.05, 0.05)$. The initial learning rate is set to 1 and is halved at each epoch after the forth epoch. All models are trained for 15 epochs with standard stochastic gradient descent (SGD). During training, we rescale the gradients whenever their norm is greater than 5 (Pascanu et al., 2013).

Sentences with the same length are grouped into buckets. Then, mini-batches of 20 sentences are drawn from each bucket. We do not use truncated backpropagation through time, instead gradients are fully back-propagated from the end of each sentence to its beginning. When feeding in a new mini-batch, the hidden states of LSTMs are reset to zeros, which ensures that the data is modeled at the sentence level. For our RMN models, instead of using padding, at time step $t < n$, we use a slice $\boldsymbol{T}[1:t] \in \mathbb{R}^{t \times d}$ of the temporal matrix $\boldsymbol{T} \in \mathbb{R}^{n \times d}$.

4.4.3 Results

Perplexities on the test data are given in Table 4.2. All RMN variants substantially outperform n-gram and MemN models, and most RMN variants also outperform the competitive LSTM baselines. The best results overall are obtained by RM with temporal matrix and gating composition (+tM-g).

Model		**De**	**It**	**En**
5-gram	–	225.8	167.5	219.0
MemN	1 layer	169.3	127.5	188.2
LSTM	1 layer	135.8	108.0	145.1
	2 layers	128.6	105.9	139.7
	3 layers	125.1	106.5	136.6
RMR	+tM-l	127.5	109.9	133.3
	–tM-l	126.4	106.1	134.5
	+tM-g	126.2	99.5	135.2
	–tM-g	**122.0**	**98.6**	**131.2**
RM	+tM-l	121.5	92.4	**127.2**
	–tM-l	122.9	94.0	130.4
	+tM-g	**118.6**	**88.9**	128.8
	–tM-g	129.7	96.6	135.7

Table 4.2: Perplexity comparison including RMN variants with and without temporal matrix (tM) and linear (l) versus gating (g) composition function.

Our results agree with the hypothesis of mitigating prediction error by explicitly using the last n words in RNNs (Karpathy et al., 2016). We further observe that using a temporal matrix always benefits the RM architectures. This can be explained by seeing the RM as a principled way to combine an LSTM and a neural n-gram model. By contrast, RMR works better without temporal matrix but its overall performance is not as good as RM. This suggests that we need a better mechanism to address the interaction between MBs, which we leave to future work. Finally, the proposed gating composition function outperforms the linear one in most cases.

For historical reasons, we also run a stacked three-layer LSTM and a RM(+tM-g) on the much smaller Penn Treebank dataset (Marcus et al., 1993) with the same setting described above. The respective perplexities are 126.1 and 123.5.

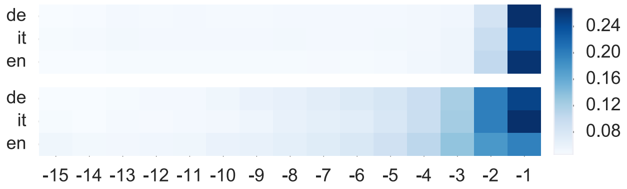


Figure 4.3: Average attention per position of RMN history. Top: RMR(–tM-g), bottom: RM(+tM-g). Rightmost positions represent most recent history.

4.5 Attention Analysis

The goal of our RMN design is twofold: (i) to obtain better predictive power and (ii) to facilitate understanding of the model and discover patterns in data. In Section §4.4, we have validated the predictive power of the RMN and below we investigate the source of this performance based on linguistic assumptions of word co-occurrences and dependency structures.

4.5.1 Positional and Lexical Analysis

As a first step towards understanding RMNs, we look at the average attention weights of each history word position in the MB of our two best model variants (Figure 4.3). One can see that the attention mass tends to concentrate at the rightmost position (the current word) and decreases when moving further to the left (less recent words). This is not surprising since the success of n-gram language models has demonstrated that the most recent words provide important information for predicting the next word. Between the two variants, the RM average attention mass is less concentrated to the right. This can be explained by the absence of an LSTM layer on top, meaning that the MB in the RM architecture has to pay more attention to the more distant words in the past. The remaining analyses described below are performed on the RM(+tM-g) architecture as this yields the best perplexity results overall.

Beyond average attention weights, we are interested in those cases where attention focuses on distant positions. To this end, we randomly sample 100 words from the test data and visualize attention distributions over the last 15 words. Figure 4.4 shows the attention distributions for random samples of German and Italian. Again, in many cases attention weights concentrate around the last word (bottom row). However, we observe that many long distance words also receive noticeable attention mass. Interestingly, for many predicted words, attention is distributed evenly over memory positions, possibly indicating cases where the LSTM state already contains enough information to predict the next word.

To explain the long-distance dependencies, we first hypothesize that our RMN mostly memorizes frequent co-occurrences. We run the RM(+tM-g) model on the German development and test sentences, and select those pairs of (*most-attended-word, word-to-predict*) where the MB's attention concentrates on a word more than six positions to the left. Then, for each set of pairs with equal distance, we compute the mean frequency of corresponding co-occurrences seen in the training data (Table 4.3). The lack of correlation between frequency and memory location suggests that RMN does more than simply memorizing frequent co-occurrences.

d	7	8	9	10	11	12	13	14	15
μ	54	63	42	67	87	47	67	44	24

Table 4.3: Mean frequency (μ) of (*most-attended-word, word-to-predict*) pairs grouped by relative distance (d).

Previous work (Hermans and Schrauwen, 2013; Karpathy et al., 2016) studied this property of LSTMs by analyzing simple cases of closing brackets. By contrast RMN allows us to discover more interesting dependencies in the data. We manually inspect those high-frequency pairs to see whether they display certain linguistic phenomena. We observe that RMN captures, for example, *separable verbs* and *fixed expressions* in German. Separable verbs are frequent in German: they typically consist of preposition+verb constructions, such *ab+hängen* ('to depend') or *aus+schließen* ('to exclude'), and can be spelled together (*abhängen*) or apart as in '*hängen von der Situation ab*' ('depend on the situation'), depending on the grammatical construction. Figure 4.5a shows a long-dependency example for the separable verb *abhängen (to depend)*. When predicting the verb's particle *ab*, the model correctly attends to the verb's core *hängt* occurring seven words to the left. Figure 4.5b and 4.5c show fixed expression examples from German and Italian, respectively: *schlüsselrolle ... spielen (play a key role)* and *insignito ... titolo (awarded title)*. Here too, the model correctly attends to the key

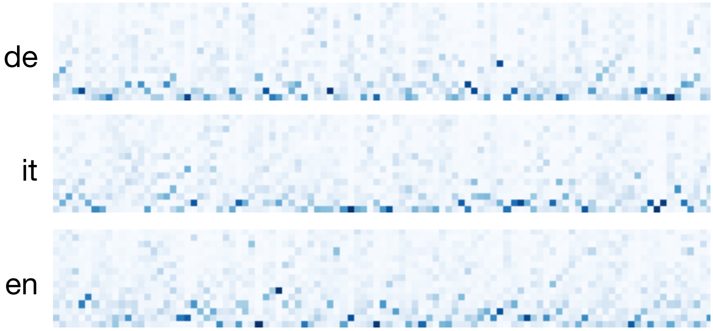
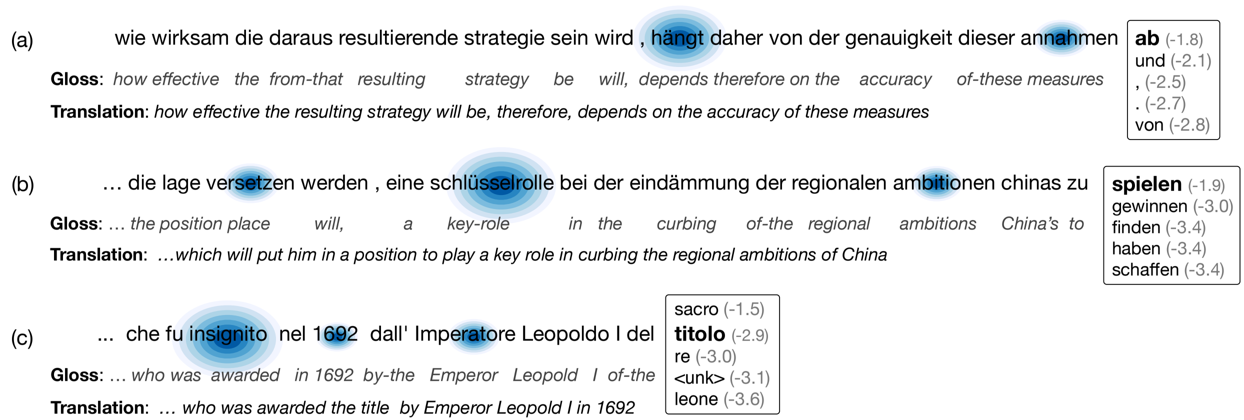


Figure 4.4: Attention visualization of 100 word samples. Bottom positions in each plot represent most recent history. Darker color means higher weight.



(a) wie wirksam die daraus resultierende strategie sein wird , hängt daher von der genauigkeit dieser annahmen

Gloss: *how effective the from-that resulting strategy be will, depends therefore on the accuracy of-these measures*

Translation: *how effective the resulting strategy will be, therefore, depends on the accuracy of these measures*

ab (-1.8)
und (-2.1)
, (-2.5)
. (-2.7)
von (-2.8)

(b) … die lage versetzen werden , eine schlüsselrolle bei der eindämmung der regionalen ambitionen chinas zu

Gloss: *… the position place will, a key-role in the curbing of-the regional ambitions China's to*

Translation: *…which will put him in a position to play a key role in curbing the regional ambitions of China*

spielen (-1.9)
gewinnen (-3.0)
finden (-3.4)
haben (-3.4)
schaffen (-3.4)

(c) … che fu insignito nel 1692 dall' Imperatore Leopoldo I del

Gloss: *… who was awarded in 1692 by-the Emperor Leopold I of-the*

Translation: *… who was awarded the title by Emperor Leopold I in 1692*

sacro (-1.5)
titolo (-2.9)
re (-3.0)
<unk> (-3.1)
leone (-3.6)

Figure 4.5: Examples of distant memory positions attended by RMN. The resulting top five word predictions are shown with the respective log-probabilities. The correct choice (in bold) was ranked first in sentences (a,b) and second in (c).

word despite its long distance from the word to predict.

Other interesting examples found by the RMN in the test data include:

German: findet *statt* (takes *place*), kehrte *zurück* (came *back*), fragen *antworten* (questions *answers*), kämpfen *gegen* (fight *against*), bleibt *erhalten* (remains *intact*), verantwortung *übernimmt* (*takes* responsibility);

Italian: sinistra *destra* (left *right*), latitudine *longitudine* (latitude *longitude*), collegata *tramite* (connected *through*), sposò *figli* (got-married *children*), insignito *titolo* (awarded *title*).

4.5.2 Syntactic Analysis

It has been conjectured that RNNs, and LSTMs in particular, model text so well because they capture syntactic structure implicitly. Unfortunately this has been hard to prove, but with our RMN model we can get closer to answering this important question.

We produce dependency parses for our test sets using the parser by Sennrich et al. (2013) for German and Attardi et al. (2009) for Italian. Next we look at how much attention mass is concentrated by the RM(+tM-g) model on different dependency types. Figure 4.6 shows for each language a selection of ten dependency types that are often long-distance.[1] The German and Italian tag sets are explained in Simi et al. (2014) and Foth (2006) respectively. Dependency direction is marked by an arrow: e.g. *→mod* means that the word to predict is a modifier of the attended word, while *mod←* means that the attended word is a modifier of the word to predict.[2] White cells denote combinations of position and dependency type that were not present in the test data.

While in most of the cases closest positions are attended the most, we can see that some dependency types also receive noticeably more attention than the average (ALL) on the long-distance positions. In German, this is mostly visible for the head of separable verb particles (*→avz*), which nicely supports our observations in the lexical analysis (Section §4.5.1). Other attended dependencies include: auxiliary verbs (*→aux*) when predicting the second element of a complex tense (*hat . . . gesagt / has said*); subordinating conjunctions (*konj←*) when predicting the clause-final inflected verb (*dass sie sagen sollten / that they should say*); control verbs (*→objі*) when predicting the infinitive verb (*versucht ihr zu helfen / tries to help her*). Out of the Italian dependency types selected for their frequent long-distance occurrences (bottom of Figure 4.6), the most attended are argument heads (*→arg*), complement heads (*→comp*), object heads (*→obj*) and subjects (*subj←*). This suggests that the RMN is mainly capturing predicate

[1] The full plots are available at https://github.com/ketranm/RMN

[2] Some dependency directions, like *obj←* in Italian, are almost never observed due to order constraints of the language.

argument structure in Italian. Notice that syntactic annotation is never used to train the model, but only to analyze its predictions.

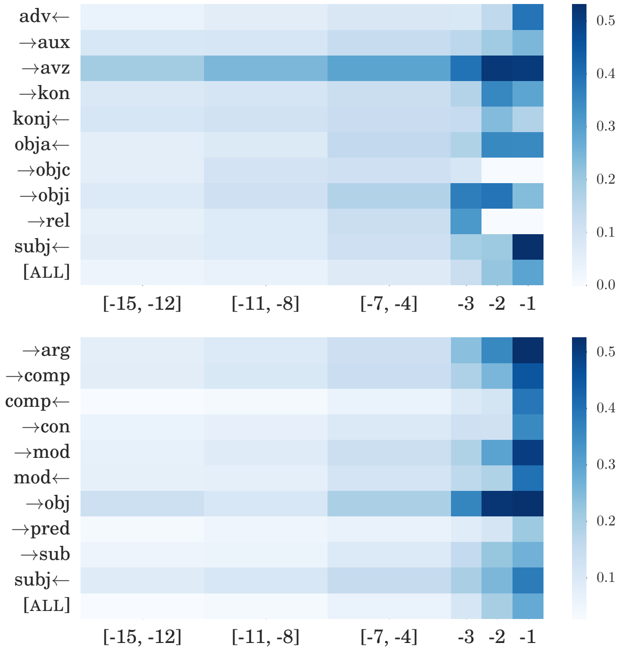


Figure 4.6: Average attention weights per position, broken down by dependency relation type+direction between the attended word and the word to predict. Top: German. Bottom: Italian. More distant positions are binned.

We can also use our RMN to discover which complex dependency paths are important for word prediction. To mention just a few examples, high attention on the German path *[subj←,→kon,→cj]* indicates that the model captures morphological agreement between coordinate clauses in non-trivial constructions of the kind: *spielen die Kinder im Garten und singen* / *the children play in the garden and sing*. In Italian, high attention on the path *[→obj,→comp,→prep]* denotes cases where the semantic relatedness between a verb and its object does not stop at the object's head, but percolates down to a prepositional phrase attached to it (*passò buona parte della sua vita* / *spent a large part of his life*). Interestingly, both local n-gram context and immediate dependency context would have missed these relations.

While much remains to be explored, our analysis shows that RMNs discover patterns far more complex than pairs of opening and closing brackets, and suggests that the network's hidden state captures to a large extent the underlying structure of text.

4.6 Sentence Completion Challenge

In addition to demonstrating that RMNs provide an interpretable interface for linguistic analysis, in this section, we evaluate the modeling power of RNNs on a challenging NLP task, namely Sentence Completion. The Microsoft Research Sentence Completion Challenge (Zweig and Burges, 2012) has recently become a test bed for advancing statistical language modeling. We choose this task to demonstrate the effectiveness of our RMN in capturing sentence coherence. The test set consists of 1,040 sentences selected from five Sherlock Holmes novels by Conan Doyle. For each sentence, a content word is removed and five candidates of the missing word are provided. The task is to identify the correct missing word among the five given candidates. The task is carefully designed to be non-solvable for local language models such as n-gram models. The best reported result is 58.9% accuracy (Mikolov et al., 2013)[3] which is far below human accuracy of 91% (Zweig and Burges, 2012).

As baseline we use a stacked three-layer LSTM. Our models are two variants of RM(+tM-g), each consisting of three LSTM layers followed by a MB. The first variant (unidirectional-RM) uses n words preceding the word to predict, the second (bidirectional-RM) uses the n words preceding *and* the n words following the word to predict, as MB input. We include bidirectional-RM in the experiments to show the flexibility of utilizing future context in RMNs.

We train all models on the standard training data of the challenge, which consists of 522 novels from Project Gutenberg, preprocessed similarly to (Mnih and Kavukcuoglu, 2013). After sentence splitting, tokenization and lowercasing, we randomly select 19,000 sentences for validation. The training and validation sets include 47M and 190K tokens respectively. The vocabulary size is about 64,000.

We initialize and train all networks as described in Section §4.4.2. Moreover, for regularization, we place dropout (Srivastava et al., 2014) after each LSTM layer as suggested in (Pham et al., 2014). The dropout rate is set to 0.3 in all the experiments.

Table 4.4 summarizes the results. It is worth mentioning that our LSTM baseline outperforms a dependency RNN making explicit use of syntactic information (Mirowski and Vlachos, 2015) and performs on par with the best published result (Mikolov et al., 2013). Our unidirectional-RM sets a new state of the art for the Sentence Completion Challenge with 69.2% accuracy. Under the same setting of d we observe that using bidirectional context does not bring additional advantages to the model. Mnih and Kavukcuoglu (2013) also report a similar observation. We believe that RMNs may achieve further improvements with hyper-parameter optimization.

[3] The authors use a weighted combination of skip-ngram and RNN without giving any technical details.

The stage lost a fine ____ , even as science lost an acute reasoner , when he became a specialist in crime
a) linguist b) hunter c) **actor**♣ d) estate e) horseman◇

What passion of hatred can it be which leads a man to ____ in such a place at such a time
a) **lurk**♣ b) dine◇ c) luxuriate d) grow e) wiggle

In my inmost heart I believed that I could ____ where others failed , and now I had the opportunity to test myself
a) smell b) **succeed**♣ c) lie d) spell e) forget◇

My heart is ____ already since i have confided my trouble to you
a) falling b) distressed◇ c) soaring d) **lightened**♣ e) punished

My morning's work has not been ____ , since it has proved that he has the very strongest motives for standing in the way of anything of the sort
a) invisible b) neglected◇♣ c) overlooked d) **wasted** e) deliberate

That is his ____ fault , but on the whole he's a good worker
a) **main** b) successful c) mother's♣ d) generous e) favourite◇

Figure 4.7: Examples of sentence completion. The correct option is in boldface. Predictions by the LSTM baseline and by our best RMN model are marked by ◇ and ♣ respectively.

Figure 4.7 shows some examples where our best RMN beats the already very competitive LSTM baseline, or where both models fail. We can see that in some sentences the necessary clues to predict the correct word occur only to its *right*. While this seems to conflict with the worse result obtained by the bidirectional-RM, it is important to realize that prediction corresponds to the whole sentence probability. Therefore a badly chosen word can have a negative effect on the score of future words. This appears to be particularly true for the RMN due to its ability to directly access (distant) words in the history. The better performance of a unidirectional versus a bidirectional-RM may indicate that the attention in the memory block can be distributed reliably only over words that have been already seen and summarized by the current LSTM state. In future work, we may investigate whether different ways to combine two RMNs running in opposite directions further improve accuracy on this challenging task.

4.7 Limitations

While our main goal is to understand the source of RNNs' impressive performance from a linguistic perspective, from a computational point of view, we note that our proposed RMNs require a significant amount of additional computation during training because they have a much larger number of parameters in comparison to LSTMs. However, it is possible to reduce the number

Model	n	d	**Accuracy**
LSTM	–	256	56.0
unidirectional-RM	15	256	64.3
	15	512	**69.2**
bidirectional-RM	7	256	59.6
	10	512	67.0

Table 4.4: Accuracy on 1,040 test sentences. We use perplexity to choose the best model. Dimension of word embeddings, LSTM hidden states, and gate g parameters are set to d.

of parameters in RMNs without sacrificing their interpretability. A straightforward solution, for example, is to share input/output memory embeddings with word embeddings.

4.8 Conclusions

In this chapter, we have answered **Research Question 2**: *From a linguistic perspective, what makes recurrent neural networks work so well for language modeling?* In order to answer this research question, we have broken it down into three sub-questions and provided an answer to each of those. In the following, we revisit the three sub-questions posed at the beginning of the chapter and highlight the main contributions and findings in addressing each individual sub-question.

RQ2.1 *Can recurrent neural networks (RNNs) be made more interpretable?*

We showed that recurrent neural networks can be made more interpretable by equipping them with an attention-based memory addressing mechanism. The Recurrent Memory Networks (RMNs) proposed in this chapter are an implementation of this idea. The Memory Blocks in RMNs provide an informative interface to investigate the internal behavior of the models. By analyzing attention distributions within Memory Blocks, we can identify important lexical features for predicting the next words and inspect them from a linguistic perspective.

RQ2.2 *What linguistic phenomena captured by RNNs make them so successful in modeling language?*

We performed the analysis of the attention weights in RMNs when they predict the next word. We found that RMNs learn important co-occurrences regardless of their distance. Even more interestingly, our RMNs implicitly capture certain dependency types that are important for word prediction, despite being trained without any syntactic information.

RQ2.3 *Do RMNs offer any extra modeling power in addition to their interpretability?*

We evaluated RMNs on the Sentence Completion Challenge dataset. We showed that RMNs obtain excellent performance at modeling sentence coherence, setting a new state of the art on the challenging sentence completion task.

Chapter 5

The Importance of Being Recurrent for Modeling Hierarchical Structure

5.1 Introduction and Research Questions

In the previous chapter, we have demonstrated that recurrent neural networks, in particular Long Short-Term Memory networks (LSTMs), can be made more powerful with augmented memory. Additionally, by analyzing augmented memory, we have shown that RNNs can capture some important syntactic dependencies without being provided with any syntactic annotations. Recent work corroborates our findings that LSTMs can implicitly encode and exploit hierarchical information when trained to solve common natural language processing tasks such as language modeling (Linzen et al., 2016) and neural machine translation (Shi et al., 2016). While LSTMs appear to be a natural choice for modeling sequential data, recently a class of non-recurrent models (Gehring et al., 2017; Vaswani et al., 2017) have shown competitive performance on sequence modeling. Gehring et al. (2017) propose a fully convolutional sequence-to-sequence model that achieves state-of-the-art performance in machine translation. Vaswani et al. (2017) introduce Transformer networks that do not use any convolution or recurrent connections while obtaining the best translation performance. These non-recurrent models are appealing due to their highly parallelizable computations on modern GPUs. But do they have the same ability to exploit hierarchical structures *implicitly* in comparison to RNNs? In this chapter, we compare the two architectures—*recurrent* versus *non-recurrent*—with respect to their ability to model hierarchical structure. We ask the following research question

Research question 3 *Do non-recurrent neural networks have the same ability to exploit hierarchical structures implicitly in comparison to their recurrent counterpart?*

Our interest here is the ability of capturing hierarchical structure without being equipped with explicit structural representations (Bowman et al., 2015b; Linzen et al., 2016). We choose Transformer as a non-recurrent model to study in this paper. We refer to Transformer as Fully Attention Network (FANs) to emphasize this characteristic. Our motivation to favor FANs over convolutional neural networks (CNNs) is that FANs always have full access to the sequence history, making them more suited for modeling long distance dependencies than CNNs. Additionally, FANs promise to be more interpretable than LSTMs by visualizing attention weights.

In order to evaluate the learned representations of both models, it is important to chose appropriate tasks wherein reasoning about hierarchical structure is the key to achieve good performance. We study two tasks in this chapter, each of which corresponds to a sub-question:

RQ3.1 *Do FANs have the same ability as LSTMs to exploit hierarchical structure in natural language data?*

RQ3.2 *Do FANs have the same ability as LSTMs to encode artificial tree-structured data?*

The rest of the chapter is organized as follows: We first describe the mechanics of FAN (§ 5.2). We then highlight the differences between the two architectures (§5.3) and introduce the two tasks (§5.4). Then we provide setup and results for each task (§5.5 and §5.6) and discuss our findings (§5.7).

5.2 Fully Attention Network

FANs consist of several sub-layers stacked on top of each other. Figure 5.1 illustrates the computational graph of a sub-layer in FANs. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be sequence of tokens x_i. For each token x_i, let $\boldsymbol{x}_i \in \mathbb{R}^d$ be its embedding. We denote the embedding matrix of sentence $\boldsymbol{x}$ by $\boldsymbol{X} = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_n] \in R^{n \times d}$. Given an input $\boldsymbol{X}$, a sub-layer L transforms $\boldsymbol{x}$ to output $\boldsymbol{Y} \in \mathbb{R}^{n \times d}$ using a series of transformations

$$\begin{aligned} \boldsymbol{Z} &= \text{MultiHead}(\boldsymbol{X}) && (5.1)\\ \boldsymbol{H}_0 &= \text{LayerNormalization}(\boldsymbol{X} + \boldsymbol{Z}) && (5.2)\\ \boldsymbol{H}_1 &= \text{FeedForward}(\boldsymbol{H}_0) + \boldsymbol{H}_0 && (5.3)\\ \boldsymbol{Y} &= \text{LayerNormalization}(\boldsymbol{H}_1) && (5.4) \end{aligned}$$

The MultiHead introduced by Vaswani et al. (2017) performs the following computations:

$$\begin{aligned}
\text{MultiHead}(\boldsymbol{x}) &= [\boldsymbol{Z}_1; \boldsymbol{Z}_2; \ldots; \boldsymbol{Z}_h]\boldsymbol{W}_o && (5.5)\\
\boldsymbol{Z}_i &= \text{softmax}(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^\top}{\sqrt{d_h}})\boldsymbol{V}_i && (5.6)\\
\boldsymbol{Q}_i &= \boldsymbol{X}\boldsymbol{W}_i^q \qquad \boldsymbol{W}_i^q \in \mathbb{R}^{d \times d_h} && (5.7)\\
\boldsymbol{K}_i &= \boldsymbol{X}\boldsymbol{W}_i^k \qquad \boldsymbol{W}_i^k \in \mathbb{R}^{d \times d_h} && (5.8)\\
\boldsymbol{V}_i &= \boldsymbol{X}\boldsymbol{W}_i^v \qquad \boldsymbol{W}_i^v \in \mathbb{R}^{d \times d_h} && (5.9)
\end{aligned}$$

where $\boldsymbol{W}_i^q$, $\boldsymbol{W}_i^q$, $\boldsymbol{W}_i^q$ and $\boldsymbol{W}_o \in \mathbb{R}^{hd_h \times d}$ are trainable parameters. The dimension d_h is chosen such that $d_h = d/h$.

Each matrix $\boldsymbol{Z}_i$ is a self-attention head that captures dependencies amongst the input words x_i. Having multiple heads can be useful to encode different dependencies within the input. For instance, one head focuses on capturing co-occurrence while another head specializes in detecting morphological agreement. This is a simplified interpretation of the role of each head for the purpose of understanding the model architecture.

The Layer Normalization is a technique to reduce training time of deep neural network by normalizing activities of the hidden units. The details of Layer Normalization are given in Ba et al. (2016b). The FeedForward layer is given by

$$\text{FeedForward}(\boldsymbol{x}) = \max(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \qquad (5.10)$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d_f}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d_f}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_f \times d}$, and $\boldsymbol{b}_2 \in \mathbb{R}^d$ are trainable parameters. In our experiments, we set $d_f = 2d$.

In order to model word order, FANs use additional positional embeddings to encode the relative position of words in the sentence. A positional embedding $\boldsymbol{e}_p[t] \in R^d$ at index t (corresponding to the t-th token) is given as

$$\begin{aligned}
\boldsymbol{e}_p[t][2i] &= \sin(\frac{t}{10000^{2i/d}}) && (5.11)\\
\boldsymbol{e}_p[t][2i+1] &= \cos(\frac{t}{10000^{2i/d}}) && (5.12)
\end{aligned}$$

Each dimension of $\boldsymbol{e}_p[t]$ corresponds to a sinusoid. Using sinusoids for positional embeddings have several advantages. First, it allows the model to generalize for longer sequences. Second, for a fixed offset k, $\boldsymbol{e}_p[t+k]$ is a linear function of $\boldsymbol{e}_p[t]$. This property makes it easy for the model to learn positional dependencies in the input.

For the mechanics of recurrent neural networks, we refer to Chapter 4 in this thesis. Next, we highlight the main differences between FANs and RNNs.

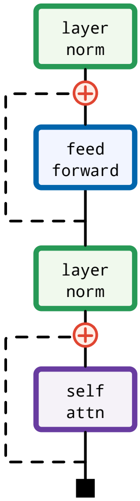


Figure 5.1: Sub-layer of FAN. The computation flows from bottom to top. Dashed lines indicate skip connection.

5.3 FANs versus LSTMs

Conceptually, FANs differ from LSTMs in the way they utilize the previous input to predict the next output. Figure 5.2 depicts the main difference in terms of computation when each model is making predictions. At time step t, a FAN can access information from all previous time steps *directly* with $\mathcal{O}(1)$ computational operations. FANs do so by employing a self-attention mechanism to compute the weighted average of all previous input representations. In contrast, LSTMs compress at each time step all previous information into a single vector *recursively* based on the current input and the previous compressed vector. By its definition, LSTMs require $\mathcal{O}(d)$ computational operations to access the information at time step $t-d$.

We now proceed to measure both models' ability to learn hierarchical structure with a set of controlled experiments.

5.4 Tasks

We choose two tasks to study in this work:

1. subject-verb agreement
2. logical inference

The first task was proposed by Linzen et al. (2016) to test the ability of recurrent neural networks to capture syntactic dependencies in natural language. We

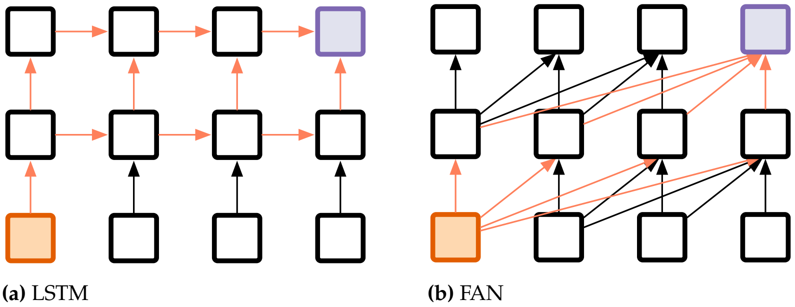


Figure 5.2: Diagram of the main difference between an LSTM and a FAN. The purple box indicates the summarized vector at current time step t which is used to make prediction. Orange arrows indicate the information flow from a previous input (orange box) to that vector.

choose this task to study **RQ3.1**. The input to the neural models is a sequence of words without any additional annotations of syntactic dependency. In order to solve the task, the model must learn the latent dependency between verb and its subject in a sentence.

The second task was introduced by Bowman et al. (2015b) to compare tree-based recursive neural networks against sequence-based recurrent networks with respect to their ability to exploit hierarchical structures. We choose this task to study **RQ3.2**. The input to the models is a string representation of a binary tree. In order to solve that task, the model must utilize the underlying tree-structured representation of the input.

The choice of tasks here is important to ensure that both models have to exploit hierarchical structural features instead of shallow ones (Jia and Liang, 2017).

5.5 Subject-Verb Agreement

Linzen et al. (2016) propose the task of predicting number agreement between subject and verb in naturally occurring English sentences as a proxy for the ability of LSTMs to capture hierarchical structure in natural language. We use the dataset provided by Linzen et al. (2016) and follow their experimental protocol of training each model using either (a) a general language model, *i.e.*, next word prediction objective, and (b) an explicit supervision objective, *i.e.*, predicting the number of the verb given its sentence history. Table 5.1 illustrates the training and testing conditions of the task.

A challenge for subject-verb agreement task is the possibility of agreement attraction errors (Bock and Miller, 1991) because dependencies are not explicitly provided to the model. The following examples illustrate the challenge posed to both FANs and LSTMs where there are intervening nouns (underlined) be-

Table 5.1: Examples of training and test conditions for the two subject-verb agreement subtasks. The full input sentence is "The **keys** to the cabinet **are** on the table" where verb and subject are bold and agreement attractor words are underlined.

	Input	Train	Test
(a)	the keys to the cabinet	are	$p(\text{are}) > p(\text{is})$?
(b)	the keys to the cabinet	plural	plural/singular?

tween the subject (**bold**) and the verb (**bold**). These intervening nouns have the opposite number as the subject and are referred to as *agreement attractors*:

(a) The only championship **banners** that are currently displayed within the building **are** for national or NCAA Championships.

(b) Yet the **ratio** of men who survive to the women and children who survive **is** not clear in this story.

5.5.1 Data

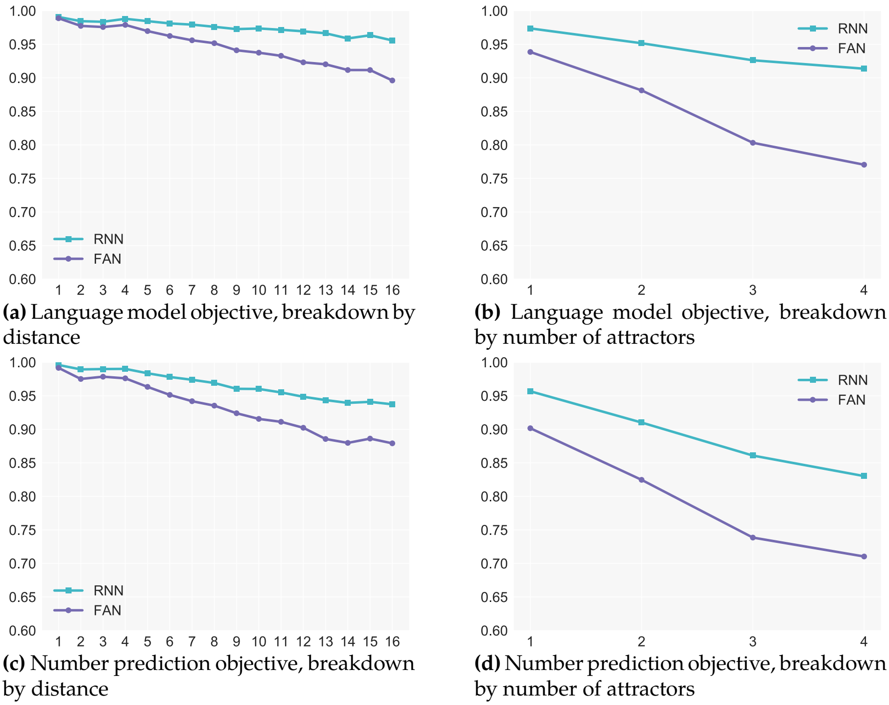
Following the original setting in (Linzen et al., 2016), we take 10% of the data for training, 1% for validation, and the rest for testing. The vocabulary consists of the 10k most frequent words, while the remaining words are replaced by their part-of-speech tag.

5.5.2 Hyper-parameters

In this experiment, both the LSTM and the FAN have 4 layers, the dropout rate is 0.2, and word-embeddings and hidden sizes are set to 128. The weights of the word embeddings and output layer are shared as suggested by Inan et al. (2016); Press and Wolf (2017). The FAN has two attention heads. LSTMs are trained with the Adam optimizer with a learning rate of 0.001. The FAN is trained with Adam for the language model objective and the YellowFin optimizer (Zhang et al., 2017)[1] for the number prediction objective. The initial learning rate is set to 0.001.

We first assess whether the LSTM and FAN models trained with respect to the language model objective assign higher probabilities to the correctly inflected verbs. As shown by Figure 5.3a and Figure 5.3b, both models achieve high accuracies for this task, but LSTMs consistently outperform FANs. Moreover, LSTMs are clearly more robust than FANs with respect to task difficulty, measured both in terms of word distance and number of agreement attractors between subject and verb.

[1] We found that YellowFin gives better results than Adam for FANs.



(a) Language model objective, breakdown by distance

(b) Language model objective, breakdown by number of attractors

(c) Number prediction objective, breakdown by distance

(d) Number prediction objective, breakdown by number of attractors

Figure 5.3: Results of subject-verb agreement with different training objectives.

Interestingly, previous studies (Christiansen and Chater, 2016; Cornish et al., 2017) have argued that human memory limitations give rise to important characteristics of natural language, including its hierarchical structure. Similarly, our experiments suggest that, by compressing the history into a single vector before making predictions, LSTMs are forced to better learn the input structure. On the other hand, despite having direct access to all words in their history, FANs are less capable of detecting the verb's subject. We note that the validation perplexities of the LSTM and FAN are 75.17 and 71.39, respectively. The lack of correlation between perplexity and agreement accuracy indicates that FANs might capture other aspects of language better than LSTMs. We leave this question to future work.

Second, we evaluate FAN and LSTM models explicitly trained to predict the verb number (Figure 5.3c and Figure 5.3d). Again, we observe that LSTMs consistently outperform FANs. This is a particularly interesting result since the self-attention mechanism in FANs connects two words in any position with a $\mathcal{O}(1)$ number of executed operations, whereas RNNs require more recurrent operations. Despite this apparent advantage of FANs, the performance gap between FANs and LSTMs increases with the distance and number of attractors. We note that our LSTM results are better than those in Linzen et al. (2016). Also surprising is that the language model objective yields higher accuracies than the number prediction objective. We believe this may be due to better model optimization and to the embedding-output layer weight sharing, but we leave a thorough investigation to future work.

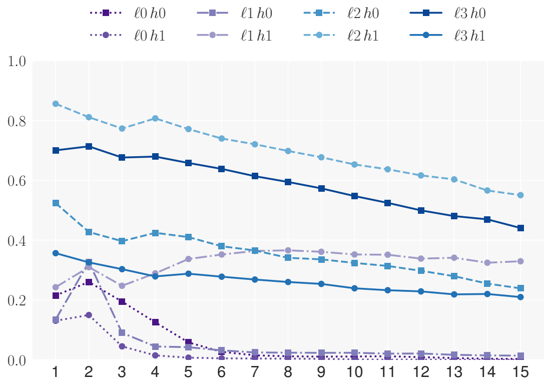


Figure 5.4: Proportion of times the subject is the most attended word by different heads at different layers ($\ell3$ is the highest layer). Only cases where the model made a correct prediction are shown.

To gain further insights into our results, we examine the attention weights computed by FANs during verb-number prediction (supervised objective). Specifically, for each attention head at each layer of the FAN, we compute the percentage of times the subject is the most attended word among all words in the history. Figure 5.4 shows the results for all cases where the model made the correct prediction. While it is hard to interpret the exact role of attention for different heads and at different layers, we find that some of the attention heads at the higher layers ($\ell2$ $h1$, $\ell3$ $h0$) frequently point to the subject with an accuracy that decreases linearly with the distance between subject and verb.

5.6 Logical Inference

In this task, we choose the artificial language introduced by Bowman et al. (2015b). This language has six word types *a*, *b*, *c*, *d*, *e*, *f* and three logical operations *or*, *and*, *not*. There are seven mutually exclusive logical relations that describe the relationship between two sentences: entailment ($\sqsubset$, $\sqsupset$), equivalence ($\equiv$), exhaustive and non-exhaustive contradiction ($\wedge$, $|$), and two types of semantic independence ($\#$, $\smile$). We generate 60,000 samples[2] with the number of logical operations ranging from 1 to 12. The train/dev/test dataset ratios are set to 0.8/0.1/0.1. In the following, we show some samples of the training data.

[2] https://github.com/sleepinyourhat/vecrtor-entailment

(d (or f))	$\sqsupset$	(f (and a))
(d (and (c (or d))))	#	(not f)
(not (d (or (f (or c)))))	$\sqsubset$	(not (c (and (not d))))
((not (c (or a))) (and e))	$\mid$	((((not d) (and e)) (and a)) (or c))

5.6.1 Why artificial data?

Despite the simplicity of the language, this task is not trivial. To correctly classify logical relations, the model must learn nested structures as well as the scope of logical operations. We verify the difficulty of the task by training three bag-of-words models followed by sum/average/max-pooling. The best of the three models achieve less than 59% accuracy on the logical inference versus 77% on the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015a). This shows that the SNLI task can be largely solved by exploiting shallow features without understanding the underlying linguistic structures.

5.6.2 Models

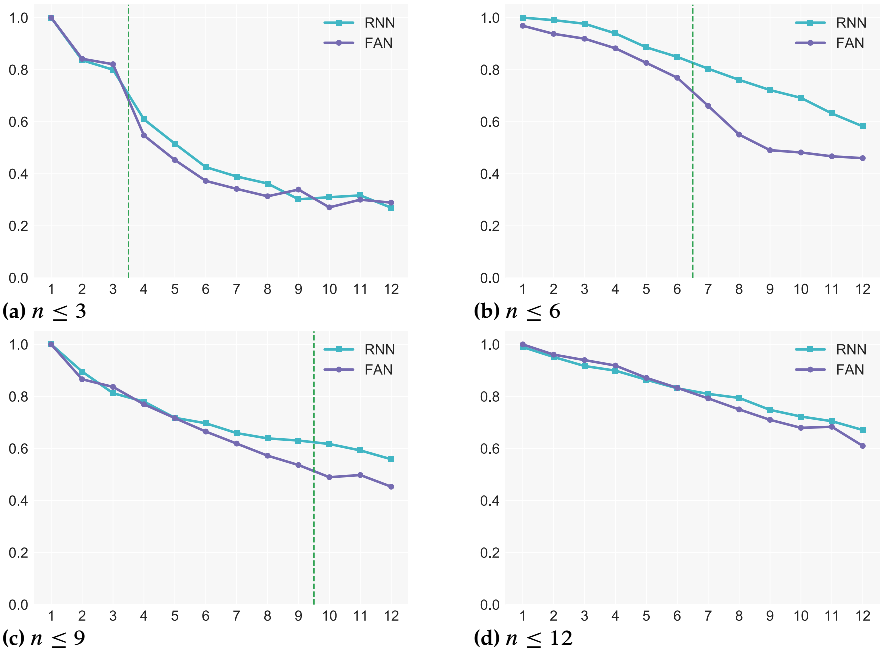
We follow the general architecture proposed in Bowman et al. (2015b): Premise and hypothesis sentences are encoded by fixed-size vectors. These two vectors are then concatenated and fed to a 3-layer feed-forward neural network with ReLU nonlinearities to perform 7-way classification.

The LSTM architecture used in this experiment is similar to that of Bowman et al. (2015b). We simply take the last hidden state of the top LSTM layer as a fixed-size vector representation of the sentence. Here we use a 2-layer LSTM with skip connections. The FAN maps a sentence x of length n to a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1; \ldots ; \boldsymbol{y}_n] \in \mathbb{R}^{n \times d}$. To obtain a fixed-size representation $\boldsymbol{z}$, we use a self-attention layer with two trainable queries $\boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathbb{R}^{1 \times d}$:

$$\begin{aligned} \boldsymbol{z}_i &= \operatorname{softmax}\left(\frac{\boldsymbol{q}_i \boldsymbol{Y}^\top}{\sqrt{d}}\right)\boldsymbol{Y} \quad i \in \{1, 2\} \\ \boldsymbol{z} &= [\boldsymbol{z}_1, \boldsymbol{z}_2] \end{aligned}$$

5.6.3 Results

Following the experimental protocol of (Bowman et al., 2015b), the data is divided into 13 bins based on the number of logical operations. Both FANs and LSTMs are trained on samples with at most n logical operations and tested on all bins. Figure 5.5 shows the result of the experiments with $n \in \{3, 6, 9, 12\}$. We see that FANs and LSTMs perform similarly when trained on the whole dataset (Figure 5.5d). However when trained on a subset of the data (Figure 5.5b), LSTMs obtain better accuracies on similar examples ($n \leq 6$) and LSTMs generalize better on longer examples ($6 < n \leq 12$).



(a) $n \leq 3$ **(b)** $n \leq 6$ **(c)** $n \leq 9$ **(d)** $n \leq 12$

Figure 5.5: Results of logical inference

5.7 Conclusions

Using two controlled experiments, we have compared a recurrent architecture (LSTM) to a non-recurrent one (FAN) with respect to the ability of capturing hierarchical structure. Our findings provide an empirical answer to **Research question 3**:

RQ3 *Do non-recurrent neural networks have the same ability to exploit hierarchical structures implicitly in comparison to their recurrent counterpart?*

Our experiments show that LSTMs slightly but consistently outperform FANs. Specifically, with the first experiment we have answered:

RQ3.1 *Do FANs have the same ability as LSTMs to exploit hierarchical structure in natural language data?*

We evaluated both LSTMs and FANs on a subject-verb agreement task. Both types of models performed well overall. However, we found that LSTMs are notably more robust with respect to the presence of misleading features, whether trained with explicit supervision or with a general language model objective.

With the second experiment we have answered:

RQ3.2 *Do FANs have the same ability as LSTMs to encode artificial tree-structured data?*

We evaluated two types of models on a logical inference task, where the input is given in the form of bracketings. Our results suggest that both LSTMs and FANs can exploit the structure of the input as they both perform better than a bag-of-words baseline. However, we found that LSTMs generalize better to longer sequences for the logical inference task.

These findings suggest that recurrency is a key model property which should not be sacrificed for efficiency when hierarchical structure matters for the task.

This does not imply that LSTMs should always be preferred over non-recurrent architectures. In fact, both FAN- and CNN-based networks have proved to perform comparably to or better than LSTM-based ones on a very complex task like machine translation (Gehring et al., 2017; Vaswani et al., 2017). Nevertheless, we believe that the ability of capturing hierarchical information in sequential data remains a fundamental need for building intelligent systems that can understand and process language. Thus we hope that our insights will be useful towards building the next generation of neural networks.

Chapter 6

Unsupervised Neural Hidden Markov Models

6.1 Introduction and Research Questions

In Chapter 4 and Chapter 5, we have shown that recurrent neural network LMs can implicitly capture shallow syntactic dependency types that are important for word prediction, despite being trained without any syntactic information. Nevertheless, the word prediction objective is a supervised objective and neural networks are known for their ability to exploit hierarchical structure in supervised learning (Linzen et al., 2016; Shi et al., 2016; Belinkov et al., 2017). In this chapter, we explore the representational power of neural networks in a more challenging setup, namely unsupervised linguistic structure prediction.

There are two important motivations for unsupervised linguistic structure prediction: First, it plays a key role in many NLP applications such as topic modeling (Blei et al., 2003) in information retrieval and word alignment in phrase-based machine translation (Koehn et al., 2003). Second, while unsupervised linguistic structure has been studied exhaustively in many aspect of computational linguistics such as part-of-speech induction (Christodoulopoulos et al., 2010) and grammar induction (Klein and Manning, 2004), we are interested in these unsupervised models that are parametrized by neural networks and that can be trained in an end-to-end fashion. With the above motivations, in this chapter we ask:

Research question 4: *Can neural networks be used to induce linguistic structure in a completely unsupervised manner?*

We will answer this research question using probabilistic graphical models. Probabilistic graphical models are among the most important tools available to the NLP community. In particular, the ability to train generative models using Expectation-Maximization (EM), Variational Inference (VI), and sampling methods like Markov chain Monte Carlo (MCMC) has enabled the development of unsupervised systems for tag and grammar induction, alignment, topic models and more. These latent variable models discover hidden structure in text which aligns to known linguistic phenomena and whose clusters are easily identifiable.

Recently, much of supervised NLP has found great success by augmenting or replacing context, features, and word representations with embeddings derived from deep neural networks. These models allow for learning highly expressive non-convex functions by simply back-propagating prediction errors. Inspired by Berg-Kirkpatrick et al. (2010), who bridged the gap between supervised and unsupervised training with features, we bring neural networks to unsupervised learning by providing evidence that even in unsupervised settings, simple neural network models trained to maximize the marginal likelihood can outperform more complicated models that use expensive inference.

We investigate **Research Question 3** by studying a simple directed graphical model, namely the Hidden Markov Model. Concretely, we ask three following sub-questions:

RQ4.1 *Can an unsupervised Hidden Markov Model be fully expressed as a neural network model?*

In this work, we show how a single latent variable sequence model, Hidden Markov Models (HMMs), can be implemented with neural networks by simply optimizing the incomplete data likelihood. The key insight is to perform standard forward-backward inference to compute posteriors of latent variables and then back-propagate the posteriors through the networks to maximize the likelihood of the data.

RQ4.2 *What are the advantages of the neural methods in comparison to traditional Bayesian methods?*

Using features in unsupervised learning has been a fruitful enterprise (Das and Petrov, 2011; Berg-Kirkpatrick and Klein, 2010; Cohen et al., 2011) and attempts to combine HMMs and Neural Networks date back to 1991 (Bengio et al., 1991). Additionally, similarity metrics derived from word embeddings have also been shown to improve unsupervised word alignment (Songyot and Chiang, 2014). Our focus in this work is to present a generative neural approach to HMMs and demonstrate how this framework lends itself to modularity (*e.g.*, the easy inclusion of morphological information via Convolutional Neural Networks §6.6), and the addition of extra conditioning context (*e.g.*, using an RNN to model the sentence §6.7). Our approach will be demonstrated and evaluated on the simple task of part-of-speech tag induction.

RQ4.3 *Are neural models sensitive to parameter initialization in the same way as non-neural models are?*

> It is well known that optimizing a non-convex unsupervised loss is problematic (Dauphin et al., 2014). In the literature of part-of-speech induction, researchers often report the average the results over many runs with different parameter initializations (Clark, 2003; Goldwater et al., 2006; Johnson, 2007). In section §6.9, we will explore the effect of parameter initialization and neural network hyper-parameters on the quality of the induced tags.

Interest in the interface of graphical models and neural networks has grown recently as new inference procedures have been proposed (Johnson et al., 2016). Common to this work and ours is the use of neural networks to produce potentials. The approach presented here is easily applied to other latent variable models where inference is tractable and which are typically trained with EM. We believe that the important strength of using a neural network to produce model probabilities is that it allows for seamless integration of additional context not easily represented by conditioning variables in a traditional model.

The rest of the chapter is organized as follow: We briefly provide a relevant background of graphical models in Section §6.2. We then present our framework in Section §6.3. Next, we introduce the task of part-of-speech induction and the derivation of Baum-Welch in Section §6.4. We provide the details of neural implementation of our framework (§6.5) and the extension (§6.6 and §6.7). Three evaluation metrics are presented in Section §6.8. The details of data and parameters are given in Section §6.9. We then present our experimental results and parameter ablation in Section §6.10. We conclude this chapter in Section §6.12.

6.2 Graphical Models

Graphical models are widely used in natural language processing such as hidden Markov models (HMM), conditional random fields (CRF) or latent Dirichlet allocation (LDA) models. In the context of unsupervised learning, we will work with *directed (probabilistic) graphical models*. Directed graphical models are a type of probabilistic model where all variables are structured into a directed acyclic graph. Let $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ be the set of random variables in a directed graphical model. The joint distribution over random variables $\boldsymbol{x}$ has the following factorization:

$$p_\theta(x_1, \ldots, x_N) = \prod_{i=1}^{N} p_\theta\left(x_i \mid pa(x_i)\right) \tag{6.1}$$

where $pa(x_i)$ denotes the set of parent variables of node x_i in the directed graph and θ are the parameters of the models.

The joint distribution $p_\theta(\boldsymbol{x})$ can also be expressed as a product of factors over

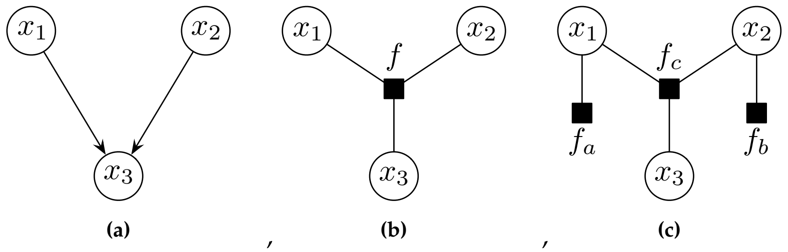


Figure 6.1: (a) A directed graph with factorization $p(x_1)p(x_2)p(x_3 \mid x_1, x_2)$. (b) A factor graph represents the same joint distribution with factor $f = p(x_1)p(x_2)p(x_3 \mid x_1, x_2)$. (c) A different factor graph with three factors $f_a = p(x_1)$, $f_b = p(x_2)$, and $f_c = p(x_3 \mid x_1, x_2)$.

subsets of variables $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = \prod_s f_s(\boldsymbol{x}_s) \tag{6.2}$$

where $\boldsymbol{x}_s$ denotes a subset of the variables and each factor f_s is a function of the set $\boldsymbol{x}_s$ This expression introduces new factor nodes to the graph in addition to variable nodes (Figure 6.1). The new graph construction is called a *factor graph*. The advantage of using factor graphs is that they make the details of computation more explicitly.

Next we will describe the sum-product algorithm—a powerful class of efficient, exact inference algorithms—that work over tree-structured graphs (Bishop, 2006). The sum-product algorithm is required in our framework (§6.3) to calculate the gradients of the likelihood. We note that our proposed framework can be applied to any type of directed graphical models consisting of *discrete* variables, which are a common interest in the field of NLP.

6.2.1 The Sum-product Algorithm

We evaluate the marginal $p(x)$ of a particular node x in the graph. This problem arises quite often in NLP contexts where we wish to evaluate how likely a word x_i is a verb in sentence $\boldsymbol{x}$. We marginalize out all the variables in $\boldsymbol{x}$ except x:

$$p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x}) \tag{6.3}$$

The summation in equation (6.3) is expensive to compute. The idea of sum-product algorithm is to interchange the summations and the products to obtain an efficient algorithm. First, we assume that the joint probability $p(\boldsymbol{x})$ can be factorized into a product over disjoint sets X_s

$$p(\boldsymbol{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) \tag{6.4}$$

where $\mathrm{ne}(x)$ denotes a set of factor nodes that are neighbor of x and X_s denotes the set of all variables in the subtree that connects to variable x via factor node f_s. Figure (6.2) illustrates this factorization. The marginal $p(x)$ is rewritten by

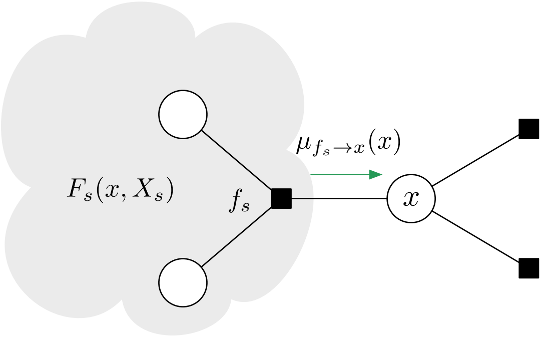


Figure 6.2: A fragment of a factor graph illustrates a message from a factor node f_s to a variable node x.

interchanging the sums and products:

$$p(x) = \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] \tag{6.5}$$

$$= \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \tag{6.6}$$

where $\mu_{f_s \to x}(x)$ is a message sent from factor f_s to node x, and is defined as

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s) \tag{6.7}$$

Assuming that the factor f_s has $M + 1$ neighbor nodes $x, x_1, \ldots, x_M$. We can write $F_s(x, X_s)$ as

$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) G_1(x_1, X_{s1}) \ldots G_m(x_m, X_{sm}) \tag{6.8}$$

Pluging Equation (6.8) to (6.7) we get

$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \backslash x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right]$$

$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \backslash x} \mu_{x_m \to f_s}(x_m) \tag{6.9}$$

where $\mu_{x_m \to f_s}(x_m)$ is a message sent from node x_m to factor $f_s(x_m)$, and is defined as

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{6.10}$$

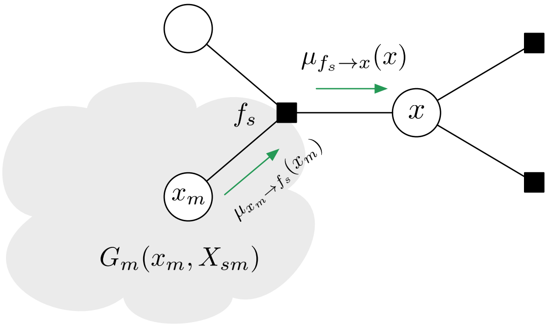


Figure 6.3: A fragment of a factor graph illustrates sent messages from all variable nodes associated with a factor node f_s.

Based on the factorization of the probability and the structure of the graph, from Figure (6.3), we see that

$$G_m(x_m, X_{sm}) \equiv \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml}) \tag{6.11}$$

Pluging Equation (6.11) into (6.10) and interchange between summations and products we arrive

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m, X_{ml}) \right] \tag{6.12}$$

$$= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m) \tag{6.13}$$

Equation (6.9) and (6.13) describe the relationships between two types of messages: messages from factor to node $\mu_{f_s \to x}(x)$ and messages from node to factor $\mu_{x_m \to f_s}(x_m)$. Finally, we define two special messages sending from leaf nodes in Figure (6.4)

$$\mu_{x \to f}(x) = 1 \tag{6.14}$$

$$\mu_{f \to x}(x) = f(x) \tag{6.15}$$

$\mu_{x \to f}(x) = 1$
x
f
$\mu_{f \to x}(x) = f(x)$
x
f

(a) Message from node to factor.

(b) Message from factor to node.

Figure 6.4: Special messages from leaf nodes.

In some applications, we wish to evaluate the marginal $p(\boldsymbol{x}_s)$ over a set of variables $\boldsymbol{x}_s$ in a factor $f_s(\boldsymbol{x}_s)$. For instance, we would like to know how likely a word x_j is as a parent of word x_i in sentence $\boldsymbol{x}$ under all possible valid dependency structures over the sentence. In this case, we still can reuse the computations of the sum-product algorithm effectively. Let $\boldsymbol{x}_s = \{x_1, \ldots, x_M\}$ be the set of M variables associated with factor $f_s(\boldsymbol{x}_s)$. The joint distribution $p(\boldsymbol{x})$ can be rewritten as

$$p(\boldsymbol{x}) = f_s(\boldsymbol{x}_s) \prod_{i=1}^{M} \prod_{t \in \mathrm{ne}(x_i)} F_t(x_i, X_t) \tag{6.16}$$

where $\mathrm{ne}(x_i)$ denotes the set of factor nodes that are neighbors of x_i except f_s, X_t denotes the set of all variables in the subtree connected to the variable node x_i via the factor node f_t, and $F_t(x_i, X_t)$ represents the product of all the factors in the group associated with factor f_t. The marginal $p(\boldsymbol{x}_s)$ is obtained by summing over all $\boldsymbol{x} \setminus \boldsymbol{x}_s$:

$$\begin{aligned} p(\boldsymbol{x}_s) &= \sum_{\boldsymbol{x} \setminus \boldsymbol{x}_s} p(\boldsymbol{x}) && (6.17) \\ &= \sum_{\boldsymbol{x} \setminus \boldsymbol{x}_s} f_s(\boldsymbol{x}_s) \prod_{i=1}^{M} \prod_{t \in \mathrm{ne}(x_i)} F_t(x_i, X_t) && (6.18) \\ &= f_s(\boldsymbol{x}_s) \prod_{i=1}^{M} \prod_{t \in \mathrm{ne}(x_i)} \left[\sum_{X_t} F_t(x_i, X_t) \right] && (6.19) \\ &= f_s(\boldsymbol{x}_s) \prod_{i=1}^{M} \prod_{t \in \mathrm{ne}(x_i)} \mu_{f_t \to x_i}(x_i) && (6.20) \\ &= f_s(\boldsymbol{x}_s) \prod_{i=1}^{M} \mu_{x_i \to f_s}(x_i) && (6.21) \end{aligned}$$

6.3 Framework

Graphical models have been widely used in NLP. Typically potential functions $\psi(\boldsymbol{z}, \boldsymbol{x})$ over a set of latent variables, $\boldsymbol{z}$, and observed variables, $\boldsymbol{x}$, are defined based on hand-crafted features. Moreover, independence assumptions between variables are often made for the sake of tractability. Here, we propose using neural networks (NNs) to produce the potentials since neural networks are universal function approximators. Neural networks can extract useful task-specific abstract representations of data. Additionally, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) based Recurrent Neural Networks (RNNs), allow for modeling unbounded context with far fewer parameters than naive one-hot feature encodings. The reparameterization of potentials with neural networks (NNs) is seamless:

$$\psi(\boldsymbol{z}, \boldsymbol{x}) = f_{\mathrm{NN}}(\boldsymbol{z}, \boldsymbol{x} \mid \theta) \tag{6.22}$$

The sequence of observed variables are denoted as $\boldsymbol{x} = \{x_1, \ldots, x_n\}$. In unsupervised learning, we aim to find model parameters θ that maximize the evidence $p(\boldsymbol{x} \mid \theta)$. We focus on cases when the posterior is tractable and we can use Generalized EM (Dempster et al., 1977) to estimate θ.

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{z} p(\boldsymbol{x}, z) && (6.23) \\ &= \mathbb{E}_{q(\boldsymbol{z})}[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)] + H[q(\boldsymbol{z})] + \mathrm{KL}\left(q(\boldsymbol{z}) \parallel p(\boldsymbol{z} \mid \boldsymbol{x}, \theta)\right) && (6.24) \end{aligned}$$

where $q(\boldsymbol{z})$ is an arbitrary distribution, and H is the entropy function. The E-step of EM estimates the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ based on the current parameters θ. In the M-step, we choose $q(\boldsymbol{z})$ to be the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$, setting the KL-divergence to zero. Additionally, the entropy term H$[q(\boldsymbol{z})]$ is a constant and can therefore be dropped. This means updating θ only requires maximizing $\mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x})}[\ln p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)]$. The gradient is therefore defined in terms of the gradient of the joint probability scaled by the posteriors:

$$J(\theta) = \sum_{\boldsymbol{z}} p(\boldsymbol{z} \mid \boldsymbol{x}) \frac{\partial \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)}{\partial \theta} \qquad (6.25)$$

In order to perform the gradient update in Equation 6.25, we need to compute the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$. This can be done efficiently with the sum-product algorithm (§6.2.1). Note that, in cases where the derivative $\frac{\partial}{\partial \theta} \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)$ is easy to evaluate, we can perform direct marginal likelihood optimization (Salakhutdinov et al., 2003). We do not address here the question of semi-supervised training, but believe the framework we present lends itself naturally to the incorporation of constraints or labeled data. Next, we demonstrate the application of this framework to HMMs for the purpose of part-of-speech tag induction.

6.4 Part-of-Speech Induction

Part-of-speech tags encode morphosyntactic information about a language and are a fundamental tool for many downstream NLP applications. In English, the Penn Treebank (Marcus et al., 1994) distinguishes 36 categories and punctuation. Tag induction is the task of taking raw text and both discovering these latent clusters and assigning them to words in text. Classes can be very specific (e.g. six types of verbs in English) to their syntactic role. Example tags are shown in Table 6.1. In this example, *board* is labeled as a singular noun while *Pierre Vinken* are singular proper nouns.

Two natural applications of induced tags are as the basis for grammar induction (Spitkovsky et al., 2011; Bisk et al., 2015) or to provide a syntactically informed, though unsupervised, source of word embeddings.

Text	Pierre	Vinken	will	join	the	board
PTB	NNP	NNP	MD	VB	DT	NN

Table 6.1: Example Part-of-Speech tagged text.

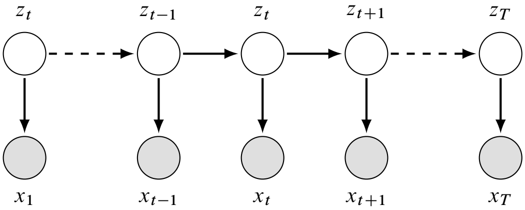


Figure 6.5: Pictorial representation of a Hidden Markov Model. Latent variable (z_t) transitions depend on the previous value (z_{t-1}), and emit an observed word (x_t) at each time step.

6.4.1 The Hidden Markov Model

A common model for part-of-speech induction, and our primary workhorse, is the Hidden Markov Model trained with the unsupervised message passing algorithm, Baum-Welch (Welch, 2003).

Model HMMs model a sentence by assuming that (a) every word token is generated by a latent class, and (b) the current class at time t is conditioned on the local history $t-1$. Formally, this gives us an emission $p(x_t \mid z_t)$ and transition $p(z_t \mid z_{t-1})$ probability. The graphical model is drawn pictorially in Figure 6.5, where shaded circles denote observations and empty ones are latent. The probability of a given sequence of observations $\mathbf{x}$ and latent variables $\mathbf{z}$ is given by multiplying transitions and emissions across all time steps (Equation 6.26). Finding the optimal sequence of latent classes corresponds to computing the argmax over the values of $\mathbf{z}$.

$$p(\boldsymbol{x}, \boldsymbol{z}) = \prod_{t=1}^{n+1} p(z_t \mid z_{t-1}) \prod_{t=1}^{n} p(x_t \mid z_t) \tag{6.26}$$

Because our task is unsupervised we do not have a priori access to these distributions, but they can be estimated via Baum-Welch. The algorithm's outline is provided in Algorithm 1.

Training an HMM with EM is highly non-convex and likely to get stuck in local optima (Johnson, 2007). Despite this, sophisticated Bayesian smoothing leads to state-of-the-art performance (Blunsom and Cohn, 2011). Blunsom and Cohn (2011) further extend the HMM by augmenting its emission distributions with character models to capture morphological information and a tri-gram transition matrix which conditions on the previous two states. Recently, Lin et al. (2015) extended several models including the HMM to include pre-trained

Algorithm 1 Baum-Welch Algorithm

```
Randomly Initialize distributions ($\theta$)
repeat
    Compute forward messages:                         $\forall_{i,t}\ \alpha_i(t)$
    Compute backward messages:                        $\forall_{i,t}\ \beta_i(t)$
    Compute posteriors:
        $p(z_t = i \mid \boldsymbol{x}, \theta) \propto \alpha_i(t)\beta_i(t)$
        $p(z_t = i, z_{t+1} = j \mid \boldsymbol{x}, \theta) \propto \alpha_i(t)p(z_{t+1}=j|z_t=i)$
                                    $\times\beta_j(t+1)p(x_{t+1}|z_{t+1}=j)$
    Update $\theta$
until Converged
```

word embeddings learned by different skip-gram models. Our work will fully neuralize the HMM and learn embeddings during the training of our generative model. There has also been recent work by Rastogi et al. (2016) on neuralizing Finite-State Transducers.

6.4.2 Additional Comparisons

The main focus of our chapter is the seamless extension of an unsupervised generative latent variable model with neural networks. For completeness we will also include comparisons to other techniques which do not adhere to the generative assumption. We include Brown clusters (Brown et al., 1992) as a baseline and two clustering techniques as state-of-the-art comparisons: Yatbaz et al. (2012) and Christodoulopoulos et al. (2011).

Of particular interest to us is the work of (Brown et al., 1992). Brown clusters group word types through a greedy agglomerative clustering according to their mutual information across the corpus based on bigram probabilities. Brown clusters do not account for a word's membership in multiple syntactic classes, but are a very strong baseline for tag induction. It is possible our approach could be improved by augmenting our objective function to include mutual information computations or a bias towards a harder clustering.

6.5 Neural HMM

The aforementioned training of an HMM assumes access to two distributions: (1) Emissions with $K \times V$ parameters, and (2) Transitions with $K \times K$ parameters. Here we assume there are K clusters and V word types in our vocabulary. Our neural HMM (NHMM) will replace these matrices with the output of simple feed-forward neural networks. All conditioning variables will be presented as input to the network and its final softmax layer will provide probabilities. This should replicate the behavior of the standard HMM, but without an explicit representation of the necessary distributions.

6.5.1 Producing Probabilities

Producing emission and transition probabilities allows standard inference to take place in the model.

Emission Architecture Let $\boldsymbol{v}_k \in \mathbb{R}^D$ be vector embedding of tag z_k, $\boldsymbol{w}_i \in \mathbb{R}^D$ and b_i vector embedding and bias of word i respectively. The emission probability $p(w_i \mid z_k)$ is given by

$$p(w_i \mid z_k) = \frac{\exp(\boldsymbol{v}_k^\top \boldsymbol{w}_i + b_i)}{\sum_{j=1}^{V} \exp(\boldsymbol{v}_k^\top \boldsymbol{w}_j + b_j)} \tag{6.27}$$

The emission probability can be implemented by a neural network where $\boldsymbol{w}_i$ is the weight of unit i at the output layer of the network. The tag embeddings $\boldsymbol{v}_k$ are obtained by a simple feed-forward neural network consisting of a lookup table following by a non-linear activation (ReLU). When using morphology information (§6.6), we will first use another network to produce the word embeddings $\boldsymbol{w}_i$.

Transition Architecture We produce the transition probability directly by using a linear layer of $K^2 \times D$. More specifically, let $\boldsymbol{q} \in \mathbb{R}^D$ be a *query embedding*. The unnormalized transition matrix $\boldsymbol{T}$ is computed as

$$\boldsymbol{T} = \boldsymbol{U}\boldsymbol{q} + \boldsymbol{b} \tag{6.28}$$

where $\boldsymbol{U} \in \mathbb{R}^{K^2 \times D}$ and $\boldsymbol{b} \in \mathbb{R}^{K^2}$. We then reshape $\boldsymbol{T}$ to a $K \times K$ matrix and apply a softmax layer per row to produce valid transition probabilities.

6.5.2 Training the Neural Network

The probabilities can now be used to perform the aforementioned forward and backward passes over the data to compute posteriors. In this way, we perform the E-step as though we were training a vanilla HMM. Traditionally, these values would simply be re-normalized during the M-step to re-estimate model parameters. Instead, we use them to re-scale our gradients (following the discussion from §6.3). Combining the HMM factorization of the joint probability

$p(\boldsymbol{x}, \boldsymbol{z})$ from Equation 6.26 with the gradient from Equation 6.25, yields the following update rule:

$$\begin{aligned} J(\theta) &= \sum_{\boldsymbol{z}} p(\boldsymbol{z} \mid \boldsymbol{x}) \frac{\partial \ln p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)}{\partial \theta} \\ &= \sum_{t} \sum_{z_t} p(z_t \mid \boldsymbol{x}) \frac{\partial \ln p(x_t \mid z_t, \theta)}{\partial \theta} \\ &\quad + \sum_{z_{t-1}} p(z_t, z_{t-1} \mid \boldsymbol{x}) \frac{\partial \ln p(z_t \mid z_{t-1}, \theta)}{\partial \theta} \end{aligned} \tag{6.29}$$

The posteriors $p(z_t \mid \boldsymbol{x})$ and $p(z_t, z_{t-1} \mid \boldsymbol{x})$ are obtained by running Baum-Welch as shown in Algorithm 1. Where traditional supervised training can follow a clear gradient signal towards a specific assignment, here we are propagating the model's (un)certainty instead. An additional complication introduced by this paradigm is the question of how many gradient steps to take on a given minibatch. In incremental EM the posteriors are simply accumulated and normalized. Here, we repeatedly recompute gradients on a minibatch until reaching the maximum number of epochs or a convergence threshold is met.

Finally, notice that the factorization of the HMM allows us to evaluate the joint distribution $p(\boldsymbol{x}, \boldsymbol{z} \mid \theta)$ easily. We therefore employ Direct Marginal Likelihood (DML) (Salakhutdinov et al., 2003) to optimize the model's parameters. After trying both EM and DML we found EM to be slower to converge and perform slightly weaker. For this reason, the presented results will all be trained with DML.

6.5.3 HMM and Neural HMM Equivalence

An important result we see in Table 6.2 is that the Neural HMM (NHMM) performs almost identically to the HMM. At this point, we have replaced the underlying machinery, but the model still has the same information bottlenecks as a standard HMM, which limit the amount and type of information carried between words in the sentence. Additionally, both approaches are optimizing the same objective function, data likelihood, via the computation of posteriors. The equivalency is an important sanity check. The following two sections will demonstrate the extensibility of this approach.

6.6 Convolutions for Morphology

The first benefit of moving to neural networks is the ease with which new information can be provided to the model. The first experiment we will perform is replacing words with embedding vectors derived from a Convolutional Neural Network (CNN) (Kim et al., 2016; Jozefowicz et al., 2016). We use a convolutional kernel with widths from 1 to 7, which covers up to 7 character n-grams

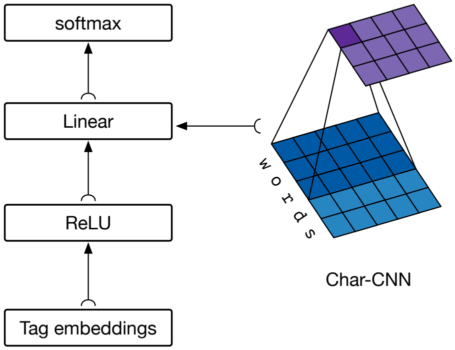


Figure 6.6: Computational graph of Char-CNN emission network. A character convolutional neural network is used to compute the weight of the linear layer for every minibatch.

(Figure 6.6). This allows the model to automatically learn lexical representations based on prefix, suffix, and stem information about a word. No additional changes to learning are required for this extension. Adding convolution does not dramatically slow down our model because the emission distributions can be computed for the whole batch in one operation. We simply pass the whole vocabulary through the convolution in a single operation.

6.7 Infinite Context with LSTMs

One of the most powerful strengths of neural networks is their ability to create compact representation of data. We will explore this here in the creation of transition matrices. In particular, we chose to augment the transition matrix with all preceding words in the sentence: $p(z_t \mid z_{t-1}, w_0, \ldots, w_{t-1})$. Incorporating this amount of context in a traditional HMM is intractable and impossible to estimate as the number of parameters grows exponentially.

For this reason, we use a stacked LSTM to form a low dimensional representation of the sentence ($C_{0\ldots t-1}$) which can be easily fed to our network when producing a transition matrix: $p(z_t \mid z_{t-1}, C_{0\ldots t-1})$ in Figure 6.7. By having the LSTM only consume up to the previous word, we do not break any sequential generative model assumptions. This interpretation does not complicate the computation of forward-backward messages when running Baum-Welch, although it does, by design, break the Markovian assumption about knowledge of the past. In terms of model architecture, the query embedding $\boldsymbol{q}$ will be replaced by a hidden state $\boldsymbol{h}_{t-1}$ of the LSTM at time step $t-1$.

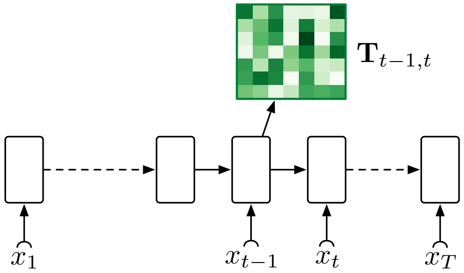


Figure 6.7: A graphical representation of our LSTM transition network. Transition matrix $\boldsymbol{T}_{t-1,t}$ from time step $t-1$ to t is computed based on the hidden state of the LSTM at time $t-1$.

6.8 Evaluation

Once a model is trained, the one-best latent sequence is extracted for every sentence and evaluated on three metrics:

Many-to-One (M-1) Many-to-one computes the most common true part-of-speech tag for each cluster. It then computes tagging accuracy as if the cluster were replaced with that tag. This metric is easily gamed by introducing a large number of clusters.

One-to-One (1-1) One-to-One performs the same computation as Many-to-One but only one cluster is allowed to be assigned to a given tag. This prevents the gaming of M-1.

V-Measure (VM) V-Measure is an F-measure which trades off conditional entropy between the clusters and gold tags. Christodoulopoulos et al. (2010) found VM is to be the most informative and consistent metric, in part because it is agnostic to the number of induced tags.

6.9 Data and Parameters

To evaluate our approaches, we follow the existing literature and train and test on the full WSJ corpus. There are three components of our models which can be tuned. This is something we have to be careful with when train and test are the same data. To avoid cheating, no values were tuned in this work.

Architecture The first parameter is the number of hidden units. We chose 512 because it was the largest power of two we could fit in memory. When we extended our model to include the convolutional emission network, we only used 128 units, due to the intensive computation of Char-CNN over the whole vocabulary per minibatch.

The second design choice was the number of LSTM layers. We used a three layer LSTM as it worked well for (Tran et al., 2016a), and we applied dropout (Srivastava et al., 2014) over the vertical connections of the LSTMs (Pham et al., 2014) with a rate of 0.5.

Finally, the maximum number of inner loop updates applied per batch is set to six. We train all the models for five epochs and perform gradient rescaling whenever the gradient norm is greater than five. To determine when to stop applying the gradient during training we simply check when the log probability has converged ($^{\text{new}-\text{old}}/_{\text{old}} < 10^{-4}$) or if the maximum number of inner loops has been reached. All optimization was done using Adam (Kingma and Ba, 2015) with default hyper-parameters.

Initialization In addition to architectural choices we have to initialize all of our parameters. Word embeddings (and character embeddings in the CNN) are drawn from a Gaussian $\mathcal{N}(0, 1)$. The weights of all linear layers in the model are drawn from a uniform distribution with mean zero and a standard deviation of $\sqrt{^1/_{n_{\text{in}}}}$, where n_{in} is the input dimension of the linear layer.[1] Additionally, weights for the LSTMs are initialized using $\mathcal{N}(0, ^1/_{2n})$, where n is the number of hidden units, and the bias of the forget gate is set to 1, as suggested by Józefowicz et al. (2015). We present some parameter and modeling ablation analysis in §6.11.

It is worth emphasizing that parameters are shared at the lower level of our network architectures (see Figure 6.6 and Figure 6.7). Sharing parameters not only allows the networks to share statistical strength, but also reduces the computational cost of computing sufficient statistics during training due to the marginalization over latent variables.

In all of our experiments, we use a mini-batch size of 256 and sentences of 40 words or less due to memory constraints. Evaluation was performed on all sentence lengths. Additionally, we map all digits to 0, but do not lower-case the data or perform any other preprocessing. All model code is available online for extension and replication at https://github.com/ketranm/neuralHMM.

6.10 Results

Our results are presented in Table 6.2 along with two baseline systems, and the four top performing and state-of-the-art approaches. As noted earlier, we are happy to see that our NHMM performs almost identically with the standard HMM. Second, we find that our approach, while simple and fast, is competitive with Blunsom and Cohn (2011)). Their Hierarchical Pitman-Yor Process

[1] This is the default parameter initialization in Torch.

	System	M-1	1-1	VM
Base	HMM	62.5	41.4	53.3
	Brown	68.2	49.9	63.0
SOTA	Clark (2003)	71.2		65.6
	Christodoulopoulos (2011)	72.8		66.1
	Blunsom (2011)	**77.5**		**69.8**
	Yatbaz (2012)	80.2		72.1
Our Work	NHMM	59.8	45.7	54.2
	+ Conv	74.1	48.3	66.1
	+ LSTM	65.1	52.4	60.4
	+ Conv & LSTM	**79.1**	**60.7**	**71.7**

Table 6.2: English Penn Treebank results with 45 induced clusters. We see significant gains from both morphology (+Conv) and extended context (+LSTM). The combination of these approaches results in a very simple system which is competitive with the best generative model in the literature.

Configuration	M-1	1-1	VM
Uniform initialization	65.5	50.1	61.7
1 LSTM layer, no dropout	69.3	52.7	63.6
1 LSTM layer, dropout	71.0	55.7	66.2
3 LSTM layers, no dropout	72.7	52.2	65.1
Best Model	**79.1**	**60.7**	**71.7**

Table 6.3: Exploring different configurations of NHMM

for trigram HMMs with character modeling is a very sophisticated Bayesian approach and the most appropriate comparison to our work.

We see that both extended context (+LSTM) and the addition of morphological information (+Conv) substantially boost the performance. Interestingly, the gains are not completely complementary, as we note that the six and twelve point gains of these additions only combine to a total of sixteen points in VM improvement. This might imply that at least some of the syntactic context being captured by the LSTM is mirrored in the morphology of the language. This hypothesis is something future work should investigate with morphologically rich languages.

Finally, the newer work of (Yatbaz et al., 2012) outperforms our approach. It is possible our performance could be improved by following their lead and including knowledge of the future.

6.11 Parameter Ablation

Our model design decisions and weight initializations were chosen based on best practices set forth in the supervised training literature. Fortunately they also behaved well in the unsupervised setting. Within unsupervised structure prediction, to the best of our knowledge, there has been no empirical study on neural network architecture design and weight initialization. We therefore provide an initial overview on the topic for several of our decisions.

Weight Initialization We run our best model (NHMM+Conv+LSTM) with all the weights initialized from a uniform distribution $\mathrm{uniform}(-10^{-4}, 10^{-4})$. We choose small standard derivation here for numerical stability when computing forward-backward messages. We find a dramatic drop in V-Measure performance (61.7 vs. 71.7 in Table 6.3). This is consistent with the common wisdom that unlike supervised learning (Luong et al., 2015a), weight initialization is important to achieve good performance on unsupervised tasks. It is possible that performance could be further enhanced via the popular technique of ensembling. This would allow for combining models which converged to different local optima.

LSTM Layers And Dropout We find that dropout is important in training an unsupervised NHMM. Removing dropout causes performance to drop six points. To avoid tuning the dropout rate, future work might investigate the effect of variational dropout (Kingma et al., 2015) in unsupervised learning. We also observed that the number of LSTM layers has an impact on V-Measure. Had we simply used a single layer we would have lost nearly five points. It is possible that more layers, perhaps coupled with more data, would yield even greater gains.

6.12 Conclusion

In this chapter, we have answered **Research Question 4** *Can neural networks be used to induce linguistic structure in a completely unsupervised manner?* In order to answer this research question, we have broken it down into three sub-questions and provided an answer to each of those. In the following, we revisit the three sub-questions posed at the beginning of the chapter and highlight the main contributions and findings in addressing each individual sub-question.

RQ4.1 *Can an unsupervised Hidden Markov Model be fully expressed as a neural network model?*

We proposed a neural network parametrization for transition and emission potentials in HMM. This allows us to express an unsupervised HMM as a computational graph consisting of two neural networks: transition and emission networks.

RQ4.2 *What are the advantages of the neural methods in comparison to traditional Bayesian methods?*

We demonstrated that neural methods enable seamless integration of morphological features via char-CNN and unbounded context via LSTM.

RQ4.3 *Are neural models sensitive to parameter initialization in the same way as non-neural models are?*

We validated empirically with different parameter initialization and have found that our neural HMMs are sensitive to initialization.

Within the context of part-of-speech induction, in addition to parameter tuning and multilingual evaluation, the biggest open questions for our approach are the effects of additional data and augmenting the loss function. Neural networks are notoriously data hungry, indicating that while we achieve competitive results, it is possible our model will scale well when running with large corpora. This would likely require the use of techniques like noise contrastive estimation (Gutmann and Hyvärinen, 2010) which have been shown to be highly effective in related tasks like neural language modeling (Mnih and Teh, 2012; Vaswani et al., 2013). Secondly, despite focusing on ways to augment an HMM, Brown clustering and systems inspired by it perform very well. They aim to maximize mutual information rather than likelihood. It is possible that augmenting or constraining our loss will yield additional performance gains.

Beyond simply maximizing performance on tag induction, a more subtle, but powerful contribution of this chapter may be its demonstration of the ease and effective nature of using neural networks with Bayesian models traditionally trained with EM. We hope this approach transfers well to many other domains and tasks.

Chapter 7

Inducing Grammars with and for Neural Machine Translation

7.1 Introduction and Research Questions

In the previous chapter, we have demonstrated an application of an unsupervised neural latent variable model for discovering syntactic categories in the data. These discovered syntactic categories can be used as additional features in an NLP system. However, the features learned by an unsupervised model might not be optimal for a given language task. In this chapter, we investigate a type of models that can jointly induce structure from the input and optimize the task performance. In this way, the learned structure is expected to be directly beneficial for the task of interested. We choose sequence to sequence (seq2seq), a general purpose model, to study in this chapter.

Seq2seq models have exploded in popularity due to their apparent simplicity and yet surprising modeling strength. These models have been strengthened with attention mechanisms (Bahdanau et al., 2015), and variational dropout (Gal and Ghahramani, 2016), in addition to important advances in expressivity via gating like Long Short-Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997) and advanced gradient optimizers like Adam (Kingma and Ba, 2015).

Despite these impressive advances, the community has still largely been at a loss to explain how these models are so successful at a wide range of linguistic tasks. Recent work has shown that the LSTM captures a surprising amount of syntax implicitly (Shi et al., 2016; Linzen et al., 2016; Belinkov et al., 2017), but this is evaluated via downstream tasks designed to test the model's abilities and does not explore what kind of structure is needed to maximize task performance.

Our goal is to develop a neural network model that operates *explicitly* on *latent linguistic structure* of the input. By making the computation explicit, we can inspect the model's learned structure from a linguistic perspective with respect to NLP tasks. An important desideratum of the model is that it should learn the structure of the data without being provided with any structured representation (*i.e.*, syntactic trees). Therefore, we design our model based on the idea of integrating *discrete* latent variable models with neural networks presented in Chapter 6. In contrast to the unsupervised model in Chapter 6, the model in this chapter is trained with a supervised objective while simultaneously discovering useful (linguistic) structure that is beneficial to a task of interest. Concretely, we ask:

Research question 5: *What type of model can learn to induce linguistic structure and utilize those learnt structures to improve task-specific performance?*

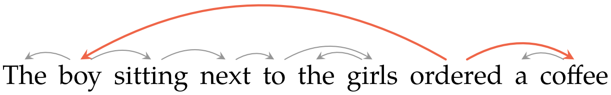


Figure 7.1: Our model aims to capture **syntactic** dependency (verb *ordered*).

In this work, we choose neural machine translation (NMT) as a task of interest. Language has syntactic structure and translation models need to understand grammatical dependencies to resolve the semantics of a sentence and preserve agreement (e.g. number, gender, etc). To motivate our work and the importance of structure in translation, consider the process of translating the sentence "*The boy sitting next to the girls ordered a coffee*" from English to German. The dependency tree of the sentence is given in Figure 7.1. In German, translating the verb "*ordered*", requires knowledge of its subject "*boy*" to correctly predict the verb's number "*bestellte*" instead of "*bestellten*" if the model wrongly identifies "*girls*" as the subject. This is a case where syntactic agreement requires long-distance information transfer. On the other hand, translating the word "*next*" can be done in isolation without knowledge of neither its head nor child dependencies. While its true the decoder can, in principle, utilize previously predicted words (*e.g.* the translation of the "*boy*") to reason about subject-verb agreement, in practice LSTMs still struggle with long-distance dependencies (Linzen et al., 2016). Moreover, Belinkov et al. (2017) showed that using attention reduces the capacity of the decoder to learn target side syntax.

Recent research in NMT has shown the benefit of modeling syntax explicitly using parse trees (Bastings et al., 2017; Li et al., 2017; Eriguchi et al., 2017) rather than assuming the model will automatically discover and encode it. Li et al. (2017) present a mixed encoding of words and a linearized constituency-based parse tree of the source sentence. Bastings et al. (2017) propose to use Graph

Convolution to encode source sentences given their dependency links and attachment labels. Here, we attempt to contribute to both modeling syntax and investigating a more interpretable interface for testing the syntactic content of a new seq2seq model's internal representation. To achieve our goal, we break **Research Question 5** into three sub-questions:

RQ5.1 *Can we design a translation model that learns to induce a dependency tree-like representation of the source sentence?*

We achieve this by augmenting seq2seq with a gate that allows the model to decide between syntactic and alignment objectives. The syntactic objective is encoded via a syntactic structured attention (Section §7.3) from which we can extract dependency trees. Our goal is to have a model which reaps the benefits of syntactic information (*i.e.* parse trees) without requiring explicit annotation. In this way, learning the internal representation of our model is related to work done in unsupervised grammar induction except that by focusing on translation we require both syntactic and semantic knowledge. The alignment objective is the word translation prediction. It is often captured by attention, as an analogy to word-alignment model in phrase-based MT (Koehn et al., 2003). The syntactic objective is captured implicitly in the decoder because it ensures the fluency of the translation. For grammar induction, the translation objective provides more guidance than the marginal likelihood typically used in unsupervised learning (Chapter 6). However, we note that the quality of the induced grammar also depends on the choice of the target language (§7.6).

RQ5.2 *Are the dependency trees learnt with the translation objective recognizable from a linguistic perspective?*

In addition to demonstrating improvements in translation quality with our proposed models, we are also interested in analyzing the aforementioned dependency trees learned by the model. Recent work has begun to analyze task-specific latent trees (Williams et al., 2018). It has been shown that incorporating hierarchical structures leads to better task performance. Unlike the previous work that induced latent trees explicitly for a language inference task, we present the first results on learning latent trees with a translation objective.

RQ5.3 *Is there any correlation between the interpretability of the learned structure and the translation quality?*

While our proposed model improves NMT, we will show that there is no correlation between the learned dependency's quality and translation quality. This suggests that when optimizing for task-specific performance, the model might benefit from using a different structure that is not recognizable to linguistics.

The rest of the chapter is organized as follow: We describe our NMT baseline in Section §7.2. Our proposed models are detailed in Section §7.3. We present the experimental setups and translation results in Section §7.4. In Section §7.5 we analyze models' behavior by means of visualization which pairs with our analysis of the latent trees induced by our model in Section §7.6. We conclude this chapter in Section §7.7.

7.2 Neural Machine Translation

Given a training pair of source and target sentences $(\boldsymbol{x}, \boldsymbol{y})$ of length n and m respectively, Neural Machine Translation (NMT) is a conditional probabilistic model $p(\boldsymbol{y} \mid \boldsymbol{x})$ implemented using neural networks

$$\log p(\boldsymbol{y} \mid \boldsymbol{x};\ \boldsymbol{\theta}) = \sum_{j=1}^{m} \log p(\boldsymbol{y}_j \mid \boldsymbol{y}_{i<j}, \boldsymbol{x};\ \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is the model's parameters. We will omit the parameters $\boldsymbol{\theta}$ herein for readability.

The NMT system used in this work is a seq2seq model that consists of a bidirectional LSTM encoder and an LSTM decoder coupled with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a). Our system is based on a PyTorch implementation[1] of OpenNMT (Klein et al., 2017). Let $\{\boldsymbol{s}_i \in \mathbb{R}^d\}_{i=1}^n$ be the output of the encoder

$$\boldsymbol{S} = \text{BiLSTM}_{\text{enc}}(\boldsymbol{x}) \tag{7.1}$$

Here we use $\boldsymbol{S} = [\boldsymbol{s}_1; \ldots; \boldsymbol{s}_n] \in \mathbb{R}^{d \times n}$ as a concatenation of $\{\boldsymbol{s}_i\}$. The decoder is composed of stacked LSTMs with input-feeding. Specifically, the inputs of the decoder at time step t are the previous hidden state $\boldsymbol{h}_{t-1}$, a concatenation of the embedding of previous generated word $\boldsymbol{y}_{t-1}$ and a vector $\boldsymbol{u}_{t-1}$:

$$\boldsymbol{h}_{t-1} = \text{LSTM}_{\text{dec}}([\boldsymbol{y}_{t-2}; \boldsymbol{u}_{t-2}]) \tag{7.2}$$

$$\boldsymbol{u}_{t-1} = g(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}) \tag{7.3}$$

where g is a one layer feed-forward network and $\boldsymbol{c}_{t-1}$ is a context vector computed by an attention mechanism

$$\boldsymbol{\alpha}_{t-1} = \text{softmax}(\boldsymbol{h}_{t-1}^\top \boldsymbol{W}_a \boldsymbol{S}) \tag{7.4}$$

$$\boldsymbol{c}_{t-1} = \boldsymbol{S}\boldsymbol{\alpha}_{t-1}^\top \tag{7.5}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{d \times d}$ is a trainable weight matrix.

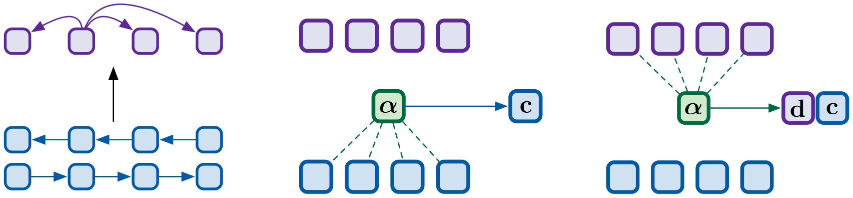
Finally a single layer feed-forward network f takes $\boldsymbol{u}_t$ as input and returns a multinomial distribution over all the target words

$$y_t \sim f(\boldsymbol{u}_t) \tag{7.6}$$

[1] http://opennmt.net/OpenNMT-py/

7.3 Syntactic Attention Models

We propose a syntactic attention model[2] (Figure 7.2) that differs from standard NMT in two crucial aspects. First, our encoder outputs two sets of annotations: content annotations $\boldsymbol{S}$ and syntactic annotations $\boldsymbol{M}$ (Figure 7.2a). The content annotations are the outputs of a standard BiLSTM while the syntactic annotations are produced by a head word selection layer (§7.3.1). The syntactic annotations $\boldsymbol{M}$ capture syntactic dependencies amongst the source words and enable syntactic transfer from the source to the target. Second, we incorporate the source side syntax into our model by modifying the standard attention (from target to source) in NMT such that it attends to both $\boldsymbol{S}$ and $\boldsymbol{M}$ through a *shared attention* layer. The shared attention layer biases our model toward capturing source side dependency. It produces a dependency context $\boldsymbol{d}$ (Figure 7.2c) in addition to the standard context vector $\boldsymbol{c}$ (Figure 7.2b) at each time step. Motivated by the example in Figure 7.1 that some words can be translated without resolving their syntactic roles in the source sentence, we include a gating mechanism that allows the decoder to decide the amount of syntax needed when it generates the next word. Next, we describe the head word selection layer and how source side syntax is incorporated into our model.



(a) Structured Self Attention Encoder: the first layer is a standard BiLSTM, the top layer is a syntactic attention network.

(b) Compute the context vector (blue) as in a standard NMT model. The attention weights $\boldsymbol{\alpha}$ are in green.

(c) Use the attention weights $\boldsymbol{\alpha}$, as computed in the previous step, to calculate syntactic vector (purple).

Figure 7.2: A visual representation of our proposed mechanism for shared attention.

7.3.1 Head Word Selection

The head word selection layer learns to select a *soft* head word for each source word via structured attention. This layer does not have access to any dependency labels from the source. The head word selection layer transforms $\boldsymbol{S}$ into a matrix $\boldsymbol{M}$ that encodes implicit dependency structure of $\boldsymbol{x}$ using *self-structured-attention*. First we apply three trainable weight matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d\times d}$ to map $\boldsymbol{S}$ to query, key, and value matrices $\boldsymbol{S}_q, \boldsymbol{S}_k, \boldsymbol{S}_v \in \mathbb{R}^{d\times n}$ respectively:

$$\boldsymbol{S}_q = \boldsymbol{W}_q \boldsymbol{S} \qquad \boldsymbol{S}_k = \boldsymbol{W}_k \boldsymbol{S} \qquad \boldsymbol{S}_v = \boldsymbol{W}_v \boldsymbol{S} \tag{7.7}$$

[2] https://github.com/ketranm/sa-nmt

Then we compute structured attention probabilities $\boldsymbol{\beta}$ relying on a function *sattn* that we will describe in detail shortly.

$$\boldsymbol{\beta} = \text{sattn}(\boldsymbol{S}_q^\top \boldsymbol{S}_k) \tag{7.8}$$

$$\boldsymbol{M} = \boldsymbol{S}_v \boldsymbol{\beta} \tag{7.9}$$

The structured self attention function *sattn* is inspired by the work of Kim et al. (2017) but differs in two important ways. First we model *non-projective dependency trees*. Second, we utilize the Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) instead of the sum-product algorithm presented in (Kim et al., 2017) for fast evaluation of the attention probabilities. We note that Liu and Lapata (2018) first propose using the Matrix-Tree Theorem for evaluating the marginals in end-to-end training of neural networks. Their work, however, focuses on the task of natural language inference (Bowman et al., 2015a) and document classification instead of machine translation. Additionally, we will evaluate our structured self attention on datasets that are up to 20 times larger than the datasets studied in previous work (Liu and Lapata, 2018).

Let $\boldsymbol{z} \in \{0,1\}^{n\times n}$ be an adjacency matrix encoding a source's dependency tree. Let $\boldsymbol{\phi} \in \mathbb{R}^{n\times n}$ be a scoring matrix such that cell $\boldsymbol{\phi}_{i,j}$ scores how likely word x_i is to be the head of word x_j. The matrix $\boldsymbol{\phi}$ is obtained simply by

$$\boldsymbol{\phi} = \boldsymbol{S}_q^\top \boldsymbol{S}_k \tag{7.10}$$

The probability of a dependency tree $\boldsymbol{z}$ is therefore given by

$$p(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\phi}) = \frac{\exp\left(\sum_{i,j} \boldsymbol{z}_{i,j}\ \boldsymbol{\phi}_{i,j}\right)}{Z(\boldsymbol{\phi})} \tag{7.11}$$

where $Z(\boldsymbol{\phi})$ is the partition function.

In the head selection model, we are interested in the marginal $p(\boldsymbol{z}_{i,j} = 1 \mid \boldsymbol{x}; \boldsymbol{\phi})$

$$\boldsymbol{\beta}_{i,j} = p(\boldsymbol{z}_{i,j} = 1 \mid \boldsymbol{x}; \boldsymbol{\phi}) = \sum_{\boldsymbol{z} \,:\, \boldsymbol{z}_{i,j}=1} p(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\phi}) \tag{7.12}$$

We use the framework presented by Koo et al. (2007) to compute the marginal of non-projective dependency structures. Koo et al. (2007) use the Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) to compute $p(\boldsymbol{z}_{i,j} = 1 \mid \boldsymbol{x}; \boldsymbol{\phi})$ by first defining the Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ as follows:

$$\boldsymbol{L}_{i,j}(\boldsymbol{\phi}) = \begin{cases} \sum\limits_{\substack{k=1 \\ k\neq j}}^{n} \exp(\boldsymbol{\phi}_{k,j}) & \text{if } i = j \\ -\exp(\boldsymbol{\phi}_{i,j}) & \text{otherwise} \end{cases} \tag{7.13}$$

Now we construct a matrix $\hat{\boldsymbol{L}}$ that accounts for root selection

$$\hat{\boldsymbol{L}}_{i,j}(\boldsymbol{\phi}) = \begin{cases} \exp(\boldsymbol{\phi}_{j,j}) & \text{if } i = 1 \\ \boldsymbol{L}_{i,j}(\boldsymbol{\phi}) & \text{if } i > 1 \end{cases} \tag{7.14}$$

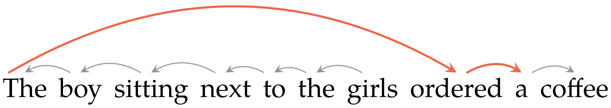


Figure 7.3: A latent tree learned by our model. Notice that the encoder decides *"The"* is the head of *"ordered"*. This dependency may provide a useful clue for the translation of *"ordered"* since the information at the head word *"The"* also contains the information about the neighboring subject *"boy"* as a result of using a bidirectional LSTM layer in the encoder.

The marginals $\boldsymbol{\beta}$ are then

$$\boldsymbol{\beta}_{i,j} = (1 - \delta_{1,j}) \exp(\boldsymbol{\phi}_{i,j}) \left[\hat{\boldsymbol{L}}^{-1}(\boldsymbol{\phi})\right]_{j,j} - (1 - \delta_{i,1}) \exp(\boldsymbol{\phi}_{i,j}) \left[\hat{\boldsymbol{L}}^{-1}(\boldsymbol{\phi})\right]_{j,i} \tag{7.15}$$

where $\delta_{i,j}$ is the Kronecker delta. For the root node, the marginals are given by

$$\boldsymbol{\beta}_{k,k} = \exp(\boldsymbol{\phi}_{k,k}) \left[\hat{\boldsymbol{L}}^{-1}(\boldsymbol{\phi})\right]_{k,1} \tag{7.16}$$

The computation of the marginals is fully differentiable, thus we can train the model in an end-to-end fashion by maximizing the conditional likelihood of the translation.

7.3.2 Incorporating Syntactic Context

We encourage the decoder to use syntactic annotations by means of attention. Essentially, if the model attends to a particular source word x_i when generating the next target word, we also want the model to attend to the head word of x_i. We implement this idea using a new *shared attention* layer from decoder's state $\boldsymbol{h}$ to encoder's annotations $\boldsymbol{S}$ and $\boldsymbol{M}$. First, we compute standard attention weights $\boldsymbol{\alpha}_{t-1} = \text{softmax}(\boldsymbol{h}_{t-1}^{\top} W_a \boldsymbol{S})$ as in Equation 7.4. We then compute a weighted syntactic vector:

$$\boldsymbol{d}_{t-1} = \boldsymbol{M}\boldsymbol{\alpha}_{t-1}^{\top} \tag{7.17}$$

Note that the syntactic vector $\boldsymbol{d}_{t-1}$ and the context vector $\boldsymbol{c}_{t-1}$ share the same attention weights $\boldsymbol{\alpha}_{t-1}$ at time step t. We share the attention weights $\boldsymbol{\alpha}_{t-1}$ because we expect that, if the model picks a source word x_i to translate with the highest probability $\boldsymbol{\alpha}_{t-1}[i]$, the contribution of x_i's head in the syntactic vector $\boldsymbol{d}_{t-1}$ should also be highest. Figure 7.3 shows the latent tree learned by our translation objective. Unlike the gold tree provided in Figure 7.1, the model decided that *"the boy"* is the head of *"ordered"*. This is common in our model because the BiLSTM context is actually a summary of its local context/constituent.

It is not always useful or necessary to access the syntactic context $\boldsymbol{d}_{t-1}$ every time step t. Ideally, we should let the model decide whether it needs to use this information. For example, the model might decide when it needs to resolve long distance dependencies in the source side. To control the amount of source side syntactic information we introduce a gating mechanism:

$$\hat{\boldsymbol{d}}_{t-1} = \boldsymbol{d}_{t-1} \odot \sigma(\boldsymbol{W}_g \boldsymbol{h}_{t-1}) \tag{7.18}$$

The vector $\boldsymbol{u}_{t-1}$ from Equation 7.2 now becomes

$$\boldsymbol{u}_{t-1} = g(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \hat{\boldsymbol{d}}_{t-1}) \tag{7.19}$$

An alternative to incorporate syntactic annotation $\boldsymbol{M}$ to the decoder is to use a separate attention layer to compute the syntactic vector $\boldsymbol{d}_{t-1}$ at time step t:

$$\boldsymbol{\gamma}_{t-1} = \mathrm{softmax}(\boldsymbol{h}_{t-1}^{\top} \boldsymbol{W}_m \boldsymbol{M}) \tag{7.20}$$

$$\boldsymbol{d}_{t-1} = \boldsymbol{M} \boldsymbol{\gamma}_{t-1}^{\top} \tag{7.21}$$

We will provide a comparison to this approach in our results.

7.3.3 Hard Attention over Tree Structures

Finally, to simulate the scenario where the model has access to a dependency tree given by an external parser we report results with hard attention. Forcing the model to make hard decisions during training mirrors the extraction and conditioning on a dependency tree (§7.6.1). We expect this technique will improve the performance on grammar induction, despite making translation lossy. A similar observation has been reported in (Hashimoto and Tsuruoka, 2017) which showed that translation performance degraded below their baseline when they provided dependency trees to the encoder.

Recall the marginal $\boldsymbol{\beta}_{i,j}$ gives us the probability that word x_i is the head of word x_j. We convert these soft weights to hard ones $\bar{\boldsymbol{\beta}}$ by

$$\bar{\boldsymbol{\beta}}_{k,j} = \begin{cases} 1 & \text{if } k = \arg\max_i \boldsymbol{\beta}_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{7.22}$$

We train this model using the straight-through estimator (Bengio et al., 2013). Note that in this setup, each word has a parent but there is no guarantee that the structure given by hard attention will result in a tree (*i.e.* it may contain cycles). A more principled way to enforce tree structure is to decode the best tree $\mathcal{T}$ using the maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967) and to set $\bar{\boldsymbol{\beta}}_{k,j} = 1$ if the edge $(x_k \rightarrow x_j) \in \mathcal{T}$. Unfortunately, maximum spanning tree decoding can be prohibitively slow as the Chu-Liu-Edmonds algorithm is not GPU friendly. We therefore resort to greedily picking a parent word for each word x_j in the sentence using Equation 7.22. This is a principled simplification as greedily assigning a parent for each word is the first step in Chu-Liu-Edmonds algorithm.

	Train	Valid	Test	Vocabulary
En↔De	5.9M	2,169	2,999 / 3,004	36,251 / 35,913
En↔Ru	2.1M	2,998	3,001	34,872 / 34,989
Ru→Ar	11.1M	4,000	4,000	32,735 / 32,955

Table 7.1: Statistics of the data.

7.4 Experiments

In this section, we will discuss our experimental setup and report results for English↔German (En↔De), English↔Russian (En↔Ru), and Russian→Arabic (Ru→Ar) translation models.

7.4.1 Data

We use the WMT17 (Bojar et al., 2017) data in our experiments. Table 7.1 shows the statistics of the data. For En↔De, we use a concatenation of Europarl, Common Crawl, Rapid corpus of EU press releases, and News Commentary v12. We use *newstest2015* for development and *newstest2016*, *newstest2017* for testing. For En↔Ru, we use Common Crawl, News Commentary v12, and Yandex Corpus. The development data comes from *newstest2016* and *newstest2017* and is reserved for testing. For Ru→Ar, we use the data from the six-way sentence-aligned subcorpus of the United Nations Parallel Corpus v1.0 (Ziemski et al., 2016). The corpus also contains the official development and test data.

Our language pairs were chosen to compare results across and between morphologically rich and poor languages. This will prove particularly interesting in our grammar induction results where different pairs must preserve different amounts of syntactic agreement information.

We use BPE (Sennrich et al., 2016) with 32,000 merge operations. We run BPE for each language instead of using BPE for the concatenation of both source and target languages.

7.4.2 Baselines

Our baseline is an NMT model with input-feeding (§7.2). As we will be making several modifications from the basic architecture in our proposed structured self attention NMT (SA-NMT), we will verify each choice in our architecture design empirically. First we validate the structured self attention module by comparing it to a self-attention module (Lin et al., 2017; Vaswani et al., 2017). Self attention computes attention weights $\boldsymbol{\beta}$ simply as $\boldsymbol{\beta} = \text{softmax}(\boldsymbol{\phi})$. Since self-attention does not assume any hierarchical structure over the source sentence, we refer it as flat-attention NMT (FA-NMT). Second, we validate the benefit of using two sets of annotations in the encoder. We combine the hidden states of the encoder $\boldsymbol{s}$ with syntactic context $\boldsymbol{d}$ to obtain a single set of annotation using the following equation:

$$\bar{\boldsymbol{s}}_i = \boldsymbol{s}_i + \sigma(\boldsymbol{W}_g \boldsymbol{s}_i) \odot \boldsymbol{d}_i \tag{7.23}$$

Here we first down-weigh the syntactic context $\boldsymbol{d}_i$ before adding it to $\boldsymbol{s}_i$. The sigmoid function $\sigma(\boldsymbol{W}_g \boldsymbol{s}_i)$ decides the weight of the head word of x_i based on whether translating x_i needs additinaly dependency information. We refer to this baseline as SA-NMT-1set. Note that in this baseline, there is only one attention layer from the target to the source $\bar{\boldsymbol{S}} = \{\bar{\boldsymbol{s}}_i\}_1^n$.

In all the models, we share the weights of target word embeddings and the output layer as suggested by Inan et al. (2016); Press and Wolf (2017).

7.4.3 Hyper-parameters and Training

For all the models, we set the word embedding size to 1024, the number of LSTM layers to 2, and the dropout rate to 0.3. Parameters are initialized uniformly in $(-0.04, 0.04)$. We use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.001. We evaluate our models on development data every 10,000 updates for De↔En and Ru→Ar, and every 5,000 updates for Ru↔En. If the validation perplexity increases, we decay the learning rate by 0.5. We stop training after decaying the learning rate five times as suggested by Denkowski and Neubig (2017). The mini-batch size is 64 in Ru→Ar experiments and 32 in the rest. We report the BLEU scores using the multi-bleu.perl script.

7.4.4 Translation Results

Table 7.2 shows the BLEU scores in our experiments. We test statistical significance using bootstrap resampling (Riezler and Maxwell, 2005). Statisical significance are marked as $^{\dagger}p < 0.05$ and $^{\ddagger}p < 0.01$ when compared against the baselines. Additionally, we also report statistical significance $^{\triangle}p < 0.05$ and $^{\blacktriangle}p < 0.01$ when compared against the FA-NMT models that have two separate attention layers from the decoder to the encoder. Overall, the SA-NMT (shared) model performs best, gaining more than 0.5 BLEU De→En on wmt16, up to 0.82 BLEU on En→De wmt17 and 0.64 BLEU En→Ru direction over a competitive NMT baseline. The gain of the SA-NMT model on Ru→Ar is small (0.45 BLEU) but significant. The results show that structured attention is useful when translating from English to languages that have long-distance dependencies and complex morphological agreements. We also see that the gain is marginal compared to self-attention models (FA-NMT) and not significant. Within FA-NMT models, sharing attention is helpful. Our results also confirm the advantage of having two separate sets of annotations in the encoder when modeling syntax.

The hard structured attention model (SA-NMT-hard) performs comparable to the baseline. While this is a somewhat expected result from the hard attention model, we will show in the next section (§7.6) that the quality of induced trees from hard attention is far better than the soft ones.

Model	Shared	De→En		Ru→En	En→De		En→Ru	Ru→Ar
		wmt16	wmt17	wmt17	wmt16	wmt17	wmt17	
NMT	-	33.16	28.94	30.17	29.92	23.44	26.41	37.04
FA	yes	33.55	29.43	30.22	30.09	24.03	26.91	37.41
	no	33.24	29.00	30.34	29.98	23.97	26.75	37.20
SA-1set	-	33.51	29.15	30.34	**30.29**†	24.12	26.96	37.34
SA-hard	yes	33.38	28.96	29.98	29.93	23.84	26.71	37.33
SA	yes	**33.73**$^{\ddagger\triangle}$	**29.45**$^{\ddagger\blacktriangle}$	**30.41**	30.22	**24.26**$^{\ddagger\triangle}$	**27.05**‡	**37.49**$^{\ddagger\triangle}$
	no	33.18	29.19	30.15	30.17	23.94	27.01	37.22

Table 7.2: Results for translating En↔De, En↔Ru, and Ru→Ar. Statistical significances are marked as $^{\dagger}p < 0.05$ and $^{\ddagger}p < 0.01$ when compared against the baselines and $^{\triangle}/^{\blacktriangle}$ when compared against the FA-NMT (no-shared). The results indicate the strength of our proposed *shared-attention* for NMT.

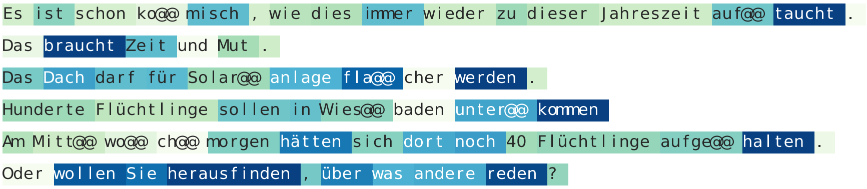
7.5 Gate Activation Visualization

As mentioned earlier, our models allow us to ask the question: *When does the target LSTM need to access source side structural information?* We investigate this by analyzing the gate activations of our best model, SA-NMT (shared). At time step t, when the model is about to predict the target word y_t, we compute the norm of the gate activations

$$z_t = \|\sigma(\boldsymbol{W}_g \boldsymbol{h}_{t-1})\|_2 \tag{7.24}$$

The activation norm z_t allows us to see how much syntactic information flows into the decoder. We collect statistics of gate activation norm on En→De *newstest2016* dataset and observe that z_t has its highest value when the decoder is about to generate a verb while it has its lowest value when the end of sentence token </s> is predicted. Figure 7.4 shows some examples of German target sentences. The darker colors represent higher activation norms and bold words indicate the highest activation norms when those words are being predicted.

Figure 7.4: Visualization of gate activation norm. Darker means the model is using more syntactic information.



Es ist schon ko@@ misch , wie dies immer wieder zu dieser Jahreszeit auf@@ taucht .

Das braucht Zeit und Mut .

Das Dach darf für Solar@@ anlage fla@@ cher werden .

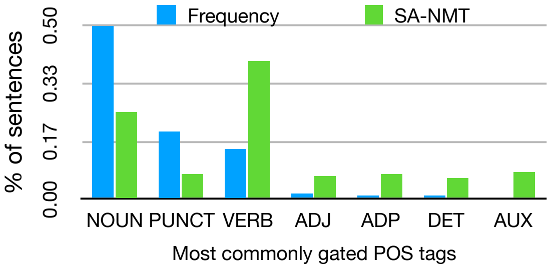
Hunderte Flüchtlinge sollen in Wies@@ baden unter@@ kommen

Am Mitt@@ wo@@ ch@@ morgen hätten sich dort noch 40 Flüchtlinge aufge@@ halten .

Oder wollen Sie herausfinden , über was andere reden ?

It is clear that translating verbs requires structural information. We also see that after verbs, the gate activation norms are highest at nouns Zeit (*time*), Mut (*courage*), Dach (*roof*) and then tail off as we move to function words which require less context to disambiguate. Below are the frequencies with which the highest activation norm in a sentence is applied to a given part-of-speech tag on *newstest2016*. We include the top 7 most common activations.

Figure 7.5: Top part-of-speech tags with the highest activation norm when the model is forced to decode the target sentences.



We see that while nouns are often the most common tag in a sentence, syntax is disproportionately used for translating verbs.

7.6 Grammar Induction

NLP has long assumed that hierarchical structured representations were important to understanding language. In this work, we have borrowed that intuition to inform the construction of our model. We investigate whether the internal latent representations discovered by our models share syntactic properties previously identified within linguistics and if not, what important differences exist. We investigate the interpretability of our model's representations by: 1) A quantitative attachment accuracy and 2) A qualitative look at model outputs.

Unlike in the grammar induction literature our model is not specifically constructed to recover traditional dependency grammars nor have we provided the model with access to part-of-speech tags or universal rules (Naseem et al., 2010; Bisk and Hockenmaier, 2013). The model only uncovers the syntactic information necessary for translation in a given language pair, though future work should investigate if structural linguistic constraints benefit MT.

7.6.1 Extracting a Tree

For extracting non-projective dependency trees, we use Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). First, we must collapse BPE segments into words. Assume the k-th word corresponds to BPE tokens from index u to v. We obtain a new matrix $\hat{\boldsymbol{\phi}}$ by summing over $\boldsymbol{\phi}_{i,j}$ that are the corresponding BPE segments.

$$\hat{\boldsymbol{\phi}}_{i,j} = \begin{cases} \boldsymbol{\phi}_{i,j} & \text{if } i \notin [u,v] \wedge j \notin [u,v] \\ \sum_{l=u}^{v} \boldsymbol{\phi}_{i,l} & \text{if } j = k \wedge i \notin [u,v] \\ \sum_{l=u}^{v} \boldsymbol{\phi}_{l,j} & \text{if } i = k \wedge j \notin [u,v] \\ \sum_{l=u}^{v} \sum_{h=u}^{v} \boldsymbol{\phi}_{l,h} & \text{otherwise} \end{cases} \tag{7.25}$$

7.6.2 Grammatical Analysis

To analyze performance we compute unlabeled directed and undirected attachment accuracies of our predicted trees on gold annotations from Universal Dependencies (UD version 2) dataset[3]. We choose this representation because of its availability in many languages, though it is atypical for grammar induction. Our five model settings in addition to left and right branching baselines are presented in Table 7.3. The results indicate that the target language effects the source encoder's induction performance and several settings are competitive with branching baselines for determining headedness. Recall that syntax is being modeled on the source language so adjacent rows are comparable.

We observe a huge boost in DA/UA scores for English and Russian in FA-NMT and SA-NMT-shared models when the target language are morphologically rich (Russian and Arabic respectively). In comparison to previous work (Belinkov et al., 2017; Shi et al., 2016) on the encoder's ability to capture source side syntax, we show a stronger result that even when the encoders are designed to capture syntax explicitly, the choice of the target language has a great influence on the amount of syntax learned by the encoder.

We also see performance gains from hard attention and several models outperform baselines for undirected dependency metrics (UA). Hard attention appears to help when the target languages are morphologically rich.

[3] http://universaldependencies.org

	FA		SA			Baseline		
	no-shared	shared	no-shared	shared	hard	L	R	Un
EN (→DE)	17.0/25.2	27.6/41.3	23.6/33.7	27.8/42.6	**31.7**/**45.6**	34.0	7.8	40.9
EN (→RU)	35.2/48.5	**36.5**/48.8	12.8/25.5	33.1/**48.9**	33.7/46.0			
DE (→EN)	21.1/33.3	20.1/33.6	12.8/22.5	21.5/38.0	**26.3**/**40.7**	34.4	8.6	41.5
RU (→EN)	19.2/33.2	20.4/34.9	19.3/34.4	**24.8**/**41.9**	23.2/33.3	32.9	15.2	47.3
RU (→AR)	21.1/41.1	22.2/42.1	11.6/21.4	28.9/50.4	**30.3**/**52.0**			

Table 7.3: Directed and Undirected (DA/UA) accuracies (without punctuation) of our models as compared to left (L) and right (R) branching baselines and their undirected accuracies (Un). Our results show an intriguing effect of the target language on grammar induction. Note the attachment accuracy discrepancy between translating Russian to English versus Arabic.

Successfully extracting linguistic structure with hard attention indicates that models can capture interesting structures beyond semantic co-occurrence via discrete actions. This corroborates previous work (Choi et al., 2017; Yogatama et al., 2017) which has shown that non-trivial structures are learned by using REINFORCE (Williams, 1992) or the Gumbel-softmax trick (Jang et al., 2016) to backprop through discrete units. Our approach also outperforms that of Hashimoto and Tsuruoka (2017) despite our model lacking access to additional resources like part-of-speech tags. Although the numbers are not directly comparable since they use WSJ corpus to evaluate the UA score.

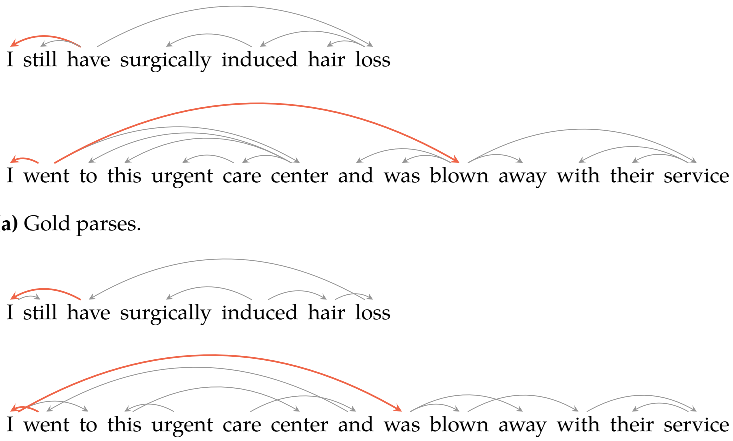
Dependency Accuracies While the SA-NMT-hard model gives the best directed attachment scores on German, English and Russian, the BLEU scores of this model are below other SA-NMT models as shown in Table 7.2. The lack of correlation between syntactic performance and NMT contradicts the intuition of previous work and suggests that useful structures learned in the context of a task might not necessarily benefit from or correspond directly to known linguistic formalisms. We want to raise three important differences between these induced structures and UD.

First, we see there is a blurred boundary between dependency and constituency representations. As noted briefly earlier, the BiLSTM provides a local summary. When the model chooses a head word, it is actually choosing hidden states from a BiLSTM and therefore gaining access to a constituent or region. This means there is likely little difference between attending to the noun vs the determiner in a phrase (despite being wrong according to UD). Future work might be able to force this distinction by replacing the BiLSTM with a bag-of-words but this will likely lead to substantial losses in NMT performance.

Second, because the model appears to be using syntax for agreement, often verb dependencies link to subjects directly to capture predicate argument structures like those in combinatory categorial grammar or semantic role labeling. UD instead follows the convention of attaching all verbs that share a subject to one another or their conjunctions. We have colored some subject–verb links in Figure 7.6. For example, we see links between *I* and both *went* and *was*.

Finally, headedness and disconnectedness. The model's notion of headedness is atypical as it roughly translates to "helpful when translating". The head word gets incorporated into the the shared representation which may cause the arrow to flip from traditional formalisms. Additionally, because the model can turn on and off syntax as necessary, it is likely to produce high confidence treelets rather than complete parses. This means arcs produced from words with weak gate activations (Figure 7.4) are not actually used during translation and likely not-syntactically meaningful.

We will not take a stance here on whether these are desirable properties or issues to address with constraints, but the model's decisions appear well motivated and our formulation allows us to have the discussion.


I still have surgically induced hair loss
I went to this urgent care center and was blown away with their service
I still have surgically induced hair loss
I went to this urgent care center and was blown away with their service

(a) Gold parses.

(b) SA-NMT (shared)

Figure 7.6: Samples of induced trees for English by our (En→Ru) model. Notice the red arrows from *subject↔verb* which are necessary for translating Russian verbs.

7.7 Conclusions

In this last research chapter, we investigated a neural latent variable model that can jointly infer a latent dependency structure of the source sentence and maximize translation performance. Specifically, we have answered **Research Question 5**:

RQ5 *What type of model can learn to induce linguistic structure and utilize those learnt structures to improve task-specific performance?*

Searching over the space of possible structures is intractable. Therefore, we limit our search space by working with a specific type of structures, namely dependency structure. We have chosen machine translation as the task of interest to study in this chapter. This leads us to an answer to the first sub-question:

RQ5.1 *Can we design a translation model that learns to induce a dependency tree-like representation of the source sentence?*

We have proposed such a design of neural architectures, namely structured-attention NMT. Our models consist of a structured attention encoder that learns to induce a dependency tree-like representation of the source sentence. The models presented here do not access any external information such as parse-trees or part-of-speech tags yet appear to use and induce structure when given the opportunity. In addition to learning latent trees of the source sentences, we showed that our models achieve significant gains in performance over a strong baseline on standard WMT benchmarks.

By design, our models allow us to extract dependency trees that are useful for translation. This lead us to an answer to the second sub-question:

RQ5.2 *Are the dependency trees learnt with the translation objective recognizable from a linguistic perspective?*

We evaluated the quality of induced trees on Universal Dependency datasets and showed that our models outperform branching baselines for English and Russian. Our grammar induction results indicate that the induced trees by the models overlap with undirected UD trees by 40-52%. A large percentage of arcs in the induced trees is not recognizable from dependency grammar. This implies that not all syntactic dependencies of the source sentence are needed when optimizing the translation objective. Interestingly, we see our induction performance is language pair dependent, which invites an interesting research discussion as to the role of syntax in translation and the importance of working with morphologically rich languages.

Having the results of grammar induction and translation, we provide an answer to the last sub-question:

RQ5.3 *Is there any correlation between the interpretability of the learned structure and the translation quality?*

Our results showed that while the hard structured attention models give the best UA score on the UD dataset, they do not outperform the baselines in translation. The lack of correlation between the quality of induced trees and the quality of translation indicates that latent trees learned with respect to a task might not be recognizable from a linguistic perspective.

Our results corroborate previous findings (Hashimoto and Tsuruoka, 2017). We agree and provide stronger evidence that syntactic information can be discovered via latent structured self attention, but we also present preliminary results that indicate that conventional definitions of syntax may be at odds with task specific performance.

Chapter 8

Conclusions

In this final chapter, we summarize the main findings of this thesis and discuss possible directions for future research.

8.1 Main Findings

This thesis focuses on incorporating linguistic knowledge into NLP systems as well as understanding the capability of neural networks in capturing linguistic phenomena.

We use neural networks as the main workhorse of this thesis for *predicting*, *analyzing*, and *discovering* structure in languages. In the following, we revisit the five main research questions asked in the thesis and summarize our main findings for each question.

Research question 1: *Do neural networks offer modeling advantages for linguistic structure prediction in comparison to non-neural methods?*

We address this question in the context of machine translation where the target languages are morphologically rich. We proposed bilingual neural network architectures that utilize morphological knowledge of the target languages to make accurate predictions. The morphological knowledge given to neural networks is obtained from supervised or unsupervised morphological segmentation tools (Chapter 2). Neural networks can also utilize soft morphological representations to make context-sensitive predictions (Chapter 3). We compared distributed representations used by neural networks with one-hot encodings typically used in log-linear models and showed the superior performance of neural network models. We integrated our neural network models into an in-house phrase-based machine translation system and demonstrated significant gains in translation quality.

Research question 2: *From a linguistic perspective, what makes recurrent neural networks work so well for language modeling?*

To answer this question, we proposed recurrent memory networks, a set of recurrent neural networks with augmented memory. The memory addressing mechanism in our models allows us to perform precise linguistic analyses when the models make their predictions. Evaluating our networks for language modeling tasks, we found that our models capture important co-occurrences and dependency types that are essential for predicting the next word. In addition to being more interpretable, we evaluated our models on the Sentence Completion Challenge dataset and obtained a new state-of-the-art result.

Research question 3: *Do non-recurrent neural networks have the same ability to exploit hierarchical structures implicitly in comparison to their recurrent counterpart?*

We evaluated the performance of LSTMs and Transformers (non-recurrent architectures) on subject-verb agreement and logical inference tasks. The subject-verb agreement task is chosen to test the ability of the models to infer hierarchical structure in unstructured text. The logical inference task is chosen to test the ability of the model to exploit hierarchical structure in tree-structured text. The results of our experiments on both tasks showed that LSTMs slightly but consistently outperform Transformers. Our findings suggested that recurrency is important for modeling hierarchical structure implicitly.

Research question 4: *Can neural networks be used to induce linguistic structure in a completely unsupervised manner?*

We investigated this question in the context of part-of-speech induction. We parametrized an unsupervised Hidden Markov Model with neural networks. The transition and emission probabilities are produced by an LSTM and a character-CNN. We provided Expectation-Maximization training algorithm for our model. We showed that our proposed framework enables seamless integration of linguistic features. We evaluated our framework on a part-of-speech induction benchmark and obtained state-of-the-art results within generative approaches.

Research question 5: *What type of model can learn to induce linguistic structure and utilize those learnt structures to improve task-specific performance?*

We provided an answer to this question in the context of neural machine translation. We proposed a class of structured attention augmented NMT models that perform translation while simultaneously inducing dependency structures of the source sentences. Our models achieved substantial gains in translation performance and outperformed branching baselines in grammar induction. We noticed that grammar induction performance is correlated with the richness of the target language morphology.

Additionally, our experimental results suggest that the models prefer to use syntactic structure when given the opportunity. However, the kind of structure learned by the models for the purpose of a task might be different from known dependency structures identified by linguists.

8.2 Future Work

Based on the answers provided by this thesis to our five research questions, we identify three possible directions for future research:

1. *Investigating a neural architecture that benefits from both recurrent connections and attention mechanisms.*

   In Chapter 5, we showed that recurrency is important for capturing hierarchical structure. The self-attention mechanism offers direct access to long distance input tokens however it increases the computational cost linearly with the size of the context (*i.e.*, number of tokens in the past). While in Chapter 4, we proposed a simple class of models that combine recurrency and attention, we believe that the future generation of neural networks could benefit from both recurrency and attention in a more elegant way such as using fast weights (Ba et al., 2016a) or read-write mechanisms (Graves et al., 2014). Additionally, the evaluation of such models should be done on more realistic and challenging language tasks, such as translating into morphologically rich languages or document level MT.

2. *Exploring alternative structural constraints that can be learned and exploited effectively by the models.*

   In Chapter 7, we focused on the dependency structure of the source and have not explored different kinds of structures such as constituent structures. Having the right structural constraints can introduce positive bias to the models and reduce the search space of all possible structures. Future research should investigate what kind of structural constraints are needed for a given language task and whether that structural constraint can be learned efficiently. Moreover, we have not yet investigated the type of structure that can be induced *incrementally* by the decoder in seq2seq models. The incrementality property is inspired by a psycholinguistic model of the human parser that allows the model to build up a partial tree structure from a partially observed sequence of words. This opens up an avenue for future exploration in language modeling and translation.

3. *Inducing grammars with and for multilingual neural machine translation.*

   In Chapter 7, we evaluated our model for grammar induction and translation performance using bilingual corpora. Future research should investigate the model's performance and its learned representation using multilingual data. Previous work (Johnson et al., 2017; Lee et al., 2017; Firat et al., 2016) has demonstrated that NMT can map multiple languages into a shared space of representation. By extending our work to multilingual NMT, we can investigate the question of whether a shared structure exists amongst languages. If there is such a shared structure, we would like to analyze it from a linguistic perspective and investigate if it helps improving translation by transferring knowledge across languages.

Bibliography

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249, Denver, Colorado. Association for Computational Linguistics.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. In *Proceedings of Evalita'09, Evaluation of NLP and Speech Tools for Italian*, Reggio Emilia, Italy.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA.

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 763–770, Columbus, Ohio. Association for Computational Linguistics.

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. 2016a. Using fast weights to attend to the recent past. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4338–4346, USA. Curran Associates Inc.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016b. Layer normalization. *ArXiv e-prints*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, San Diego, CA, USA.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1947–1957. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Yoshua Bengio, Renato De Mori, Flammia Giovanni, and Ralf Kompe. 1991. Global optimization of a neural network - Hidden Markov Model hybrid. In *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Transaction on Neural Networks*, 5(2):157–166.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden.

Arianna Bisazza and Christof Monz. 2014. Class-based language modeling for translating into morphologically rich languages. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1918–1927, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus Interpolation Methods for Phrase-based SMT Adaptation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143, San Francisco, CA.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Yonatan Bisk, Christos Christodoulopoulos, and Julia Hockenmaier. 2015. Labeled grammar induction with minimal supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 870–876, Beijing, China.

Yonatan Bisk and Julia Hockenmaier. 2013. An hdp model for inducing combinatory categorial grammars. *Transactions of the Association for Computational Linguistics*, pages 75–88.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Phil Blunsom and Trevor Cohn. 2011. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA. Association for Computational Linguistics.

Victor Bocharov, Svetlana Alexeeva, Dmitry Granovsky, Ekaterina Protopopova, Maria Stepanova, and Alexey Surikov. 2013. Crowdsourcing morphological annotation. In *Proceedings of the International Conference "Dialogue"*, Bekasovo, Russia.

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

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. 2012. The joy of parallelism with czeng 1.0. In *Proceedings of LREC2012*, Istanbul, Turkey. ELRA, European Language Resources Association.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of Proceedings of the NIPS 2015 Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*.

Peter F Brown, Peter V deSouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 2–17, Lake Tahoe, California.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1677–1687, Seattle, USA.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 4(13):359–393.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1975–1985, Berlin, Germany. Association for Computational Linguistics.

Colin Cherry, Robert C. Moore, and Chris Quirk. 2012. On hierarchical reordering and permutation parsing for phrase-based decoding. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 200–209, Montréal, Canada. Association for Computational Linguistics.

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

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2017. Learning to compose task-specific tree structures. *AAAI*.

Morten H. Christiansen and Nick Chater. 2016. The now-or-never bottleneck: A fundamental constraint on language. *Behavioral and Brain Sciences*, 39.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two Decades of Unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2011. A Bayesian Mixture Model for Part-of-Speech Induction Using Multiple Features. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 59–66, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th Annual International Conference on Machine Learning*, volume 12, pages 2493–2537.

Hannah Cornish, Rick Dale, Simon Kirby, and Morten H Christiansen. 2017. Sequence memory constraints give rise to language-like structure through iterated learning. *PloS one*, 12(1):e0168532.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2933–2941, Cambridge, MA, USA. MIT Press.

A Dempster, N Laird, and D Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27, Vancouver. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 678–683, Sofia, Bulgaria.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013. Model with minimal translation units, but decode with phrases. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Atlanta, Georgia, USA.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*

Processing, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum Branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78. Association for Computational Linguistics.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M. Bender. 2017. Towards linguistically generalizable NLP systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 1–10, Copenhagen, Denmark. Association for Computational Linguistics.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Kilian A. Foth. 2006. *Eine umfassende Constraint-Dependenz-Grammatik des Deutschen*. Fachbereich Informatik.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in smt. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, Avignon, France. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA. Curran Associates Inc.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 847–855, Honolulu, Hawaii. Association for Computational Linguistics.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the*

52nd Annual Meeting of the Association for Computational Linguistics, pages 699–709. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Kevin Gimpel and Noah A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 9–17, Columbus, Ohio, USA.

Sharon Goldwater, Mark Johnson, and Thomas L. Griffiths. 2006. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *CoRR*, abs/1410.5401.

Spence Green and John DeNero. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 146–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1462–1471.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*.

Rejwanul Haque, Sudip Kumar Naskar, Antal Bosch, and Andy Way. 2011. Integrating source-language context into phrase-based statistical machine translation. *Machine Translation*, 25(3):239–285.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 125–135. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ArXiv e-prints*.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 190–198. Curran Associates, Inc.

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

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29, Gothenburg, Sweden. Association for Computational Linguistics.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *ArXiv e-prints*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2011–2021, Copenhagen, Denmark. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't em find good hmm pos-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic. Association for Computational Linguistics.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. 2016. Composing graphical models with neural networks for structured representations and fast inference. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2946–2954. Curran Associates, Inc.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *ArXiv e-prints*.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, USA.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2016. Grid long short-term memory. In *International Conference on Learning Representations*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *International Conference on Learning Representations*.

Ahmed El Kholy and Nizar Habash. 2012. Translate, predict or generate: Modeling rich morphology in statistical machine translation. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *International Conference on Learning Representations*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, San Diego, CA, USA.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of* *the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translations in the Americas (AMTA 2004)*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Philipp Koehn. 2010. *Statistical Machine Translation*, 1st edition. Cambridge University Press, New York, NY, USA.

Philipp Koehn, Abhishek Arun, and Hieu Hoang. 2008. Towards better machine translation quality for the German-English language pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 139–142, Columbus, Ohio. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, J Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of Proceedings of ICASSP*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–9, Portland, Oregon, USA. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697. Association for Computational Linguistics.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1311–1316, Denver, Colorado.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *International Conference on Learning Representations*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Henrik Dittmann, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISÀ corpus of Italian web texts. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pages 36–43, Gothenburg, Sweden. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Mitchell P Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *ARPA Human Language Technology Workshop*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 210–218, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

George A. Miller. 1995. Wordnet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 128–135.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 511–517, Beijing, China. Association for Computational Linguistics.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, New York, NY, USA.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1751–1758, New York, NY, USA.

Thomas Müller, Hinrich Schütze, and Helmut Schmid. 2012. A comparative investigation of morphological language modeling for the languages of the European Union. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 386–395, Montréal, Canada. Association for Computational Linguistics.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley-Interscience.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the*

40th Annual Meeting of the Association of Computational Linguistics (ACL), pages 311–318, Philadelphia, PA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1310–1318.

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

Vu Pham, Christopher Bluche, Théodore Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, pages 172–181.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? challenging the conventional wisdom in statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 9–16, New York City, USA. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Optimization with em and expectation-conjugate-gradient. In *Proceedings, Intl. Conf. on Machine Learning (ICML*, pages 672–679.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, Manchester, UK.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING*.

Holger Schwenk, Daniel Dechelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Conference*, pages 723–730, Sydney, Australia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for german dependency parsing, pos-tagging, and morphological analysis. In *Recent Advances in Natural Language Processing (RANLP 2013)*, pages 601–609.

Serge Sharoff, Mikhail Kopotev, Tomaz Erjavec, Anna Feldman, and Dagmar Divjak. 2008. Designing and evaluating a Russian tagset. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Maria Simi, Cristina Bosco, and Simonetta Montemagni. 2014. Less is more? towards a reduced inventory of categories for training a parser for the Italian Stanford dependencies. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Noah A. Smith. 2011. *Linguistic Structure Prediction*, 1st edition. Morgan & Claypool Publishers.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical*

Methods in Natural Language Processing, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kai Song, Yue Zhang, Min Zhang, and Weihua Luo. 2018. Improved English to Russian translation by neural suffix prediction. In *AAAI*.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1840–1845, Doha, Qatar.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 67–74, Prague, Czech Republic. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Michael Subotin. 2011. An exponential translation model for target language morphology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Portland, Oregon, USA. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2431–2439. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350, Austin, Texas. Association for Computational Linguistics.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, pages 514–522, Columbus, Ohio. Association for Computational Linguistics.

Ke Tran, Arianna Bisazza, and Christof Monz. 2015. A distributed inflection model for translating into morphologically rich languages. In *Proceedings of the 15th Machine Translation Summit (MT-Summit 2015)*, pages 145–159, Miama, USA.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016a. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.

Ke Tran and Yonatan Bisk. 2018. Inducing grammars with and for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 25–35, Melbourne, Australia. Association for Computational Linguistics.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2014. Word translation prediction for morphologically rich languages with bilingual neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1676–1688, Doha, Qatar. Association for Computational Linguistics.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016b. Unsupervised Neural Hidden Markov Models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX. Association for Computational Linguistics.

W. T Tutte. 1984. *Graph theory*. Cambridge University Press.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, pages 755–762, Columbus, Ohio. Association for Computational Linguistics.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle.

Lloyd R Welch. 2003. Hidden Markov Models and the Baum-Welch Algorithm. *IEEE Information Theory Society Newsletter*, 53(4):1–24.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951, Jeju Island, Korea. Association for Computational Linguistics.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. *International Conference on Learning Representations*.

Eros Zanchetta and Marco Baroni. 2005. Morph-it! a free corpus-based morphological resource for the Italian language. *Corpus Linguistics 2005*, 1(1).

Jian Zhang, Ioannis Mitliagkas, and Christopher Ré. 2017. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, USA.

Geoffrey Zweig and Chris J. C. Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, WLM '12, pages 29–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Summary

In the field of natural language processing (NLP), recent research has shown that deep neural network models are quite brittle and can only model linguistic principles to a limited degree. In order to build NLP systems that can generalize and work well in practice, it is important to integrate linguistic knowledge into such systems as well as investigate the ability of current models in capturing linguistic phenomena. This thesis attempts to address those two aspects from four different angles.

First, this thesis demonstrates that neural network models enable the integration of morphological knowledge seamlessly into phrase-based machine translation systems without any feature engineering.

Second, this thesis investigates what linguistic phenomena are implicitly captured by recurrent neural networks by augmenting them with an external memory. This thesis also studies the impact of recurrent vs non-recurrent architectures in modeling hierarchical structure.

Third, while neural networks are well known to be powerful supervised learners, this thesis investigates whether they offer the same benefits for unsupervised structure learning. This thesis proposes an unsupervised Neural Hidden Markov Model for the purpose of part-of-speech induction.

Finally, this thesis asks whether neural networks can induce meaningful structure from non-annotated text. This thesis proposes structured attention models that induce a dependency-like tree representation of the input sentence for the purpose of translation. Moreover, this thesis shows that the models learn some basic elements of the source language grammar.

Samenvatting

Recent onderzoek op het gebied van Natural Language Processing (NLP) heeft aangetoond dat diepe neurale netwerken behoorlijk broos zijn en linguïstische principes slechts in beperkte mate kunnen modeleren. Om NLP-systemen te kunnen bouwen die goed kunnen generaliseren en die goed in de praktijk werken, is het belangrijk om linguïstische kennis in dergelijke system te integreren. Bovendien is het van belang om huidige systemen te onderzoeken op hun vermogen om linguïstische verschijnselen te vatten. In dit proefschrift worden deze twee aspecten vanuit vier verschillende invalshoeken benaderd.

Ten eerste wordt in dit proefschrift aangetoond dat neurale netwerken de integratie van morfologische kennis in phrase-based machine translation systemen mogelijk maken, zonder enige vorm van feature engineering.

Ten tweede wordt in dit proefschrift onderzocht welke linguïstische verschijnselen door recurrent neural networks impliciet worden vastgelegd. Dit wordt onderzocht door deze netwerken te verrijken met een extern geheugen. In dit proefschrift wordt ook de impact van recurrent versus non-recurrent architecturen op het modeleren van hiërarchische structuren bekeken.

Ten derde, waar neurale netwerken bekend staan goed te kunnen leren in een supervised setting, wordt in dit proefschrift ook de unsupervised setting bekeken. In dit proefschrift wordt een unsupervised Neural Hidden Markov Model aangedragen voor part-of-speech inductie.

Ten slotte wordt in dit proefschrift ook de vraag gesteld of neurale netwerken betekenisvolle structuren kunnen afleiden uit niet geannoteerde tekst. In dit proefschrift worden gestructureerde attention modellen voorgesteld die een dependency-achtige tree representatie van de inputzin afleiden die gebruikt kan worden voor machine translation. In dit proefschrift wordt ook aangetoond dat de modellen bepaalde basiselementen van de grammatica van de brontaal leren.