Advancing Neural Language Modeling in Automatic Speech Recognition

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Kazuki Irie,
Diplôme d'Ingénieur,
M.S. Applied Mathematics
aus Kagawa, Japan

Berichter: Univ.-Prof. Dr.-Ing. Hermann Ney
Prof. em. Dr. Renato De Mori

Tag der mündlichen Prüfung: 5. Mai 2020

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Eidesstattliche Erklärung

Ich, Kazuki Irie,
erklärt hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;
2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;
3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;
4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;
5. Alle wesentlichen Quellen von Unterstützung wurden benannt;
6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;
7. Teile dieser Arbeit wurden zuvor veröffentlicht (Details in Kapitel 9).

Aachen, den 5. Mai 2020

Kazuki Irie

Acknowledgments

This thesis would have never been possible without help and support from many people. In particular, I would like to thank the following people.

First and foremost, I would like to thank my advisor Prof. Dr.-Ing. Hermann Ney. I am very fortunate to have been his student. He taught me the fundamental manners of doing research in language modeling in speech recognition, while supporting me to develop my own ideas. I always found his words to have a power to unlock our potential, by motivating us to always try a bit harder than the best we can do. He also allowed me to travel a lot to conferences, and he taught me how to interact with people in a research community. He showed me both how we should criticize and appreciate research works. He also taught me how to teach and collaborate with students. Thank you for this long training and the trust over this long period.

Second, I would like to express my gratitude to Prof. Dr. Renato De Mori who has kindly accepted to be the second referee of this thesis. In particular, I would like to thank him for his interest in my work, since the very first time we met in May 2019, while he was visiting Aachen. It has been extremely motivating for me to hear his supportive feedback on my work.

I also would like to thank Priv.-Doz. Dr. Ralf Schüter for suggestions and proof-reading of my research publications over the last years, even when my request was at the last minute. Thank you for having always taken care of this crucial step for the quality of our papers. I also thank him for many pieces of practical advice in teaching.

From my current and previous colleagues at i6, I would like to first thank Martin Sundermeyer who had developed his excellent LSTM language modeling know-how and the software at i6, ahead of his time. Taking over his setups allowed me to work with strong baselines directly from the beginning of my thesis.

I also would like to specially thank Pavel Golik and Zoltán Tüske, who have been my big brothers in the same office for over three years. From tuning of neural networks to research in speech in general, including many other random conversations, you taught me really a lot. I also thank Tamer Alkhouli for many research, teaching, and other conversations. I also learned a lot from you. Finally, I deeply thank Albert Zeyer with whom I had a chance to do many joint works and travels together in the last six years. I thank him for his open-minded thinking which allowed me to discuss and talk about everything, in research, in programming, or any other things in general. I hope we will be able to do further joint work in the near future again.

I am also grateful to some of my colleagues who helped me to directly improve the quality of this thesis. I would like to thank Yingbo (Ringo) Gao, Yunsu Kim, Parnia Bahar, Tamer Alkhouli, Pavel Golik, and Albert Zeyer for having critically proof-read this thesis and patiently suggested corrections. Also independent of proof-reading, I have always enjoyed discussing ideas with all of you. I also would like to thank Ralf Schlüter for suggestions for some section titles. In addition, I would like to thank Alexander Gerstenberger, Albert Zeyer and Pavel Golik for corrections and suggestions on the German abstract.

Further thanks go to my current and previous colleagues during my time at i6: Albert, Amr, Andy, Basha, three Christian, two Christoph, David, Eugen, Farzad, Harald, Ilya, Jan, Jens, Jörn, Jan-Thorsten, Julian, Mahdi, three Markus, Martin, Michal, Mingwei, Mirko, Mohammad, Muhammad, Nick, Oscar, Parnia, Patrick, Pavel, Peter, Ringo, Saab, Simon, Stefan, Tamer, Tobias, Volker, Wei, Weiyue, Willi, Yunsu, and Zoltán, as well as temporal colleagues: Adrià, Dewi, Guillaume, Javier, Karel, Pau, and Pema, and finally Bachelor and Master students with whom I had a chance to work: Rami, Zhihong, Liuhui, Zijian, Arne, and Alexander. Thank you for many teamworks and nice time we shared together.

I further would like to thank our system administrators Stefan, Thomas, Kai, Jan-Thorsten, Pavel, Jan, Eugen, Weiyue and Christoph for their hard work in maintaining the infrastructure. In particular, I must thank Pavel and Christoph who have helped me scheduling my urgent jobs before deadlines, by accepting my help requests even during weekends. I also would like to thank Steffi, Andrea, Dhenya, and Anna for their administrative helps.

During my Ph.D. studies, I had opportunities to do two very fortunate internships at Google. I first would like to deeply thank my hosts and main collaborators, Shankar Kumar, Hank Liao, and Michael Nirschl for the first internship in NYC, and Rohit Prabhavalkar, Anjuli Kannan, Patrick Nguyen, Antoine (Tony) Bruguier, and David Rybach for the second internship in Mountain View. I was very fortunate to meet and interact with many people during the internships, and I would like to thank them for their warm welcome and for many research discussions, in particular: Tara Sainath, Michiel Bacchiani, Hasim Sak, Hagen Soltau, Ke (Kevin) Hu, Richard (Rick) Rose, Olivier Siohan, Takaki Makino, Golan Pundak, Hadrien Gelas, Pierric Sans, Michael Riley, Ke Wu, Hao Zhang, Ananda Theertha Suresh, Brian Roark, Ruoming Pang, Arun Narayanan, Yanzhang (Ryan) He, Bo Li, Khe Chai Sim, Ananya Misra, Mitchel (Mitch) Weintraub, Vijayaditya (Vijay) Peddinti, Erik McDermott, Ehsan Variani, Tom Bagby, Matt Shannon, Anshuman Tripathi, Han Lu, Stephen Koo, Kenny Leung, Qian Zhang, Joe Caroselli, Yu Zhang, William Chan, Yonghui Wu, Zhifeng Chen, Heiga Zen, and Yuxuan Wang. I would like to thank Shankar once again, because all my wonderful experiences and meetings at Google have started when I met Shankar at Interspeech 2016 in San Francisco.

Furthermore, my education and learning in speech recognition during this thesis was stimulated by many fortunate meetings and interactions with other people in the speech research community. I would like to thank in particular: Andreas Stolcke, Andros Tjandra, Bhuvana Ramabhadran, Gakuto Kurata, Ilya Oparin, Jahn Heymann, Jinyu Li, Joris Pelemans, Julian Chan, Kyu Han, Marc Delcroix, Michael Picheny, Prof. Reinhold Häb-Umbach, Prof. Satoshi Nakamura, Prof. Shinji Watanabe, Shigeki Karita, Shubham Toshniwal, Tomohiro Nakatani, Wei-Ning Hsu, Xie (Jeff) Chen, Yotaro Kubo, Siva Reddy Gangireddy, and Thiago Fraga da Silva.

Finally, I would like to thank my family!

Abstract

Statistical language modeling is one of the fundamental problems in natural language processing. In the recent years, language modeling has seen great advances by active research and engineering efforts in applying artificial neural networks, especially those which are recurrent. The application of neural language models to speech recognition has now become well established and ubiquitous. Despite this impression of some degree of maturity, we claim that the full potential of the neural network based language modeling is yet to be explored. In this thesis, we further advance neural language modeling in automatic speech recognition, by investigating a number of new perspectives.

From the architectural view point, we investigate the newly proposed Transformer neural networks for language modeling application. The original model architecture proposed for machine translation is studied and modified to accommodate the specific task of language modeling. Particularly deep models with about one hundred layers are developed. We present an in-depth comparison with the state-of-the-art recurrent neural network language models based on the long short-term memory.

While scaling up language modeling to larger scale datasets, the diversity of the data emerges as an opportunity and a challenge. The current state-of-the-art neural language modeling lacks a mechanism of handling diverse data from different domains for a single model to perform well across different domains. In this context, we introduce domain robust language modeling with neural networks, and propose two solutions. As a first solution, we propose a new type of adaptive mixture of experts model which is fully based on neural networks. In the second approach, we investigate knowledge distillation from multiple domain expert models, as a solution to the large model size problem seen in the first approach. Methods for practical applications of knowledge distillation to large vocabulary language modeling are proposed, and studied to a large extent.

Finally, we investigate the potential of neural language models to leverage long-span cross-sentence contexts for cross-utterance speech recognition. The appropriate training method for such a scenario is under-explored in the existing works. We carry out systematic comparisons of the training methods, allowing us to achieve improvements in cross-utterance speech recognition. In the same context, we study the sequence length robustness for both recurrent neural networks based on the long short-term memory and Transformers, because such a robustness is one of the fundamental properties we wish to have, in neural networks with the ability to handle variable length contexts.

Throughout the thesis, we tackle these problems through novel perspectives of neural language modeling, while keeping the traditional spirit of language modeling in speech recognition.

Kurzfassung

Die statistische Sprachmodellierung ist eines der grundlegenden Probleme bei der maschinellen Verarbeitung natürlicher Sprache. In den letzten Jahren hat die Sprachmodellierung große Fortschritte gemacht, durch aktiven Arbeitsaufwand bei der Anwendung künstlicher neuronaler Netzwerke, insbesondere der rekurrenten Netzwerke. Die Anwendung neuronaler Sprachmodelle auf die Spracherkennung ist inzwischen gut etabliert und allgegenwärtig. Dennoch argumentieren wir, dass das volle Potenzial der neuronalen Netzwerken basierenden Sprachmodellierung noch nicht ausgeschöpft ist. In dieser Arbeit entwickeln wir die neuronale Sprachmodellierung in der automatischen Spracherkennung weiter. Dazu untersuchen wir eine Reihe neuer Perspektiven.

Wir untersuchen die neu vorgeschlagenen Transformer-Modelle für die Anwendung in der Sprachmodellierung. Die für die maschinelle Übersetzung ursprüngliche vorgeschlagene Transformer-Modellarchitektur wird untersucht und an die spezifischen Anforderungen der Sprachmodellierung angepasst. Sehr tiefe Modelle mit etwa hundert Schichten werden entwickelt. Wir führen einen detaillierten Vergleich mit den Long-Short-Term-Memory basierten Sprachmodellen.

Bei der Skalierung der Sprachmodellierung auf größere Datensätze erscheint die Vielfalt der Daten als Chance und Herausforderung. Der aktuellen besten neuronalen Sprachmodellierung fehlt ein Mechanismus zur Handhabung unterschiedlicher Daten aus verschiedenen Domänen, damit ein einziges Modell in verschiedenen Domänen gut funktioniert. In diesem Zusammenhang stellen wir eine domänenrobuste Sprachmodellierung mit neuronalen Netzwerken vor. Wir stellen zwei Lösungen vor. Als erste Lösung schlagen wir eine neue Art von adaptiver Mixture-of-Experts Modellen vor, die vollständig auf neuronalen Netzwerken basieren. Dieser Ansatz hat einen Nachteil der sperrigen Modellgröße. Im zweiten Ansatz untersuchen wir daher die Knowledge-Distillation aus Expertenmodellen mit mehreren Domänen. Methoden zur praktischen Anwendung der Knowledge-Distillation auf die Sprachmodellierung mit großem Vokabular werden vorgeschlagen und ausführlich untersucht.

Schließlich untersuchen wir das Potenzial neuronaler Sprachmodelle zur Nutzung von langen satzübergreifenden Kontexten für verbesserte Spracherkennung. Die geeignete Trainingsmethode für ein solches Szenario ist in den existierenden Arbeiten noch nicht ausreichend erforscht. Wir führen einen systematischen Vergleich der Trainingsmethoden durch, wodurch wir Verbesserungen bei der satzübergreifenden Spracherkennung erzielen. Im gleichen Zusammenhang untersuchen wir die Robustheit verschiedener Sequenzlängen sowohl für rekurrente Long-Short-Term-Memory neuronale Netzwerke als auch für Transformer-Modelle. Eine solche Robustheit ist eine der grundlegenden Eigenschaften, die wir uns in neuronalen Netzwerken mit der Fähigkeit zur Handhabung von Kontexten variabler Länge wünschen.

In der gesamten Arbeit gehen wir diese Themen mit neuen Perspektiven der neuronalen Sprachmodellierung an, wobei wir die traditionelle Weise der Sprachmodellierung in der automatischen Spracherkennung beibehalten.

Contents

1 Introduction **1**
1.1 Statistical Language Modeling . . . 1
1.1.1 Definition . . . 1
1.1.2 Perplexity . . . 2
1.1.3 N-gram Count Models . . . 3
1.1.4 Neural Language Models . . . 5
1.2 Automatic Speech Recognition . . . 6
1.2.1 Conventional HMM based Automatic Speech Recognition . . . 7
1.2.2 Lattice Rescoring in Two-Pass Speech Recognition . . . 8
1.2.3 End-to-end Speech Recognition with Encoder-Decoder Models . . . 11

2 Scientific Goals **13**

3 Basic Concepts and Developments of Neural Language Modeling **17**
3.1 State-of-the-art LSTM-RNN Language Models . . . 18
3.1.1 Standard Architecture . . . 18
3.1.2 Improvements by Larger and Well Regularized Models . . . 19
3.1.3 Training and Evaluation Sequence Construction . . . 21
3.1.4 Domain Adaptation . . . 22
3.1.5 Modeling Units . . . 23
3.1.6 A Brief Detour into Highway Connections . . . 24
3.2 Attention in Language Modeling: Shallow Attempts . . . 28
3.2.1 Attention . . . 28
3.2.2 Bag-of-Words . . . 28
3.2.3 Attention for Learning Word Triggers . . . 31
3.3 Correlation Between Perplexity and Word Error Rate . . . 34
3.3.1 Corpus level Correlation Using Multiple Models . . . 34
3.3.2 Local Correlation Using One Model . . . 36
3.4 Summary . . . 36

4 State-of-the-art ASR Language Modeling with Transformers **39**
4.1 Deep Transformers for Language Modeling . . . 40
4.1.1 Transformer Language Models . . . 40
4.1.2 Tuning Hyper-Parameters in Transformers . . . 42
4.1.3 Residual vs. Highway Connection . . . 45
4.1.4 Parameter Tying . . . 46
4.1.5 ASR Experiments . . . 47
4.1.6 Conclusion . . . 48

4.2 Analysis for Better Understanding Transformer Language Models . . . 49
4.2.1 Transformer Language Models Without Positional Encoding . . . 49
4.2.2 Identifying 4 Functional Groups of Layers . . . 50
4.3 Alternative Architecture for More Memory Efficient Search . . . 54
4.3.1 Transformer Language Model with Reduced State Size . . . 54
4.3.2 Experimental Setups . . . 56
4.3.3 Effect of DNN Inside Transformer Layer . . . 57
4.3.4 Effect of Tying Key and Value Matrices . . . 58
4.3.5 ASR Experiments . . . 59
4.3.6 Conclusion . . . 59
4.4 Comparing LSTM and Transformers Across Different Datasets . . . 60
4.4.1 Performance Overview . . . 60
4.4.2 Combination of LSTM and Transformer Language Models . . . 61
4.5 Summary . . . 62

5 Knowledge Distillation for Language Modeling 63
5.1 Knowledge Distillation for Large Vocabulary Language Models . . . 64
5.1.1 Distillation with Sampling based Losses . . . 64
5.1.2 Class based Language Modeling Case . . . 65
5.1.3 Distillation with Mean Squared Error Between Hidden States . . . 66
5.2 Application Scenarios . . . 66
5.2.1 Distillation from Transformer to LSTM . . . 66
5.2.2 Distillation from LSTM to N-gram Feed-forward Models? . . . 66
5.3 Summary . . . 71

6 Domain Robust Language Modeling 73
6.1 Recurrent Adaptive Mixture Models . . . 74
6.1.1 Recurrent Adaptive Mixture Model for Language Modeling . . . 74
6.1.2 Training Strategy . . . 75
6.1.3 YouTube Speech Recognition Dataset . . . 76
6.1.4 Effectiveness of the Mixer . . . 79
6.1.5 ASR Experiments . . . 79
6.1.6 Scaling Up Further . . . 82
6.1.7 Conclusion . . . 82
6.2 Knowledge Distillation From Domain Experts . . . 83
6.2.1 Knowledge Distillation for Domain Robust Language Modeling . . . 83
6.2.2 AppTek English Multi-domain Dataset . . . 83
6.2.3 Results for Sampled Softmax based Distillation . . . 85
6.2.4 Results for NCE based Distillation . . . 86
6.2.5 Transformer Experts for an LSTM Student . . . 87
6.2.6 ASR Experiments . . . 88
6.2.7 Conclusion . . . 88
6.3 Summary . . . 88

7 Cross-Sentence Long-Span Language Modeling 89
7.1 Cross-Sentence Language Modeling for ASR . . . 90
7.1.1 Problem Setup . . . 91
7.1.2 Training Sequence Construction Methods . . . 91
7.1.3 Experimental Setups . . . 93
7.1.4 Cross-Utterance ASR via Lattice Rescoring . . . 93

7.1.5 Text based Experiments: LSTM-RNNs . . . 95
7.1.6 ASR Experiments: LSTM-RNNs . . . 97
7.1.7 Text based Experiments: Transformers . . . 98
7.1.8 ASR Experiments: Transformers . . . 100
7.1.9 Conclusion . . . 100
7.2 Translation as Long-Span Language Modeling . . . 101
7.2.1 Task Definition . . . 101
7.2.2 Experimental Results . . . 101
7.2.3 Visualizing Functionality of Each Layer . . . 102
7.3 Summary . . . 106

8 Scientific Achievements **107**

9 Individual Contributions **109**

10 Outlook **113**

A Overview of the Corpora and Systems **115**
A.1 LibriSpeech . . . 115
A.2 TED-LIUM Release 2 . . . 116
A.3 Quaero English . . . 117
A.4 Switchboard 300 h . . . 117
A.5 AMI . . . 118
A.6 Google YouTube Dataset . . . 119
A.7 AppTek Multi-Domain Dataset . . . 119
A.8 WMT 2016 Romanian to English . . . 120

B More On the Role of Count Language Models **121**

List of Figures **123**

List of Tables **127**

Bibliography **131**

1. Introduction

Speech and written languages are the most natural means of communication for human. This has naturally given importance to the science and engineering domains which study natural language and speech processing. The *statistical language modeling* which is at the center of this thesis is one of the most elementary but fundamental tasks in the field. It has many applications in language technology, including a historical and arguably the most prominent application to automatic speech recognition.

Seen through the lens of probabilistic modeling, language modeling also has its importance as a discrete domain instance of the more generic *sequence prediction problem* of predicting the next event based on the past events. Thanks to the simplicity of handling text data, it has been treated as a good practical task within the reach of everyone for testing some new models for sequence prediction.

In this chapter, we briefly introduce the core concepts of statistical language modeling and its application to automatic speech recognition (ASR), which is the main downstream task in this thesis we are interested in measuring the progress of language modeling.

1.1 Statistical Language Modeling

1.1.1 Definition

The language model is a statistical model which estimates the probability $p(w_1^M)$ of a sequence of tokens $w_1^M := w_1, ..., w_M$ (e.g. words) where the last term w_M is the sequence end symbol. A language model is always defined with its *vocabulary* which defines the set of token classes on which the model's output distribution is normalized. Since this probability can be factorized by the chain rule of probability as follows

$$p(w_1^M) = \prod_{m=1}^{M} p(w_m|w_0^{m-1}) \text{ where } w_0 \text{ is the start symbol,} \tag{1.1}$$

the task of language modeling becomes to estimate these conditional probabilities $p(w_m|w_0^{m-1})$. In consequence, the language model is also often defined as the problem of predicting the next token w_m given its predecessor tokens w_0^{m-1}, typically referred to as *context* or *history*.

The language modeling is therefore all about exploiting the context for estimating a probability distribution of the next token. Different approaches differ from each other in how this is done. In this thesis, we are interested in the two major approaches: n-gram count based language models (*count model*) and neural network based language models (*neural language model*).

This definition of language models as a model for estimating the sequence joint probability goes back to Frederick Jelinek [Jelinek & Bahl$^+$ 75, Jelinek 76] and the explicit factorization presented in [Bahl & Jelinek$^+$ 83]. While these works have naturally introduced language models in the

context of automatic speech recognition [Jelinek & Bahl$^+$ 75], the resulting language model itself has its root in Claude Shannon's work [Shannon 48, Shannon 51].

Through a semantic change, the use of the terminology "language modeling" has been extended to usages which do not correspond to the original definition above introduced in ASR, but to other models related to some representation or ranking of language: such as *discriminative language models* [Roark & Saraclar$^+$ 04] or more recently *masked language models* [Devlin & Chang$^+$ 19] and *bi-directional language models* [Chen & Ragni$^+$ 17, Peters & Neumann$^+$ 18].

The scope of this thesis is on advancing the conventional *ASR language model* as described above, which we will therefore simply refer to as "language model" in this thesis. In the following, we introduce the evaluation measure, perplexity, as well as n-gram count based language modeling and neural language modeling.

1.1.2 Perplexity

The major evaluation metric in language modeling is the perplexity [Jelinek & Mercer$^+$ 77]. The perplexity is defined as an inverse geometric average of the conditional probabilities for each token given its predecessor context, according to the language model. Therefore, the perplexity of the sequence w_1^M for a language model $p(.|.)$ is computed as

$$\text{Perplexity} = \frac{1}{\sqrt[M]{\prod_{m=1}^{M} p(w_m|w_0^{m-1})}} = e^{-\frac{1}{M}\sum_{m=1}^{M} \log p(w_m|w_0^{m-1})} \tag{1.2}$$

where w_0 is the start symbol and log denotes the natural logarithm.

It can be qualitatively thought of as the average of *effective vocabulary size* (in other words, the average number of alternative words) the model considers at each position [Bahl & Jelinek$^+$ 83]. For instance, if the model is a uniform distribution over the vocabulary, the perplexity would be the vocabulary size for any text, which is therefore one reference number for a very bad perplexity in the given problem. Also, the log perplexity corresponds to the *cross entropy* between the model's distribution and the empirical distribution of the data.

It can also be noted that if one of the terms in the product is zero, the perplexity would be infinity independent of the values of all other terms. It is therefore preferable to design a language model which assigns a non-zero probability for any event.

While the perplexity is a text-based evaluation measure which can be computed independent of other components in the downstream speech recognition system, the ASR performance itself is finally evaluated by the word error rate (WER), defined as the Levenshtein distance [Levenshtein 66] between the recognition output and the actual transcription, divided by the number of words in the transcription. However, it has been empirically shown that there is a good correlation between the perplexity and word error rate in speech recognition [Bahl & Jelinek$^+$ 83, Chen & Beeferman$^+$ 98, Klakow & Peters 02, Sundermeyer & Ney$^+$ 15]. This correlation is further discussed and illustrated later in the preliminary Chapter 3 Sec. 3.3, and all along this thesis.

This empirical correlation therefore conveniently allows us to primarily work on language models independent of the downstream system, by considering a model with a lower perplexity to be a *better* language model in a general sense. For instance, we will be training neural network based language models by minimizing its perplexity on a training text and select the model based on its perplexity on a development text, by assuming that this selection would also result in the best model for speech recognition experiments.

1.1.3 N-gram Count Models

An n-gram language model is obtained by applying an $(n-1)$-order Markov assumption to the model's context dependency, i.e. by truncating the context in the conditional probability to its $n-1$ predecessor tokens

$$p(w_1^M) = \prod_{m=1}^{M} p(w_m|w_0^{m-1}) \overset{\text{model}}{=} \prod_{m=1}^{M} p(w_m|w_{m-n+1}^{m-1}) \tag{1.3}$$

where $w_{m-n+1}^{m-1} = w_0^{m-1}$ for $m \leq n$.

The n-gram *count based* language model (or shorter, n-gram count model) is an n-gram language model which estimates these conditional probabilities $p(w_m|w_{m-n+1}^{m-1})$ based on the relative frequencies of *event counts* (such as raw n-gram counts or other related counts) and some *smoothing techniques* to distribute probability mass to unseen n-gram events (in order to avoid zero probabilities as mentioned in the Sec 1.1.2 above), while preserving the normalization constraints necessary for a probability distribution.

Canonical formula. Any n-gram count based language model with an n-gram inventory $\mathcal{D}$ (a list of all k-grams stored in the model for $1 \leq k \leq n$)[1] can be written in the following canonical form, for each order $1 \leq k \leq n$

$$q_k(w|h) = \begin{cases} r_k(hw) & \text{if } hw \in \mathcal{D} \\ \gamma(h) q_{k-1}(w|\bar{h}) & \text{otherwise} \end{cases} \quad \text{and } p(w|h) = q_n(w|h) \tag{1.4}$$

where $\bar{h}$ denotes the shorter context where the leftmost (therefore the oldest) token[2] is removed from h. The factor $\gamma(h)$ is typically referred to as a *back-off weight*. As a side note, this formulation directly allows us to write an n-gram count based language model into the standard *ARPA format*[3] commonly used by different softwares, such as `SRILM` [Stolcke 02]. Because of this formula, these models are also referred to as *back-off language models* in the literature. In this thesis, we simply call them count models in contrast to neural language models.

Different n-gram count models therefore differ from each other in the estimation of these parameters which define k-th order estimates $r_k(w|h)$ and the back-off weights $\gamma(h)$ for all lower order k-grams h in $\mathcal{D}$ with $1 \leq k \leq n-1$, as well as the n-gram inventory $\mathcal{D}$.

For an overview of smoothing techniques for language modeling, we refer the readers to [Chen & Goodman 99]. In this thesis, we make use of the so-called *Kneser-Ney language model* as a baseline n-gram count based language models. Such a model is based on the Kneser-Ney smoothing [Kneser & Ney 95] and a couple of further specification as described in the next paragraph.

Kneser-Ney language model. In this section, we introduce the baseline count based language model which we refer to as *n-gram Kneser-Ney language model* throughout this thesis, as is commonly done in ASR publications. Simply formulated, the corresponding model is an "interpolation variant" of the Kneser-Ney smoothing [Kneser & Ney 95], which has been introduced and empirically proven to work the best among other smoothing techniques in [Chen & Goodman 99]. The Kneser-Ney smoothing combines absolute discounting [Ney & Essen 91] with back-off distribution estimation based on *diversity counts* which can naturally be derived by considering marginalization[4] constraints as introduced in [Kneser & Ney 95]. The highest order estimate is

[1] An example definition of $\mathcal{D} = \{hw \in \mathit{TrainData} \,|\, N(h,w) > 0\}$ if no pruning is applied.

[2] For example, if $h = abc$, $\bar{h} = bc$.

[3] Named so as proposed by Doug Paul at MIT Lincoln Labs for research sponsored by the U.S. Department of Defense Advanced Research Project Agency.

[4] For example, for 3-gram, the left marginalization gives: $N(v,w) = \sum_u N(u,v,w) = \sum_u p(w|u,v)N(u,v)$.

based on the *raw counts* (or simply *counts*) $N(h, w)$ which denotes the counts of event hw in the training text. For example, for a 3-gram model:

$$q_3(w|uv) = \begin{cases} r_3(uvw) & \text{if } uvw \in \mathcal{D} \\ \gamma(uv)q_2(w|v) & \text{otherwise} \end{cases} \tag{1.5}$$

$$\text{with } r_3(uvw) = \frac{\max[N(uv, w) - b_3, 0]}{N(uv, .)} + \gamma(uv)q_2(w|v) \text{ and } \gamma(uv) = b_3 \frac{N(uv, +)}{N(uv, .)} \tag{1.6}$$

where $N(uv, +)$ is the *diversity count*, number of unique 3-gram events with a prefix uv, i.e., $N(uv, +) = \sum_{a:N(uv,a)>0} 1$. Because of the second term in $r_3(uvw)$, which interpolates the lower order term $q_2(w|v)$, this is referred to as the "interpolated" variant, as opposed to the "back-off" variant which only makes use of the first term, while both approaches can be written in the canonical back-off format as above, with conditions checking whether the n-gram is in the model's inventory. Another specificity of the Kneser-Ney smoothing is that the lower order estimates are based on these diversity counts $N(+v, w) = \sum_{a:N(av,w)>0} 1$ as follows:

$$q_2(w|v) = \begin{cases} r_2(vw) & \text{if } vw \in \mathcal{D} \\ \gamma(v)q_1(w) & \text{otherwise} \end{cases} \tag{1.7}$$

$$\text{with } r_2(vw) = \frac{\max[N(+v, w) - b_2, 0]}{N(+v, .)} \text{ and } \gamma(v) = b_2 \frac{N(+v, +)}{N(+v, .)} \tag{1.8}$$

$$\text{where } N(+v, .) = \sum_{w \in \mathcal{D}} N(+v, w) \text{ and } N(+v, +) = \sum_{a:N(+v,a)>0} 1 \tag{1.9}$$

and finally

$$q_1(w) = \begin{cases} \dfrac{\max[N(+, w) - b_1, 0]}{N(+, .)} & \text{if } w \in \mathcal{D} \setminus \{\texttt{UNK}\} \\ p(\texttt{UNK}) & \text{otherwise.} \end{cases} \tag{1.10}$$

where b_1, b_2, and b_3 are the discount parameters, and $p(\texttt{UNK})$ is the unigram probability for the unknown token (set to satisfy the normalization for the unigram probability $q_1(w)$ over the vocabulary and the unknown token)[5]. The *modified* Kneser-Ney smoothing [Chen & Goodman 99] makes use of three discount parameters for each order, depending on the raw event counts (either $N(hw) = 1$, $N(hw) = 2$, or $N(hw) \geq 3$). In this thesis, we refer to these models as Kneser-Ney language models independent of the number of discount parameters (whether the used Kneser-Ney smoothing is modified or not) and simply specify the number of discount numbers when we define the model. In principle, there is no heuristic to determine which of the modified and unmodified variants perform the best on a specific dataset, without trying out both and measuring their development perplexities.

The values of these discount parameters can be computed using either some analytical formula (for its upper bound) [Ney & Essen$^+$ 94] based again on some counts (to be more specific, count of counts[6]) when the formula is well defined (e.g. a division by zero can happen when the corresponding counts are zero), or better by a numerical optimization based on a development set [Sundermeyer & Schlüter$^+$ 11].

[5] It should also be noted that it is often the case in practice that all tokens in the training data which are not in the model's vocabulary are mapped to the unknown token. Therefore, n-gram events containing unknown tokens which are in the training data can also get a probability estimate like any other events.

[6] For example, a count of ones, which is also commonly called *singleton*, is a count of different events which are observed exactly once in the given dataset.

Role of n-gram count based language models? While largely outperformed by neural language models in terms of performance, the n-gram count models still remains an essential tool for language modeling; at least in the context of automatic speech recognition. First of all, it still remains the base language model for the *first pass* in the conventional speech recognition (as will be described in Sec. 1.2). In addition, it also quickly provides some statistics about the dataset: in particular, it allows to identify domain matches in the data which is a crucial information for carrying out *domain adaptation* of neural language models (Sec. 3.1.4), or detect some *accidents* in the data preparation, such as an overlap between the training and test data. We refer the interested readers to the Appendix B for further comments and explanation on the importance of these roles. It should however be noted that the research in improving count based language models (e.g. by improving the smoothing method) is rare in the recent years, with a few exceptions such as [Parikh & Saluja$^+$ 14, Shareghi & Gerz$^+$ 19]. As far as we are concerned, in this thesis, we take the count language model as a well established, simple but important tool in automatic speech recognition.

1.1.4 Neural Language Models

Instead of using counts, we can parametrize and estimate $p(w_m|w_0^{m-1})$ in Eq. (1.1) by an artificial neural network [Rumelhart & Hinton$^+$ 86, Lippmann 88]. In contrast to the count model, we refer to these models as neural network based language models or simply *neural language models* (as is in the title of this thesis).

Such a prediction problem naturally fits to the framework of artificial neural networks (especially recurrent neural networks) which is precisely a statistical tool for learning to map an input vector to an output vector. The only small tweak needed to frame the language modeling problem with a neural network is to represent each token in the vocabulary, which is a discrete symbol, by a so called *one hot* vector. One hot vectors are vectors of size of the vocabulary, where all entries are zeros, except for one entry of value one, at the index corresponding to the token to be represented. In consequence, the first layer of a neural language model is an *embedding layer* whose weight matrix stores the vector representation for each discrete input token. The multiplication between the input one-hot vector and the weight matrix therefore reduces to a simple look-up operation of the *embedding vector* corresponding to the input token. The embedding vector is then processed by hidden layers. Each of them maps its input to an output vector, and finally, a final *softmax layer* gives the probability distribution $p(.|w_0^{m-1})$ for the next token over the vocabulary.

When a recurrent neural network (RNN) [Elman 90] is used to parametrize $p(w_m|w_0^{m-1})$, the embedding vector corresponding to the predecessor token is fed to the network at each time step. In principle, the recurrent hidden layers of the RNN therefore keep dependencies to the full context w_0^{m-1}, independent of the context length, via its fixed size continuous state vector, which characterizes these *recurrent language models* [Mikolov & Karafiát$^+$ 10].

In contrast, in case of *n-gram feed-forward neural language models*, the Markov assumption is applied to truncate the context dependencies to the last $n-1$ tokens as in Eq. (1.3). The input to the neural network is then constructed by concatenating all embedding vectors for all $n-1$ input tokens, which is then processed by feed-forward layers.

Seminal works in neural language modeling. In fact, the first applications of artificial neural network to language modeling have been of this type of n-gram feed-forward models, presented in [Nakamura & Shikano 89, Nakamura & Maruyama$^+$ 90] under the name *NETgram*. Later, the n-gram feed-forward neural language model has been re-introduced for modern speech recognition in Yoshua Bengio and others' works [Bengio & Ducharme$^+$ 00, Bengio & Ducharme$^+$ 03] which have been followed up and further developed by Holger Schwenk and Jean-Luc Gauvain's works [Schwenk & Gauvain 02, Schwenk & Gauvain 04, Schwenk & Gauvain 05, Schwenk 07].

However, the neural language modeling has only become popular after the breakthrough papers on recurrent neural network based language models by Tomas Mikolov and others [Mikolov & Karafiát$^+$ 10, Mikolov & Kombrink$^+$ 11]. The popularization of the recurrent language model has motivated many research and engineering efforts in applying the recurrent language model to automatic speech recognition for the following years. Also importantly, the successful application of recurrent neural networks to language modeling, together with that of word embeddings [Mikolov & Sutskever$^+$ 13], has triggered people's interests in trying out neural networks, in particular RNNs which were believed to be hard to *make it work* in practice, across different applications, including neural machine translation.

Soon after [Mikolov & Karafiát$^+$ 10, Mikolov & Kombrink$^+$ 11], Martin Sundermeyer and others [Sundermeyer & Schlüter$^+$ 12] have empirically proven a recurrent neural language model based on an improved network architecture, long short-term memory (LSTM) [Hochreiter & Schmidhuber 96, Hochreiter & Schmidhuber 97] to outperform the standard RNN language model. These *LSTM language models* (or LSTM-RNN language models when the recurrence needs to be stressed) are today's *de facto* standard approach in language modeling. We will introduce recent developments and advances, as well as practical aspects of this model in the preliminary **Chapter 3**.

Tuning. An important aspect of neural language modeling which we need to introduce here is *tuning*. As will be also illustrated throughout this thesis, neural language models have many hyper-parameters which need to be specified to fully describe a model. The capacity of the model will depend on these hyper-parameters. However, the optimization process for training these neural language models being non convex, the performance of the model (specified by its neural network architecture and all its model hyper-parameters) after training will depend on the *training hyper-parameters*. Tuning corresponds to playing trials and errors with both these models (such as the hidden layer size and number of layers) and training (such as the learning rate and choice of optimizer) hyper-parameters in order to obtain the best performance. This leaves us comparing performance of different approaches, assuming that we were good enough at tuning the model well. While one can question the scientific validity of such an approach, we must argue that such a process is the current practice of applied deep learning. We note however, that we take extra care of doing our best to tune the baseline models well, in order to avoid the classic caveat of showing improvements over some weak baselines. Some concrete illustrations on the hyper-parameters tuning will be presented in the preliminary **Chapter 3**.

1.2 Automatic Speech Recognition

The automatic speech recognition is the task of transcribing a speech audio signal into a written language. While the history of statistical approaches for automatic speech recognition is long, the automatic speech recognition based on the hidden Markov model (HMM) [Baker 75, Rabiner 89] has been dominant since [Bahl & Jelinek$^+$ 83], which we refer to as *conventional HMM based speech recognition*. More recently, a new paradigm has emerged: *all neural end-to-end speech recognition* tries to implicitly implement all sub-components of automatic speech recognition in a single neural network with a single optimization process.

Currently these two paradigms coexist in both research and industry, because as of this writing, there has been no clear experimental evidence which has shown that either method predominates the other under all conditions, e.g. more data seem to be favorable for the end-to-end approaches.

From the view point of statistical language modeling, automatic speech recognition is the first problem which has introduced the practical usage of language models, and it remains the major application field of language modeling par excellence. In this section, we briefly introduce both of these conventional and end-to-end approaches with a special focus on how a language model

can be integrated into such systems.

For further generic details on ASR, we refer the readers to the corresponding textbooks such as [Rabiner 93, Huang & Acero$^+$ 01, Bourlard & Morgan 94, Yu & Deng 16].

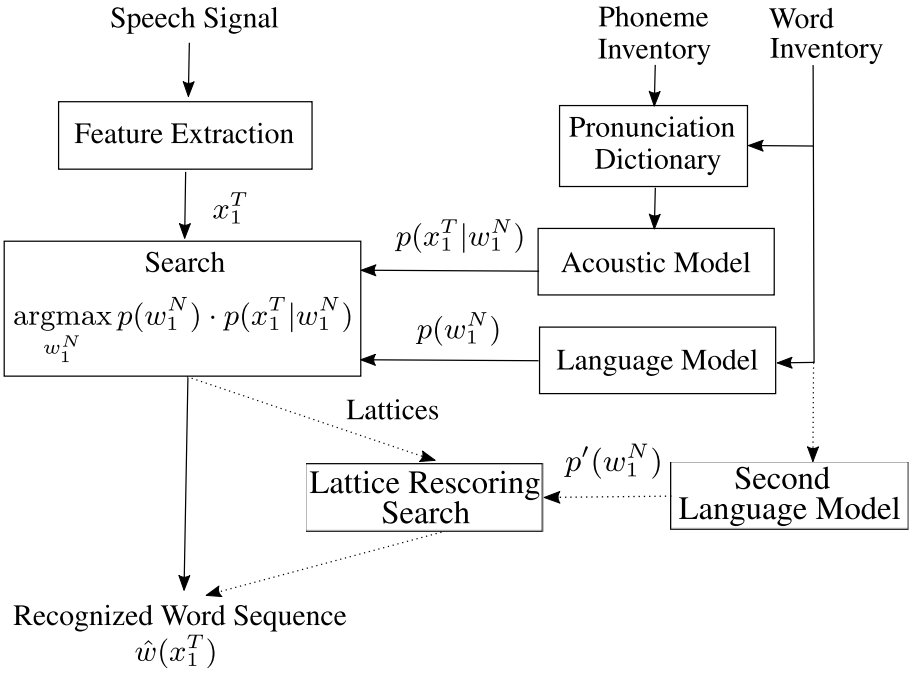

Figure 1.1: *Scheme for a conventional HMM based statistical speech recognition system based on Bayes decision rule [Bahl & Jelinek$^+$ 83]. The dashed arrows indicate the second pass lattice rescoring with a second language model (1.2.2), as is done with neural language models in this thesis.*

1.2.1 Conventional HMM based Automatic Speech Recognition

Automatic speech recognition is a task of finding the word sequence w_1^N corresponding to the input audio speech represented by a sequence of acoustic feature vectors x_1^T. The conventional statistical automatic speech recognition is based on the following Bayes' decision rule:

$$x_1^T \rightarrow \hat{w}_1^N(x_1^T) = \underset{w_1^N}{\text{argmax}}\{p(w_1^N|x_1^T)\} \tag{1.11}$$

$$= \underset{w_1^N}{\text{argmax}}\{p(w_1^N) \cdot p(x_1^T|w_1^N)\} \tag{1.12}$$

While the first equation Eq. (1.11) might seem more intuitive today where sequence-to-sequence learning with neural networks [Sutskever & Vinyals$^+$ 14] has become standard, the second equation Eq. (1.12), derived via Bayes' rule [Bayes 63], had two appealing properties in the past (and still today!). This decomposition dissociates the language model $p(w_1^N)$, which can be trained on text-only data, therefore allowing us to make use of text data without need for human labelling, from the *acoustic model* $p(x_1^T|w_1^N)$ for which a generative model based on Gaussian mixture models was available in the toolbox of statistical modeling at the time [Duda & Hart 73]. This equation, therefore, naturally introduces language modeling into speech recognition (as opposed to the end-to-end framework as we will see later in Sec. 1.2.3). This decomposition is also referred to as noisy channel decomposition in relationship with [Shannon 48, Shannon & Weaver 49].

Figure 1.1 illustrates the input and output of such a system, which are the speech signal and the recognized word sequence, and highlights the main system components in-between, namely feature extraction, acoustic and language models, and search.

The *feature extraction* transforms the raw time speech signal into a 10ms-level sequence of acoustic feature vectors x_1^T via signal processing pipeline such as MFCC [David & Mermelstein 80]. The acoustic model $p(x_1^T|w_1^N)$ is based on hidden Markov model (HMM) [Baker 75, Rabiner 89] whose hidden variable allows to model variability in speaking rate, by introducing the concept of alignment between the acoustic features x_1^T and the word sequence w_1^N. In a typical system, instead of directly using the word as an acoustic modeling unit, the word is decomposed into a sequence of some phonetically motivated subword units by following a manually designed *pronunciation dictionary* also called *lexicon*. The subword unit is typically constructed based on phonemes which defines the minimum acoustic unit. By indexing a phoneme by its predecessor and successor phonemes to take into consideration the co-articulation, we obtain the so-called *triphones*. Since the number of triphones can be large (and also in consequence, the estimation for rare events could be poor), they are typically clustered into generalized triphones called CART labels [Young 92], which finally define the unit for acoustic modeling.

The underlying generative probability (*emission probability*) of the acoustic feature vectors given the acoustic unit was primarily based on the Gaussian mixture model (GMM) [Duda & Hart 73], which together with a heuristic choice of an HMM topology and transition probabilities, results in an acoustic model based on GMM-HMM.

Given the acoustic and language models, the goal of the search process is then to efficiently find the most likely sequence of words for the input audio features. The corresponding algorithm is based on dynamic programming [Bellman 57, Ney 84] and makes use of heuristics for effective hypotheses pruning [Nolden 17].

Later, a number of works [Yu & Deng$^+$ 10, Dahl & Yu$^+$ 12, Seide & Li$^+$ 11] have revisited acoustic models (more specifically the emission probability of the HMM) which make use of neural networks instead of GMM. This approach is referred to as hybrid neural network-HMM (NN-HMM) approach [Bourlard & Morgan 89, Franzini & Lee$^+$ 90, Robinson & Fallside 91, Renals & Morgan$^+$ 94, Bourlard & Morgan 94][7], which has become one of the state-of-the-art approaches in ASR, allowing us to benefit from advances in neural network architectures, including LSTM [Sak & Senior$^+$ 14], improved TDNN [Povey & Cheng$^+$ 18] (whose precursors go back to [Makino & Kawabata$^+$ 83, Waibel & Hanazawa$^+$ 89]), and Transformer [Wang & Mohamed$^+$ 19].

1.2.2 Lattice Rescoring in Two-Pass Speech Recognition

The recognition process described in the previous section Sec. 1.2.1 is typically referred to as the *first pass* recognition. In the first pass, an n-gram count based language model (Sec. 1.1.3) rather than neural language models (Sec. 1.1.4) is typically used. This is simply because neural language models tend to be computationally more expensive (even for n-gram feed-forward models) and some models (e.g. recurrent language models) involve long-span context dependencies which does not directly fit to the original search process which assumes n-gram dependencies, unless we introduce modifications in pruning strategies.

While some works have successfully applied neural language models in the first pass, for both low order n-gram feed-forward language models [Schwenk & Gauvain 02, Huang & Sethy$^+$ 17] and recurrent neural language models [Huang & Zweig$^+$ 14, Hori & Kubo$^+$ 14, Lee & Park$^+$ 15, Beck & Zhou$^+$ 19], in this thesis, the application of neural network language model into an HMM based ASR system is mainly done via *lattice rescoring* [Weng & Stolcke$^+$ 98, Schwenk 07, Liu & Wang$^+$ 14, Sundermeyer & Tüske$^+$ 14] in the second pass. Lattice rescoring typically allows to exploit

[7] To be more specific, this is one of the NN-HMM hybrid approaches among others, cf. [Bengio & De Mori$^+$ 91].

most of the benefits from the neural language models [Sundermeyer & Ney$^+$ 15] unless the fact of having a system with two passes which could potentially result in a higher latency, is a technical concern, which is not the case for the experiments in this thesis.

An extra advantage of lattice rescoring is that it allows us to apply a lattice based decoding algorithm such as confusion network decoding [Wessel & Schluter$^+$ 01, Hoffmeister 11], which is theoretically more consistent with the ASR word error rate (Levenshtein distance) and empirically results in slight improvements over the first pass Viterbi decoding.

In Figure 1.1, the dashed lines illustrate the additional second pass lattice rescoring. Lattices (also called word graphs) [Oerder & Ney 93] are generated from the first pass recognition process [Ortmanns & Ney$^+$ 97]. They compactly represent most likely hypotheses from the first pass system in the form of a directed graph, in which each arc stores acoustic and language model scores together with the corresponding word identity, and each node stores the audio time stamp. The main goal of the second pass lattice rescoring consists of "overwriting" the language model scores on the arcs by using a more powerful language model. However, the use of full context language model (such as an RNN language model) would result in an exponentially increasing number of hypotheses to be evaluated, and all scores can not be written back in the original lattice topology generated by an n-gram model, as the longer dependency results in more scores than the number of arcs in the original lattice. A version of push-forward algorithm [Auli & Galley$^+$ 13] augmented with pruning method has been proposed by [Sundermeyer & Tüske$^+$ 14]. While a number of alternative methods have been also investigated [Liu & Chen$^+$ 16], in this thesis, we make use of Sundermeyer's push-forward algorithm [Sundermeyer & Tüske$^+$ 14] which makes use of the traditional hypotheses pruning strategies for ASR. Following [Sundermeyer & Tüske$^+$ 14], we expand the lattice after rescoring.

The corresponding algorithm is illustrated in Figure 1.2. The core idea of the algorithm is intuitive: as we visit each node in the lattice in a topological and time increasing order (so that we can evaluate all hypotheses from all incoming arcs to the visited node), the partial hypotheses are stored on the nodes. As the number of hypotheses per node becomes larger as we go deeper in the lattice, we apply pruning to hypotheses sharing the same acoustic coverage, in order to control the number of surviving hypotheses to keep computation practically manageable. In the end of the algorithm, we obtain the corresponding traceback array storing all arc information, which we can follow to construct the new (larger) lattice (*expanded lattice* in the terminology of [Sundermeyer & Tüske$^+$ 14]) with new language model scores on the arcs.

Data: Language model $\mathcal{M}$,
Lattice represented by nodes N (`get_outgoing_arcs`, `get_time`), arcs A (`get_to_node`, `get_word`), initial node s, final node f
Require: Pruning threshold γ, recombination order m
Result: Traceback array for the rescored lattice $\mathcal{T}$ (`set_predecessor`), New best hypothesis (optional)
$N' \leftarrow$ `sort_topological_and_time`(N)
$H(n)$ stores a triplet (score, LM state vector, partial hypothesis string) for all partial hypotheses stored on node n
Set the start node
$H(s)$.`push`$((0, 0, \text{“”}))$
$t = 0$
# Visit each node in topological and time ascendant order
for $n \in N'$ **do**
  if n.`get_time` $> t$ **then**
    # Keep only one hypothesis if two hypotheses share the last m words
    # Apply beam pruning with threshold γ.
    `prune`$(m, \gamma, H, n$.`get_time`$)$
    $t = n$.`get_time`
  end
  # For each outgoing arcs (n, n') from the node n
  for $(n, n') \in n$.`get_outgoing_arcs` **do**
    # Forward each hypothesis stored on the node n
    for $h \in H(n)$ **do**
      $h' \leftarrow$ `evaluate`$\big(\mathcal{M}, h,$ `get_word`$(n, n')\big)$;
      $\mathcal{T}$.`set_predecessor`(h, h');
      $H(n')$.`push`(h');
    end
  end
end
$H(f)$.`best_scoring_hyp`() gives the new best hypothesis.
$\mathcal{T}$ stores all connection information needed to construct rescored lattice.
return $\big(H(f)$.`best_scoring_hyp`$(), \mathcal{T}\big)$;

Figure 1.2: Pseudo-code adapted from [Sundermeyer 16] for Sundermeyer's push-forward lattice rescoring algorithm.

1.2.3 End-to-end Speech Recognition with Encoder-Decoder Models

This section shortly describes the second ASR paradigm where we show improvements in and by language modeling in this thesis, which is the *end-to-end speech recognition with encoder-decoder models*. As opposed to the HMM based ASR approach presented in the previous section (Sec. 1.2.1), a single neural network is used to parametrize $p(w_1^N|x_1^T)$ in Eq. (1.11) without decomposition as in Eq. (1.12), to directly learn to map the frame-level input audio features to the output word sequence. In fact, instead of using words as the output unit, models are typically trained to output *character-based* subword units: graphemes, byte-pair encodings (BPEs) [Sennrich & Haddow$^+$ 16b], or word-pieces [Schuster & Nakajima 12], without using a hand-crafted pronunciation lexicon. Therefore, in principle, these encoder-decoder speech recognition models jointly learn the acoustic model, pronunciation model, and language model within a single neural network. This property makes it an *end-to-end* approach for speech recognition.

To be more specific, the models we will be referring to are categorized as *Listen, Attend, and Spell* (LAS) models [Chan & Jaitly$^+$ 16]. The LAS model, which is depicted in Figure 1.3, has encoder, attention, and decoder modules. The *encoder* transforms the input frame-level audio feature sequence into a sequence of hidden activations. The *attention module* summarizes the encoder sequence into a single vector for each prediction step, and finally, the *decoder* models the distribution of the output sequence conditioned on the history of previously predicted labels. Both the encoder and the decoder are modeled using neural networks, and thus the entire model can be jointly optimized. We refer the readers interested in further details of these models to [Chan & Jaitly$^+$ 16, Prabhavalkar & Rao$^+$ 17, Weiss & Chorowski$^+$ 17, Zeyer & Irie$^+$ 18]. It has been shown that such models can outperform the conventional HMM based system (Sec. 1.2.1) when they are trained on sufficiently large amount of data [Chiu & Sainath$^+$ 18].

The emergence of such approaches were natural. The success of sequence-to-sequence learning with encoder decoder neural networks in machine translation [Sutskever & Vinyals$^+$ 14, Cho & Gülçehre$^+$ 14], especially those using attention [Bahdanau & Cho$^+$ 15], has quickly triggered interests in applying the same paradigm to speech recognition [Lu & Zhang$^+$ 15, Chorowski & Bahdanau$^+$ 15, Bahdanau & Chorowski$^+$ 16, Chan & Jaitly$^+$ 16]. The only modification specific for the speech application is the handling of the large number of input frames in the encoder for the acoustic features, which is typically resolved via some downsampling [Chan & Jaitly$^+$ 16].

While we focus on this LAS type of models to evaluate the language model in an end-to-end ASR system, there are many other end-to-end ASR models, such as recurrent neural network transducer (RNN-T) [Graves 12, Rao & Sak$^+$ 17] or models trained with the connectionist temporal classification criterion [Soltau & Liao$^+$ 17, Graves & Fernández$^+$ 06]. In particular, the introduction of RNN-T models go back to [Graves 12] which was before the popularization of today's sequence-to-sequence learning.

Language model integration. As has been noted above, the encoder-decoder attention ASR model does not make use of the Bayes' rule based decomposition which results in separate acoustic and language models. Therefore, in principle, the model does not require a language model by construction. In fact, the *decoder* component already disposes some internal language model. Therefore, a separate conventional language model is often rather referred to as an *external language model*.

However, it remains true that such external language models can be trained on large amount of data, without requiring the data with audio and paired human transcriptions. The use of an external language model is therefore still appealing as a method to exploit abundant text-only data in an end-to-end speech recognition paradigm. A number of methods have been investigated previously [Gülçehre & Firat$^+$ 17, Sriram & Jun$^+$ 18, Stahlberg & Cross$^+$ 18] in how to integrate an external language model into end-to-end systems for both machine translation and automatic

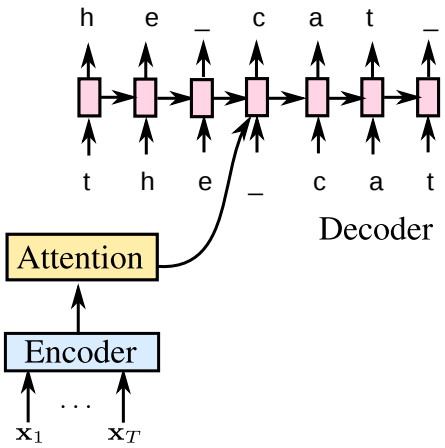

Figure 1.3: *Listen Attend and Spell.* Figure taken from [Irie & Prabhavalkar$^+$ 19b].

speech recognition. They differ from each other in at which level (e.g. hidden state vs. probability distribution level) the language model is combined with the main LAS model. While none of them seems to have reached the single method which works the best in all cases, the simplest log linear interpolation on the probability level [Toshniwal & Kannan$^+$ 18, Zeyer & Irie$^+$ 18] has been consistently shown to work well. We refer to this log linear interpolation as *shallow fusion* to respect the original authors' terminology [Gülçehre & Firat$^+$ 17]. In this thesis, we therefore focus on the shallow fusion as a method of integrating and evaluating a language model in the encoder decoder ASR framework. The combined score used for the beam search at the target position m is thus:

$$\log p(w_m|w_0^{m-1}, x_1^T) + \lambda \log p(w_m|w_0^{m-1}) \tag{1.13}$$

where w_m denotes sub-word unit token and the *language model scale* λ is chosen to minimize the WER on the development set. A few extension which makes use of an additional end of sentence penalty [Hannun & Lee$^+$ 19] is studied and discussed in this thesis.

As a side note, it is worth noting that in machine translation, *back-translation* [Sennrich & Haddow$^+$ 16a] has been shown to be a better method of making use of the unpaired data (*monolingual data* as referred to in the context of machine translation) than external language models. Back-translation lies in generating synthetic paired data using another translation model which translate languages in the backward direction. The generated paired data is mixed with the human generated bilingual texts (often with some oversampling of the true bilingual data) to train the translation model in the original direction. While we can potentially get additional improvements by applying an external language model on top of the back-translation, the mainstream for exploiting monolingual data in machine translation today has become back-translation. The counter-part of back-translation for the ASR, which makes use of some text-to-speech system [Tjandra & Sakti$^+$ 17, Rosenberg & Zhang$^+$ 19] to generate synthetic data has only recently emerged.

2. Scientific Goals

Statistical language modeling is one of the fundamental problems in natural language processing. It has many applications in language technology including the historical and arguably the most prominent application to automatic speech recognition. The general goal of this thesis is to study and advance language modeling for improving automatic speech recognition.

Based on pioneers' tremendous works on recurrent neural network based language modeling, such an approach has been the main source of recent improvements in language modeling. Its application to automatic speech recognition is ubiquitous in research papers. This omnipresence even gives impression that existing works already provide a short answer to all crucial aspects of neural language modeling in speech recognition, covering modeling (LSTM-RNN), training (GPU), search (rescoring), and adaptation (fine-tuning). In fact, it is true that the pipeline is already well established. Especially on the modeling side, full context aware LSTM-RNNs seem to provide a powerful and generic solution for language modeling, solving problems of the limited context and data sparsity which all previous approaches have been suffering from.

While this might give impression of some degree of maturity, we claim that, in contrary, the full potential of the neural network based language modeling is yet to be explored. In this thesis, we further advance neural language modeling in automatic speech recognition, by investigating a number of new perspectives. The overview of the scientific goals is as follows:

Improving language modeling with deep Transformers. Recently, the new Transformer neural network architecture has been introduced for machine translation. The model has been applied to many other natural language processing tasks since then, outperforming the state-of-the-art LSTM based models on multiple tasks. In **Chapter 4**, we aim at adopting Transformers for language modeling. As a model originally introduced for machine translation, the need for scaling up the model naturally arises, as the task of language modeling disposes much more data for training. It is also unclear whether Transformers are better suited for language modeling than LSTMs: on one hand a previous work [Chen & Firat$^+$ 18] has reported that for the encoder decoder based translation, the LSTM based decoder (which is the component often referred to as operating as a language model to some extent) works better than a Transformer based one, while on the other hand, [Al-Rfou & Choe$^+$ 19] more recently have reported deep Transformers to perform well for character-based language modeling in their text based experiments. A systematic experimentation is therefore needed to validate whether Transformers can perform well for language modeling with applications to ASR, and eventually outperform the state of the art LSTM-RNN (or like many other methods, it finally turns out to underperform a well tuned LSTM baseline). Its *applicability for speech recognition* as well as its *scalability* with respect to the amount of training data must be investigated. Also, Transformers' memory requirement is more demanding than that of an LSTM: some *better architectural variants* are desirable. The goal of **Chapter 4** is thus to consider and validate Transformer based neural language models for automatic speech recognition.

Improving models which have a better structure for decoding via knowledge distillation. Neural language modeling allows us to benefit from new techniques in deep learning developed for general purposes. In **Chapter 5**, we are interested in applying one of such techniques: *knowledge distillation*. In particular, we investigate potential improvements of models whose original performance lags behind that of models with a state-of-the-art model architecture, but which have a structure or property which is convenient for decoding. The practical complexity of language modeling increases from n-gram feed-forward models to LSTM models, and from LSTM models to Transformers. N-gram feed-forward models are easier to be integrated into the first pass speech recognition process than the full context LSTM models. The LSTM allows fixed size memory during evaluation independent of the sequence length, as opposed to Transformers. Therefore, there are practical interests in studying the real limit and potential improvements of these models which are more convenient for search. In addition, despite this clear motivation and the popularity of knowledge distillation in general, there has been no application of knowledge distillation for language modeling in speech recognition. In fact, the naive application using the full softmax output layer would be costly for large vocabulary language modeling. We need to study the combination of distillation methods with the speed-up techniques used for large vocabulary language models. The goal of **Chapter 5** is therefore two-fold. First, we introduce knowledge distillation for large vocabulary language modeling, which will be also a crucial tool for investigating the goal of **Chapter 6** on domain robust language modeling below. Second, we explore whether the performance gap from the architectural difference (and therefore modeling power) between different neural language models can be reduced via knowledge distillation.

Domain robust language modeling. While scaling up language modeling to larger datasets, the diversity of the data emerges as an opportunity and a challenge. The current state-of-the-art neural language modeling lacks a mechanism of handling diverse data from different domains for a single model to perform well across different domains. The existing works only focus on obtaining a model which is trained to perform well on a single domain test data. To that end, a domain adaptation can be carried out: a simple fine-tuning on the relevant subset of the training data typically work best. But such a straightforward process only works when there is a single target domain. There is no solution in the literature for obtaining a single neural language model which can perform well across various target domains. In **Chapter 6**, we introduce *domain robust language modeling* which aims at building a single model which performs well on multiple domains, even potentially robust to unseen domains. We take the diversity which is a natural property of a large dataset, as an opportunity to further advance neural language modeling. We propose two solutions for this problem. The first method is a *recurrent adaptive mixture model*, which is inspired by the Bayesian interpolation of n-gram count based language models. The second method makes use of *knowledge distillation* from multiple domain expert language models. The objective of **Chapter 6** is thus to develop and validate these methods of domain robust language modeling on large industry level speech recognition tasks, namely YouTube speech recognition at Google and AppTek multi-domain speech recognition.

Cross-sentence long-span language modeling. Neural language modeling seems to facilitate long-span modeling and evaluation. For example, an LSTM language model can be evaluated on the whole document considering context across sentence boundaries. However, the *robustness* of these models with respect to the evaluation sequence length is rarely discussed. In particular, the training sequence construction must be revised and ideally made consistent with the long-span evaluation. Also, such a robustness could largely depend on the model architecture. We investigate this fundamental property of neural networks which can handle variable length contexts, for both LSTM and Transformer language models. The resulting models must be applied to cross-utterance

automatic speech recognition for validation.

The possibility to handle long contexts also extends the reach of language modeling. Many natural language processing tasks can be formulated as a long-span language modeling task. For example, the sequence to sequence machine translation task can be linearized to a single sequence language modeling task by concatenating the source and target language sentences. Transformer language model seems to be particularly suited for such a task, as the model does not present any apparent limitation for that end. Comparing such an approach with the standard encoder-decoder attention approach also allows to evaluate the limit of the current sequence processing model on handling long-span dependencies.

The goal of **Chapter 7** is therefore to study the potential of these cross-sentence long-span language models with application to both speech recognition and machine translation.

The rest of this thesis is organized according to these goals as defined above. The coming up next **Chapter 3** is a preliminary. It provides an overview of concepts and preliminary models and experiments, which must be introduced before addressing these core goals, including the **foundation of the current state-of-the-art LSTM-RNN language models** developed in the course of this thesis.

3. Basic Concepts and Developments of Neural Language Modeling

Before delving into the core scientific goals addressed in this thesis aforementioned in the previous Chapter 2, this chapter introduces a number of preliminary results which will serve as background knowledge in the following chapters.

First and foremost, in the following Sec. 3.1, the developments and recent advances in the state-of-the-art LSTM language models are presented with experimental results. While the model itself has been known to perform well since its introduction in [Sundermeyer & Schlüter$^+$ 12], a number of enhancements to the original work, as well as comparison to alternative models has been investigated by many researchers including ourselves along the development time of this thesis. **The recipe for the current state-of-the-art LSTM language models** are in fact build on top of these incremental trials and errors, and progressive integration of practical knowledge. Sec. 3.1 highlights the key advances and practical methods (such as training sequence construction and domain adaptation), as well as a detour into models based on highway connections [Srivastava & Greff$^+$ 15a, Srivastava & Greff$^+$ 15b] which are derived from the successful gating mechanism of the LSTM. These setups allow us to obtain strong baselines in this thesis.

Second, in Sec. 3.2, we present a couple of primitive **tentatives in applying the attention mechanism to language modeling**, in the time between the emergence of the attention mechanism [Bahdanau & Cho$^+$ 15] and the advent of the Transformer architecture [Vaswani & Shazeer$^+$ 17] which established a general purpose usage of attention. We briefly presents these precursors of Transformer language models which will be fully discussed in Chapter 4, as related models. This will not only allow us to provide some historical backgrounds, but also to contrast and emphasize the origin of the power of Transformer architectures, and to some extent the necessity for a deep architecture later.

Finally, the aforementioned **correlation between the language model perplexity and the ASR word error rate** are further discussed with experimental illustrations in Sec. 3.3. This is an essential step before discussing language modeling for automatic speech recognition, as we typically have two stages in preparing language models for ASR. In a primarily tuning step, language models are evaluated and selected solely based on the perplexity. In the second phase, these best models are integrated into an ASR system and evaluated based on the final word error rate.

Compared with the following chapters with core scientific questions, the experimental results in this chapter are kept brief on purpose, as they are solely intended to highlight some key properties of neural language models. We note that the statistics and the descriptions of the dataset can be found in Appendix A. Unless the information is relevant for discussing the experimental results, we will therefore refer to the dataset without giving a specific description at each reference.

3.1 State-of-the-art LSTM-RNN Language Models

This section highlights the ingredients for building current state-of-the-art LSTM language models, together with its key properties, as well as some historical research directions which are worth being presented for better understanding the foundation of current state-of-the-art methods.

The LSTM-RNN language model has first appeared in the literature in [Sundermeyer & Schlüter$^+$ 12] shortly followed by [Frinken & Zamora-Martinez$^+$ 12] and [Graves 13]. After some time, the LSTM has become the *de facto* standard approach for language modeling [Sundermeyer & Ney$^+$ 15, Jozefowicz & Vinyals$^+$ 16, Xiong & Droppo$^+$ 17]. In the meanwhile, the LSTM language model has also gone through a test time. The first type of the research in this period can be categorized as investigations on the model architecture variants (Sec. 3.1.1) [Jozefowicz & Zaremba$^+$ 15, Greff & Srivastava$^+$ 17]. These research works have shown limited benefits from slight architectural changes in the original LSTM, but played an important role of showing the robustness of the original architecture[1]. On the other hand, the design of the LSTM, in particular the use of intuitive multiplicative gates has inspired many other, also feed-forward network, architectures. The most notable of them is the highway network [Srivastava & Greff$^+$ 15a, Srivastava & Greff$^+$ 15b]. We will make a brief detour into the use of highway connections in language modeling in Sec. 3.1.6 as the mechanism is directly driven from the LSTM: as we will see, this has also not given large improvement over the standard LSTM.

In contrast, large improvements in LSTM language modeling has obtained by the introduction of the dropout [Hinton & Srivastava$^+$ 12, Srivastava & Hinton$^+$ 14]. It has been shown that the use of dropout together with increase in the model size turns out to be very effective [Zaremba & Sutskever$^+$ 14], which has become today's common spirit for tuning neural language models.

This section is dedicated to summarize these important aspects of LSTM language modeling.

3.1.1 Standard Architecture

The modern LSTM architecture is based on the original LSTM-RNN cell [Hochreiter & Schmidhuber 96, Hochreiter & Schmidhuber 97] augmented with the forget gate [Gers & Schmidhuber$^+$ 00] and the peephole connections [Gers & Schraudolph$^+$ 03]. A typical LSTM layer with weight matrices W_y, W_i, W_f, W_o, R_y, R_i, R_f, R_f, and weight and bias vectors w_i, w_f, w_o, b_i, b_f, b_o, b_y, transforms the input vector x_t to the output vector h_t via internal memory cell c_t as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + R_i h_{t-1} + b_i + w_i \odot c_{t-1}) && (3.1)\\
f_t &= \sigma(W_f x_t + R_f h_{t-1} + b_f + w_f \odot c_{t-1}) && (3.2)\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_y x_t + R_y h_{t-1} + b_y) && (3.3)\\
o_t &= \sigma(W_o x_t + R_o h_{t-1} + b_o + w_o \odot c_t) && (3.4)\\
h_t &= o_t \odot \tanh(c_t) && (3.5)
\end{aligned}
$$

which is often shortened as follows when only highlighting the inputs and outputs is needed:

$$
(h_t, c_t) = \mathrm{LSTM}(h_{t-1}, c_{t-1}, x_t) \quad (3.6)
$$

In practice, the effect of peephole is rather limited[2]. Therefore, the corresponding terms (last terms in the equations for the gates Eqs. (3.1, 3.2, 3.4)) are omitted in modern implementations of LSTM (e.g. in TensorFlow [Abadi & Barham$^+$ 16]) for efficiency.

As can be seen from the LSTM equations above, there seems to be a plenty of room for creativity in slightly modifying the architecture. A popular research direction following the recent

[1] Simply formulated, it was hard to beat a well tuned LSTM language model!

[2] In our experiments for language modeling, we observe less than 1% relative change in terms of final perplexity.

re-emergence of LSTMs were thus the ablation study and the investigations for better or simpler architectures. We refer for example to [Jozefowicz & Zaremba$^+$ 15, Greff & Srivastava$^+$ 17]. While it has been difficult to conclude on one ultimate architecture which gives the best performance for all tasks under all conditions, an important outcome from these investigations is that the reliability of the original LSTM architecture across different tasks has been empirically proven. In other words, only limited improvements can be expected from slight architectural changes around the LSTM, which dismissed the future research work from focusing on the architecture when it is not at the center of the corresponding research.

For example, the most popular of architectural alternatives to the LSTM is arguably the gated recurrent unit (GRU) [Cho & Gülçehre$^+$ 14]. A version (since the GRU itself has some architectural variants!) proposed by [Chung & Gülçehre$^+$ 14]) is defined as follows:

$$z_t = \sigma(W_z x_t + R_z h_{t-1} + b_z) \tag{3.7}$$
$$r_t = \sigma(W_r x_t + R_r h_{t-1} + b_r) \tag{3.8}$$
$$y_t = \tanh(W_h x_t + R_h(r_t \odot h_{t-1}) + b_h) \tag{3.9}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot y_t \tag{3.10}$$

where W_z, W_r, W_h, R_z, R_r, and R_h are weight matrices, and b_z, b_r, b_h are bias vectors. The GRU has only two gates (reset r_t and update z_t) and does not have the memory cell. This belongs thus to simplification efforts in the LSTM architecture. As an illustration, we report our comparison between the LSTM and GRU from [Irie & Tüske$^+$ 16] for language modeling for the Quaero English task (Appendix A.3) in Table 3.1. We concluded that the LSTM seems to work better for language modeling.

Table 3.1: *Perplexities on Quaero English development data for standalone LSTM and GRU. The perplexities for 1- and 2-layer LSTMs are taken from [Sundermeyer & Ney$^+$ 15]. Exceptionally here, in order to be consistent with [Sundermeyer & Ney$^+$ 15] the perplexities are evaluated by concatenating evaluation sentences into sequences in the original order such that each sequence contains at most 100 words.*

Num. units	LSTM		GRU	
	Number of layers			
	1	2	1	2
100	147.0	139.6	143.9	136.4
200	127.7	117.7	121.9	116.8
300	117.6	109.1	115.7	110.7
400	112.8	104.6	114.7	110.0
500	109.2	101.8	112.6	**108.1**
600	107.8	**100.5**	112.2	108.9

The trend of investigating architectural changes has however continued, but often, it has been found difficult to achieve improvements over a well tuned LSTM baseline [Melis & Dyer$^+$ 18].

3.1.2 Improvements by Larger and Well Regularized Models

The previous subsection has shown that the investigations on the architectural variants of the LSTM have given only limited improvements in terms of performance.

In contrast, the introduction of dropout [Hinton & Srivastava$^+$ 12, Srivastava & Hinton$^+$ 14] has played a crucial role in the progress of neural language modeling, as it has been successfully demonstrated in [Zaremba & Sutskever$^+$ 14]. These works have also contributed in making LSTM

language model more popular [Jozefowicz & Vinyals+ 16]. This trend has triggered the idea of making neural language models *bigger with some good regularization*, which has today become a master recipe for obtaining a good neural language model in general. We note however that such a trend was made possible because of the popularization of GPU hardwares together with open source implementations of core components of language models [Abadi & Barham+ 16, Zeyer & Alkhouli+ 18].

The dropout itself has a couple of different variants [Gal & Ghahramani 16]. One popular discussion has been whether we should better apply dropout on the recurrent connections or not. In our preliminary experiments, we did not find such recurrent dropout to give improvements, while it also did not make the model performance worse. Thus, for the study in this thesis, we chose to always only apply dropout on the feed-forward connections (e.g. in case of an LSTM layer only to x_t in Eqs. (3.1 - 3.4)).

As an illustration, Table 3.2 shows the improvements by this large and regularized language model, again on the Quaero English task. This configuration of having two LSTM layers with 2048 nodes each with dropout rate of 0.2 has become standard in our recipe. We note that all models in the table have been trained with the standard stochastic gradient descent with a learning rate of 1. The global norm clipping threshold is set to 2 for all model except for the small 600-size model for which 1 worked better[3].

Table 3.2: *Large and regularized models work well. Perplexities of 2-layer LSTM language model on Quaero English. The baseline 600-unit model architecture corresponds to the best model at the time of [Sundermeyer & Ney+ 15] (re-trained on the sentence level for a fairer comparison, instead of directly using the model from [Sundermeyer & Ney+ 15] trained on the concatenated sentences, as we report perplexities on the sentence level here).*

Num. units	Dim. input embedding	Dropout rate	Perplexity	
			Dev	Eval
600	600	0.0	107.1	106.6
1024	128		102.6	102.3
2048			105.3	104.8
		0.2	**84.5**	**86.3**

This trend also has triggered further research on regularization methods for language modeling [Merity & Keskar+ 18]. Some of the ASR tasks we consider (such as Switchboard) indeed suffer from overfitting, therefore benefit from regularization.

However, we note that other large scale tasks on the contrary rather suffer from underfitting, given the size of LSTM models we can train in a reasonable amount of time using the current state of the hardware[4]. For example, for the LibriSpeech dataset (Appendix A.1) containing about 850 M running words for training, we had to remove the dropout to obtain the best model. The corresponding perplexities are presented in Table 3.3. We use the official 4-gram count language model provided with the LibriSpeech dataset [Panayotov & Chen+ 15]. No improvement in perplexity was observed when going up to 5-grams. For LSTM-RNN language models, we first

[3] 1 for *small* models and 2 for *larger* models have been empirically found to be a good heuristic, which is also part of our recipe.

[4] Here is one side note about the hardware: at the beginning of the work on this thesis, we used to train our models on multi-threaded CPUs using Martin Sundermeyer's C++ based software `rwthlm` [Sundermeyer & Schlüter+ 14]. Later, we have moved to the RETURNN framework [Zeyer & Alkhouli+ 18] in which the TensorFlow [Abadi & Barham+ 16] support has been developed by Albert Zeyer, and we consequently moved to training models on GPUs. Also in terms of GPU types, we got access to 1080s, and finally even V100 which have been made available at RWTH IT Center, towards the end of this thesis work period.

trained our base configuration: the model has 2 LSTM-RNN layers with 2048 nodes and the input projection layer of 128, where the dropout with a rate of 0.2 is applied between each layer. Since we observed that this model underfits the LibriSpeech training set, we removed the dropout and further increased the model size, which effectively gave better perplexities as shown in Table 3.3. We can see that improvements from simply stacking layers saturate at 4 layers, even without overfitting. Introducing a small linear bottleneck layer (size 512 here) before the output layer can make the models compact, but with a loss in performance. The best model we obtained has 2 layers with 4096 nodes. Relative improvements greater than 58% has been obtained by the LSTM over the 4-gram language model.

Importantly, it should be noted that if the computation allows, we could potentially further increase the model size and apply regularization to achieve improvements (without innovation!) in language modeling.

Table 3.3: *Perplexities of LSTM language models on LibriSpeech. Illustrating model tuning on a large dataset.*

Model	Dropout	Bottleneck	Num. units	Num. layers	Params in M	Dev	Test
4-gram	-	-	-	-	230	146.2	151.8
LSTM	0.2	None	2048	2	487	71.3	74.8
	0.0					66.6	69.9
				3	520	64.0	67.2
				4	554	**61.9**	**64.9**
				5	587	62.7	65.9
				6	621	64.5	67.5
				8	688	67.2	70.3
			4096	2	1048	**60.2**	**63.2**
		512			334	63.1	66.3
			2048	4	248	64.5	67.7

3.1.3 Training and Evaluation Sequence Construction

The two sections above has focused on the model architecture. This section deals with the practical details of the training. All neural language models in this thesis (and also certainly all existing neural language models) are trained using back-propagation with stochastic gradient descent. Specifically, we make use of the plain stochastic gradient (SGD) optimizer using a high learning rate of 1 unless otherwise specified[5] with global gradient clipping and Newbob learning rate scheduling [ICSI 00] which consists in reducing the learning rate based on the development perplexity. We also note that we always train our models using early stopping [Bengio 12] by selecting the final model based on the development perplexity. Despite our multiple trials, we always observed the final perplexity after convergence to be better when the plain SGD is used, compared with the more popular Adam algorithm [Kingma & Ba 15].

Another important aspect to be specified for training neural language models is the batch construction or the definition of training sequences. Training on GPUs while having multiple sequences with different lengths in the same batch implies zero padding. For an optimal usage

[5]We learned this recipe from Jacob Devlin's slides for his invited talk at WMT 2015 [Devlin 15].

of resources, zero padding must be reduced. We refer e.g. to [Chen & Wang+ 14] for further discussion on the topic.

In our default setups, all our training sequences are sentences. We do not apply any truncation to the back-propagation through time, because the full backpropagation is more computationally efficient [Zeyer & Alkhouli+ 18]. And we construct batches by shuffling all sentences randomly. Unless otherwise specified, this construction method is used by default.

Alternatively, we first shuffle the sentences, sort them by length, then create bins, and finally bins are shuffled. The sorting process helps reducing the zero padding. This method has been found to accelerate the training typically by a factor of two, without degradation in terms of final perplexity[6].

The reason for always using the sentences as sequence unit in this thesis is simple. Unless otherwise explicitly specified, all perplexities reported in this thesis is computed by treating each sentence in the text independently. The training and evaluation are therefore consistent, and we experimentally find that this results in optimal performance. We note that this preliminary chapter reports many perplexity results based on the concatenation of sentences, as exceptions, because of consistency with some previous works[7].

As the use of context beyond sentence boundary, as well as the impact of consistency between training and evaluation, are interesting topics in their own with limited previous works, we will have a dedicated study later in this thesis, in **Chapter 7**, Sec. 7.1. For this section, we only show a simple example on the AMI task (Appendix A.5) to illustrate the importance of caring about such an effect. The AMI task has two versions of the datasets which differ from each other in whether the utterance is segmented after punctuation (therefore changing how sentences are defined) or not. Table 3.4 illustrates the clear benefit of being consistent in training.

Table 3.4: *Perplexities of LSTM language models on AMI. Effect of training consistent with evaluation segmentation (split after punctuation). The development and evaluation sets are* ***not segmented****.*

Training Segmentation	Perplexity	
	Dev	Eval
Yes	62.1	67.3
No	**57.3**	**60.2**

3.1.4 Domain Adaptation

Another important practical technique for neural language modeling in ASR is the domain adaptation. It is often the case that there is a large amount of general purpose training data (background data) for language modeling and rather limited amount of data which matches the domain of the target task. The domain adaptation becomes relevant as soon as we are in such a scenario. A couple of research works [Ter-Sarkisov & Schwenk+ 15, Gangireddy & Swietojanski+ 16, Ma & Nirschl+ 17, Tüske & Irie+ 16] have investigated different methods for domain adaptation in neural language modeling, such as introduction of some adaptation layer in the model to be trained on the target domain data. A global statement which can be extracted from these studies is that, the simple method of fine-tuning the whole model on the target domain data performs competitively well compared with other more sophisticated approaches.

[6] This method has been tried out relatively late in the time of the thesis. Therefore, it is only used for a limited number of experiments.

[7] This chapter contains relatively old results which were generated using the old software [Sundermeyer & Schlüter+ 14] where the training and evaluation of perplexities were done on the concatenation of sentences in such a way that we do not exceed the pre-determined number of words, typically 100.

Again, the AMI task (A.5) can serve as an illustrative example. In AMI task, we only have 850 K-word training data from AMI transcriptions. In a typical setup, we therefore use in addition the Switchboard 27 M words out of domain dataset. We first train the LSTM language model on the whole data, and then after convergence, we continue training only on the AMI dataset. Table 3.5 illustrates the effect of fine-tuning.

This elementary property is exploited later, when we build *domain robust language model* in **Chapter 6**.

Table 3.5: *Effect of fine-tuning on the target domain data. Perplexities of an LSTM language model on AMI.*

Fine-tuning	Perplexity	
	Dev	Eval
No	68.8	72.5
Yes	**57.3**	**60.2**

3.1.5 Modeling Units

The main focus of language modeling for ASR is rather on the word-level modeling in this thesis, as is typically the case of the state-of-the-art NN-HMM based ASR systems (Sec. 1.2.1). However, as noted in Chapter 1, Sec. 1.2.3, we also train subword level language models to be combined with end-to-end ASR models. The natural question is then whether these models defined on different vocabularies differ in terms of performance. The perplexities of these models are obviously not comparable as they are defined over different vocabularies, and the tokenizations of the text is different, which results in different number of tokens. The latter mismatch can be easily resolved by introducing a re-normalization of the perplexity in terms of the number of tokens, when computing the average in the definition of perplexity (Eq. (1.2)). Namely, we can convert any perplexity into a character level one by the following renormalization, as shown as an example of conversion from the word level to the character level:

$$\mathrm{PP}_{\mathrm{char}} = \left(\mathrm{PP}_{\mathrm{word}}\right)^{-\frac{\#\mathrm{words}}{\#\mathrm{char}}} = e^{-\frac{\#\mathrm{words}}{\#\mathrm{char}} \log \mathrm{PP}_{\mathrm{word}}} \tag{3.11}$$

where #char denotes the number of characters in the text including word end symbol (spaces) even for the last word in a sentence, and the sentence end tokens.

While this comparison still does a hand-waving on the fact that the normalization over the same length does not fully make the comparison fair as the models do not have the same vocabulary coverage, this measure is often used [Kozielski & Nuhn$^+$ 14, Jozefowicz & Vinyals$^+$ 16, Hwang & Sung 17] in order to compare language models defined over different subword units.

For the sake of illustration, Table 3.6 shows character level perplexities of 10 K BPE level and 200 K word level language models on LibriSpeech. The general trend when this measure is used, is that the subword level models give higher perplexity compared with its word level counterpart, which is also confirmed here.

In addition to these differences, we will also see, later in Chapter 4, some differences in terms of optimal model parametrizations, as well as some emphasize of the length effect, as the subword level tokenization results in longer sequences than the word level ones.

It should also be noted that many benchmark tasks for language modeling in the more general machine learning community are based on character level [Al-Rfou & Choe$^+$ 19].

Table 3.6: *Character level perplexities of word-level and BPE-level LSTM language models on LibriSpeech.*

Unit	Char PPL	
	Dev	Eval
200K Word	**2.24**	**2.25**
10K BPE	2.35	2.36

Note. This is the end of the introduction on the essentials on the current state-of-the-art LSTM language models. The next section is a short research detour into integrating highway connections in language models, which in the end, did not end up in our core recipe. However, the study gives some intuition on highway connections which are directly related to residual connections which will be a crucial component in the Transformer language models developed in Chapter 4.

3.1.6 A Brief Detour into Highway Connections

The success and development of LSTM have not only contributed in direct improvements in many applications, it has opened room for creativity in designing neural networks with multiplicative gates which are the innovative design from the LSTM. Most notable of them is the highway feed-forward layer [Srivastava & Greff$^+$ 15a].

In this final section on the LSTM language modeling preliminary, we deviate shortly from the LSTM based models themselves, and we briefly present an investigation in neural language modeling by using highway connections for both feed-forward and recurrent models, which was a natural research detour while working on LSTM language modeling at the time.

Preliminary experiments with feed-forward models. While earlier works [Miller & Giles 93] motivated multiplications to make higher-order neural networks, recent works used them as a means to control the information flow inside the network. The feed-forward generalization of the LSTM, the highway network [Srivastava & Greff$^+$ 15a, Srivastava & Greff$^+$ 15b] has gates to ensure the unobstructed information flow across the depth of the network.

A commonly used highway layer is defined to transform its input vector x by:

$$h = \sigma(W_h x + b_h) \tag{3.12}$$

$$g = \sigma(W_g x + b_g) \tag{3.13}$$

$$y = g \odot h + (1 - g) \odot x \tag{3.14}$$

where W_h and W_g are weight matrices, b_h and b_g are biases.

The transformed feature h (Eq. (3.12) is interpolated (Eq. 3.14) with the untransformed feature x using weights g which are learned as a neural network (Eq. 3.13).

The original motivation of this architecture was to ensure an unobstructed information flow between adjacent layers via linear connection, called highway connection which is the second term in the right-hand side of Eq. 3.14. In [Srivastava & Greff$^+$ 15b], it has been shown that such an architecture effectively enables the training of very deep networks (up to 900 layers). However, in practice for language modeling, the benefit for such a connection has been reported for models with much fewer layers. In [Kim & Rush 16b], the highway is used in language modeling as a means to combine the word-level feature with character-level local features; while using only two layers of the highway, the improvements in perplexity were reported.

After all, the highway can also be seen as a pure feature combination operation between the features from different stages of transformation.

The highway connection is related to more elementary gating which can be found in tensor networks [Yu & Deng$^+$ 13], also known as the lateral network [Devlin & Quirk$^+$ 15]. It can also been seen as a gated version of residual networks [He & Zhang$^+$ 16a]. The equations for a lateral network can be obtained by using Eqs. (3.12 - 3.13) and:

$$y = g \odot h \tag{3.15}$$

This can be also seen as a variant of maxout networks [Goodfellow & Warde-Farley$^+$ 13] with two groups, which is obtained by redefining the $\odot$ operation as an element-wise maximum operation instead of a product. In [Devlin & Quirk$^+$ 15], this model has been evaluated for language modeling and has been shown to outperform its variant based on the maximum operation.

We show experimental results for the Quaero task (A.3). We trained 20-gram feed-forward models with a projection layer of 300 units per word and multiple stacked hidden layers of 600 units. In addition to the standard sigmoid function, the exponential linear unit (ELU) [Clevert & Unterthiner$^+$ 16] was also tested.

Table 3.7 shows the performance of different layer types for models with 2 layers. The first layer after the projection layer of the highway model (Highway) is the standard feed-forward layer. The perplexities of all layer types were about the same except the lateral network which performed slightly better.

In order to assess the effect of the highway connection in deep models, we increased the number of layers until five: Table 3.8 shows perplexities on the development set. First of all, we observed that the performance of baseline feed-forward model (Sigmoid) saturated at four layers, while no degradation was observed for highway models (Highway) until five layers. Furthermore, the highway models performed 4% relative better than the baseline. The lateral network saturated with 3 layers and its best perplexity was slightly worse than that of the highway model. This result illustrate the importance of the highway linear connection for training deeper models[8], since the lateral network only differs from the highway network because of the corresponding connection.

Table 3.7: *Comparison of different feed-forward layer types. Perplexities are reported with **2-layer** models on Quaero development set.*

Model	Perplexity
Sigmoid	126.4
Highway	126.5
ELU	126.3
Lateral	**123.4**

Table 3.8: *Effect of the depth. Perplexities on Quaero development set.*

Model	Number of layers			
	2	3	4	5
Sigmoid	126.4	124.9	**124.6**	126.7
Highway	126.5	120.4	119.8	**119.7**
Lateral	123.4	**122.0**	122.2	-

[8] In the following main chapters of this thesis, we will see that these models are anyway not that deep, and the residual connections are better alternatives for training deeper language models.

Incorporating Highway into Gated RNNs. The highway network was originally introduced for feed-forward models. However, its motivation of creating a shortcut connection through the depth of the network also applies to deep recurrent networks.

Many works [Zhang & Chen$^+$ 16, Yao & Cohn$^+$ 15] suggested the extension of stacked LSTMs with additional linear connections between memory cells of adjacent LSTM layers. This is a natural extension for the LSTM, since its memory cell has already linear connection over time (Eq. 3.3). In [Zhang & Chen$^+$ 16], such an architecture has been used for acoustic modeling and has been shown to outperform the standard LSTM, especially in the context of discriminative training. The proposed LSTM architecture, depth-gated LSTM or highway LSTM is obtained by adding a gated dependency to the cell state $c_t^{(\ell-1)}$ of the predecessor LSTM layer, to the original Eq. (3.3):

$$c_t^{(\ell)} = i_t \odot y_t + f_t \odot c_{t-1}^{(\ell)} + d_t \odot c_t^{(\ell-1)} \tag{3.16}$$

where the *depth gate* is computed as

$$d_t = \sigma(W_d x_t^{(\ell)} + w_1 \odot c_{t-1}^{(\ell)} + b_d + w_2 \odot c_t^{(\ell-1)}) \tag{3.17}$$

if the predecessor layer $(\ell-1)$ is also an LSTM layer, otherwise the direct connection to the input $x_t^{(\ell)}$ is used:

$$c_t^{(\ell)} = i_t \odot y_t + f_t \odot c_{t-1}^{(\ell)} + d_t \odot x_t^{(\ell)} \tag{3.18}$$

$$d_t = \sigma(W_d x_t^{(\ell)} + w_1 \odot c_{t-1}^{(\ell)} + b_d) \tag{3.19}$$

where W_d is a weight matrix, w_1 and w_2 are weight vectors, and b_d is a bias vector.

By construction, the hidden layer size in the layer ℓ and its predecessor layer $(\ell-1)$ should match otherwise a projection layer is needed in addition.

This extension is specific to the LSTM which has an internal memory cell, in addition to the standard RNN state. In contrast, we investigated a direct application of the highway operation, which can be used for any RNNs, therefore both GRU and LSTM. Here, we give an example description for LSTMs. Since the highway layer consists of an interpolation of transformed and untransformed features Eq. (3.14), the transformation part Eq. (3.12) can be replaced by any other operation, for example by an LSTM:

$$(h_t, c_t) = \text{LSTM}(y_{t-1}, c_{t-1}, x_t) \tag{3.20}$$

$$g_t = \sigma(W_g x_t + R_g y_{t-1} + b_g) \tag{3.21}$$

$$y_t = g_t \odot h_t + (1 - g_t) \odot x_t \tag{3.22}$$

where W_g and R_g are weight matrices, and b_g is a bias vector.

As can be seen from the equations above, by replacing the LSTM by GRU, we obtain the GRU version of the model.

In [Irie & Tüske$^+$ 16], we focused on evaluating this approach for the GRU. The perplexities are presented in Table 3.9. The standard GRU got degradation with more than two layers while GRU-Highway allowed deeper structures and achieved a 4% relative improvement from 110.7 to 106.3 for model with 300 nodes. However, the overall gain was too marginal, to make it a part of our default recipe for building the best neural language model at that time.

After publication of [Irie & Tüske$^+$ 16], we investigated an even simpler highway connection to be applied for any RNNs. We made use of feed-forward highway connections and did not include the output after gating in the recurrent dependency, as illustrated for the LSTM below:

$$(h_t, c_t) = \text{LSTM}(h_{t-1}, c_{t-1}, x_t) \tag{3.23}$$

$$g_t = \sigma(W_g x_t + b_g) \tag{3.24}$$

$$y_t = g_t \odot h_t + (1 - g_t) \odot x_t \tag{3.25}$$

Table 3.9: *Perplexities on Quaero development set. The number of hidden units are set to 300 in each layer.*

Model	Number of layers		
	2	3	4
GRU	**110.7**	114.5	116.4
GRU-Highway	109.1	**106.3**	106.6

where W_g is a weight matrix and b_g is a bias vector.

We experimented with this model in our best Quaero LSTM model configuration. Table 3.10 shows the perplexity performance. While this result illustrates again the limited benefit of this approach, we made use of this type of LSTM-Highway in our **CHiME-4 evaluation** system [Menne & Heymann$^+$ 16], where even small improvements could count. Also, we made use of the combination of character-level convolutional layer and the highway connections proposed by [Kim & Rush 16b] in our byte-level version [Irie & Golik$^+$ 17] to improve speech recognition and key word search in low resource conditions [Golik & Tüske$^+$ 17] in the context of **IARPA BABEL project**.

Table 3.10: *Perplexities on Quaero set. The number of hidden units are set to 2048 in each layer. Dropout of 20% is used.*

Model	Num. layers	Perplexity	
		Dev	Eval
LSTM	2	84.5	86.3
LSTM-Highway	4	**81.6**	**83.1**

3.2 Attention in Language Modeling: Shallow Attempts

In this section, we briefly introduce attention [Bahdanau & Cho$^+$ 15] and some of the elementary language models which are related to, or based on attention.

There has been a couple of primitive tentatives in applying the attention mechanism to language modeling including ours [Irie & Tüske$^+$ 16], in the time between the emergence of the attention mechanism [Bahdanau & Cho$^+$ 15] and the advent of the Transformer architecture [Vaswani & Shazeer$^+$ 17] which has established a general purpose usage of attention.

We briefly presents these precursors of Transformer language models which will be fully discussed in Chapter 4. This will not only allow us to provide some historical backgrounds, but also by contrast to emphasize the origin of the power of Transformer architecture and to some extent the necessity for a deep architecture.

3.2.1 Attention

The attention in a neural network [Bahdanau & Cho$^+$ 15] is an operation which takes two sets of paired vectors, keys $\mathcal{K} = (k_1, ..., k_N) \in \mathbb{R}^{N \times d_{\text{key}}}$ and values $\mathcal{V} = (v_1, ..., v_N) \in \mathbb{R}^{N \times d_{\text{value}}}$ with the same number of vectors N with respective dimensions of d_{key} and d_{value}, and one query vector $q \in \mathbb{R}^{1 \times d_{\text{key}}}$ as inputs. While there are a couple of variants of attention [Luong & Pham$^+$ 15, Chan & Jaitly$^+$ 16], the most popular variant today is arguably the dot attention, which computes for each $1 \leq i \leq N$,

$$s_i = k_i \bullet q \quad \text{where } \bullet \text{ denotes the dot product.} \tag{3.26}$$

$$\alpha = \text{softmax}(s) \quad \text{where } s = (s_1, .., s_i, .., s_N). \tag{3.27}$$

$$\text{Attention}(\mathcal{K}, \mathcal{V}, q) = \sum_{i=1}^{N} \alpha_i v_i \quad \text{where } \alpha_i \text{ are defined as } \alpha = (\alpha_1, .., \alpha_i, .., \alpha_N). \tag{3.28}$$

which can be squeezed into matrix operations:

$$\text{Attention}(\mathcal{K}, \mathcal{V}, q) = \text{softmax}(q\mathcal{K}^\intercal)\mathcal{V} \tag{3.29}$$

Other variant such as MLP attention [Bahdanau & Cho$^+$ 15] makes use of a feed-forward layer to parametrize the scores in Eq. (3.26).

This mechanism has been first proposed for machine translation [Bahdanau & Cho$^+$ 15] to augment the plain sequence-to-sequence [Sutskever & Vinyals$^+$ 14] learning with neural networks, by allowing the model to only focus on a portion of the source text while generating a portion of target text. This remediated the memory bottleneck problem of recurrent neural networks[9].

The rest of this section presents two elementary language models which are related to this attention mechanism: weighted bag-of-words and neural word triggers.

3.2.2 Bag-of-Words

As can be seen from Eq. (3.28) in the definition of attention above, the final operation of attention consists of a weighted average of value vectors, which is also a common operation in a bag-of-words in natural language processing. Therefore, attention can be seen as a bag-of-concepts with adaptive weights parametrized by a differentiable function.

This section presents our preliminary investigation [Irie & Schlüter$^+$ 15] on the usage of bag of words (BOW) in neural language modeling. In [Irie & Schlüter$^+$ 15], we investigated the bag-of-words as an alternative to RNNs to obtain fixed-size vector representation of word sequences,

[9] While ideally we might also want to have a more powerful recurrent neural networks which do not require such a prior knowledge!

in the objective of augmenting n-gram feed-forward neural language models with longer term dependencies. This will serve us as an illustration for the power of simple weighted averaging in neural language modeling.

The bag-of-words of a word sequence is simply defined as the sum of 1-of-N representation of all words in the sequence. We simply use such a feature as an extra input by concatenation to the standard neural language model, in the style of [Mikolov & Zweig 12]. Equivalently, it corresponds to providing the model with an extra input feature computed as the sum of word embedding vectors of words in the *bag*, i.e. a context window of last L words (where L is much larger than the standard n-gram window, 50 in our experiments).

In this framework, the bag-of-words vector B_t with a context size L, is defined at each time step as:

$$B_t = \sum_{i=0}^{L-1} w_{t-i} \tag{3.30}$$

$$= B_{t-1} + w_t - w_{t-L} \tag{3.31}$$

where w_t denotes the word embedding for the word at position t. In our experiments, we made use of an extra word embedding matrix which is separate from the embedding matrix for the standard input word.

While this can be done for both n-gram feed-forward language models and LSTM language models, this approach has a clear motivation for the feed-forward case. In fact, the input vector to the feed-forward network representing the n-gram context is build by concatenating $(n-1)$ word embedding vector. Therefore, adding more words in the context of an n-gram feed-forward language model increases the input vector size linearly. This is not appealing when considering a long context length, such as 50-gram. The bag-of-word representation can cover as many predecessor words as one wishes without a need for scaling up the model size. Like an RNN, it is a fixed size representation independent of the context size. Also, for LSTM language models, while theoretically the model has no problem handling the full word history, the question whether alternative representation of long context as an extra input feature helps was unknown.

On the other hand, the drawback of the bag-of-words is that, because of the sum, the word sequence order information is lost. A similar problem has been pointed out in [Clarkson & Robinson 97] for the cache language model [Kuhn & De Mori 90]. They have introduced an exponential decay to express the distance of predecessor words from the current word. We adopted this approach for the bag-of-words, by defining the bag-of-words vector with decay, $B_t^{(\text{decay})}$ as follows:

$$B_t^{(\text{decay})} = \sum_{i=0}^{L-1} \gamma^i w_{t-i} \tag{3.32}$$

where γ is the decaying factor in [0,1]. We therefore end up having a weighted bag-of-words, where the *attention* was manually designed in such a way that it gives more importance to the recent words, to be compared with Eq. (3.28). We also note that, as opposed to attention, the weights do not normalize to one.

Experimental results. We conducted the experiments on the Quaero task using the preliminary ASR baseline system (Appendix A.3).

All neural language models were trained with the projection layer size of 300 for each word. We set the bag-of-words' projection layer size such that it has the same size as the feature of one word, while preserving the total size of the projection layer. A bag-of-words with a context size of 50 words and with decay of 0.9 was used. In [Irie & Schlüter$^+$ 15], we found that the use of the decay weights in the bag-of-words were crucial: all bag-of-word feature here thus makes use of

the decaying weights. We trained bigram, 4-gram, 10-gram feed-forward and LSTM-RNN models with the bag-of-words input. Table 3.11 shows the perplexity results. It shows that the bag-of-words input does not improve the LSTM models, while improving all n-gram feed-forward models. It is noteworthy that the 4-gram feed-forward model with the bag-of-word input performed best among the feed-forward models, and the gap from the LSTM language model is largely reduce by the introduction of long contexts via bag-of-word features. As shown in Table 3.12, these improvements in perplexity carries over to improvements in WER.

Table 3.11: *Perplexity results on Quaero for neural language models with an additional bag-of-words input feature. All models including the 4-gram Kneser-Ney model are trained on 50 M words for comparison. A hidden layer size of 500 is used.*

		Bag-of-words	Dev	Eval
4-gram Kneser-Ney		-	163.0	160.3
Feed-forward	Bigram	No	212.3	205.0
		Yes	**140.1**	**140.5**
	4-gram	No	149.6	145.0
		Yes	**125.9**	**125.7**
	10-gram	No	138.2	136.6
		Yes	**128.1**	**128.1**
LSTM-RNN		No	**110.5**	**110.1**
		Yes	111.9	112.5

Table 3.12: *Perplexity and WER (in %) results on Quaero for neural language models with an additional bag-of-words input feature. Perplexities are those of models interpolated with the 4-gram Kneser-Ney model trained on 3.1 B.*

		Bag-of-words	Dev		Eval	
			PPL	WER	PPL	WER
4-gram Kneser-Ney		-	132.7	13.9	131.2	11.7
Feed-forward	Bigram	No	130.5	13.8	128.1	11.7
		Yes	111.4	13.2	111.1	11.0
	4-gram	No	120.9	13.3	118.7	11.2
		Yes	107.0	**13.0**	106.8	**10.8**
	10-gram	No	113.8	13.1	112.7	11.1
		Yes	107.6	13.1	107.2	10.9
LSTM-RNN		No	98.4	**12.5**	97.7	**10.2**

Related works and follow-ups. We also note that the weighted bag-of-words model is a special case of standard recurrent neural network with the recurrent matrix reduced to one scalar. Similar approaches have been proposed in [Mikolov & Joulin$^+$ 15, Zhang & Jiang$^+$ 15] independent of our work around the same time. In particular, it is noteworthy that the model proposed by [Zhang & Jiang$^+$ 15] under the name fixed-size ordinally-forgetting encoding (FOFE) method, which is purely based on the weighted bag of words (without any n-gram feature as in our model), has had a series of follow up extension works [Zhang & Jiang$^+$ 16, Zhang & Liu$^+$ 17].

3.2.3 Attention for Learning Word Triggers

Motivated by both the success in machine translation and a better visualization that the attention mechanism [Bahdanau & Cho$^+$ 15] offers, it was natural to try to find a way to apply the same technique to language modeling.

A number of works [Tran & Bisazza$^+$ 16, Cheng & Dong$^+$ 16] have investigated such a possibility before the introduction of Transformers [Vaswani & Shazeer$^+$ 17]. Most notably, [Cheng & Dong$^+$ 16] has introduced *self-attention* mechanism which has later become the core component of the Transformer architecture.

In this section, we briefly present our efforts motivated by the same spirit at the time. In [Irie & Tüske$^+$ 16], we investigated two simple approaches for making use of attention in language modeling. In both models, our main motivation was the analogy to the (multi-)*word triggers* [Tillmann & Ney 97, Rosenfeld 96] in the count-based approach. Certain words in the context can be particularly relevant to predict some words. Our objective was to create a neural version of such an approach by using attention, and to visualize the triggering effect via attention weights. More specifically, we tried to integrate such a mechanism into recurrent language models.

Recurrent attention layer. First of all, we defined a minimalistic attention layer (which today looks obsolete and awkward as we are aware of Transformers...) which computes at time step t:

$$s_i = w^\intercal \tanh(W x_i + R y_{t-1} + b) \qquad \text{for each } i,\ 1 \leq i \leq t. \tag{3.33}$$

$$\alpha = \text{softmax}(s) \quad \text{where } \ s = (s_1, .., s_i, .., s_t). \tag{3.34}$$

$$y_t = \sum_{i=1}^{t} \alpha_i x_i \qquad \text{where } \alpha_i \text{ are defined as } \alpha = (\alpha_1, .., \alpha_i, .., \alpha_t). \tag{3.35}$$

where W and R are weight matrices, w is a weight vector, and b is a bias vector. The input at time t is the outputs of the previous layer over time $(x_1, ..., x_t)$. It computes a scalar score s_i for each context x_i (Eq. 3.33). The resulting score vector $s = (s_1, ..., s_t)$ is then normalized (Eq. 3.34) and the output is computed as the weighted average of contexts (Eq. 3.35).

Neural word trigger models, two approaches. We inserted such an attention layer to a simple language model composed of 3 layers: the input embedding layer, a GRU (as a simple, generic recurrent layer[10]) and output layers. Given this model, the attention layer can be inserted either between the input embedding layer and the GRU layer, or between the GRU and output layer. The two approaches are illustrated in Figures 3.1 and 3.2. We used the attention limited on a local window of 19 predecessor words (20-gram) [Luong & Pham$^+$ 15], which we found to give better perplexities than the unlimited-size window. We used a hidden layer size of 300 for both the input embedding and the GRU layers.

[10] Today, we would simply use an LSTM. These experiments have been conducted at the time when the small speed improvements offered by the GRU were appealing for a fast testing of the idea, as we were training these models on CPUs.

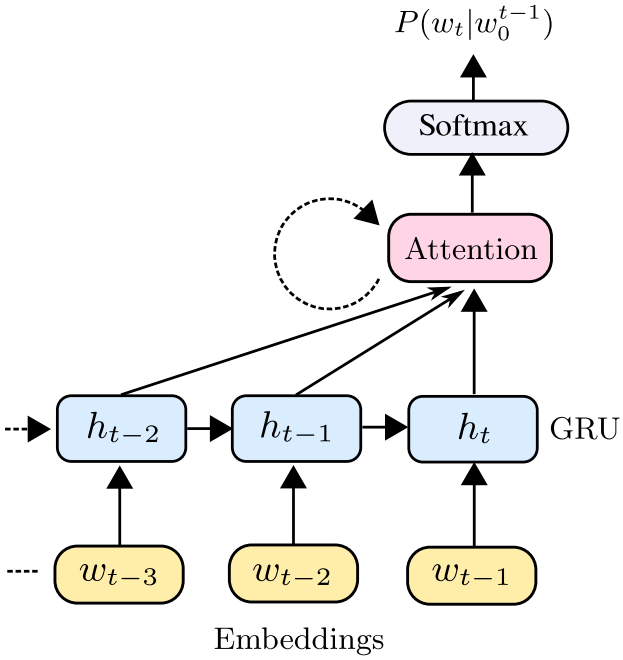

Figure 3.1: *Model of type: Attention after the recurrent layer. No trigger is obtained, the model chooses the most recent context from the GRU. Quaero development perplexity of 109.1, which is similar to 110.6 of the model without the attention layer.*

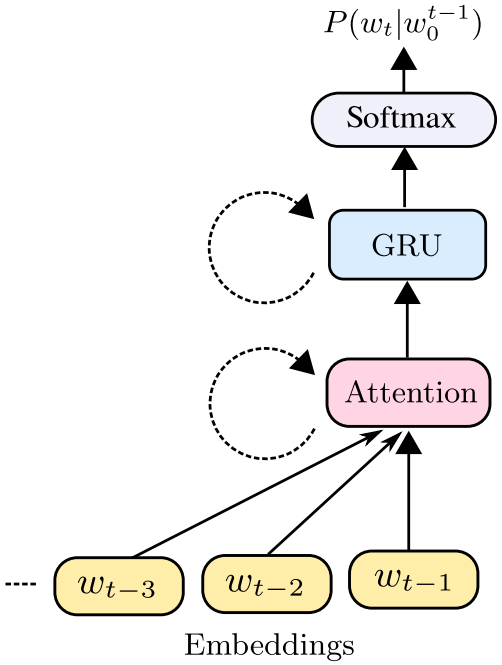

Figure 3.2: *Model of type: Attention before the recurrent layer. Some triggers can be observed, but the perplexity is bad: 157.6 which is close to the perplexity of 4-gram model, 163.0.*

Results. We conducted our experiments on the Quaero dataset. The experimental result has shown that the latter model, i.e. the model with an attention layer after the GRU layer (Figure 3.1), was not suited for the word trigger. This was found to be the case, because in such a model, the attention layer exclusively uses the latest state of the GRU ($\alpha_t \approx 1$ in Eq. (3.35)) which has seen the full context, therefore containing the largest amount of information among other states presented to the attention layer[11]. The model gave a development perplexity of 109.1, which is roughly the same as 110.6, which is the perplexity of the model with the same architecture without the attention layer (which is not a surprise, since the attention layer in this model had learned to consistently select the latest RNN state).

Therefore, with the objective of observing word triggers in mind, we focused on the former case, in which the attention layer directly follows the embedding layer (Figure 3.2). The attention layer in this case is therefore a bag-of-word layer (Sec. 3.2.2) with context dependent weights.

Such a model achieved a development perplexity of 157.6, which is only slightly better than the 4-gram count model trained on the same amount of data (with a development perplexity of 163.0), and much worse than the baseline GRU (110.7).

Despite this relatively high perplexity, some qualitatively meaningful triggers could be observed in some sentences. Examples are shown in Figure 3.3.

Furthermore, contrary to the tendency of count-based triggers [Rosenfeld 96], we did not find self-triggers to be common.

While we found these results qualitatively interesting, the performance of this naive model was not satisfactory. The choice of query vector (here the recurrent state h_{t-1}) in the score function (Eq. 3.33) is likely to turn out to be a bad one[12]. A contemporary approach [Tran & Bisazza$^+$ 16] which makes use of attention over an external memory to the LSTM, has been shown to be more successful in augmenting the LSTM language model with an attention mechanism at that time.

$\$^{6}$ **Thanks**^{10} for^{3} taking^{9} the^{2} time^{4} to^{3} **download**^{22} this^{5} **BBC**^{12} **radio**^{11} five^{4} live^{8} *podcast*

$**^{22} In^{4} this^{7} **book^{17} there^{7} are^{5} **things**^{13} that^{7} are^{5} **very**^{14} *complicated*

Figure 3.3: *Two example of attention weights from the model in Figure 3.2. For each sentence, the word inside a box is the target word. The numbers in exponent of the context words are the scores in percentage given by the model to predict the target word. Words with the highest weights (triggers) are highlighted with* ***bold*** *font. $ denotes sentence begin token.*

[11] We note that this observation played a key role in getting an intuition that the Transformer language model does not require positional encoding, as we will show later in Chapter 4.

[12] To be compared with the design of the successful self-attention, later in Chapter 4.

3.3 Correlation Between Perplexity and Word Error Rate

The two previous preliminary sections respectively have introduced the essential concepts in state-of-the-art LSTM language modeling and some elementary language models using attention.

In this section, we introduce one last concept which is of crucial nature for this thesis, as is the case for any study on language modeling for automatic speech recognition: the correlation between perplexity and word error rate. Such a correlation has been claimed since the introduction of perplexity as an evaluation measure for language models [Jelinek & Mercer$^+$ 77], which have been later confirmed experimentally, across different tasks [Bahl & Jelinek$^+$ 83, Makhoul & Schwartz 95, Chen & Beeferman$^+$ 98, Klakow & Peters 02, Sundermeyer & Ney$^+$ 15, Halpern & Hall$^+$ 16].

Here, we empirically illustrate this correlation using two approaches. The first one makes use of multiple models to show a correlation between perplexity and word error rate on the corpus level (which is useful in practice), and in the second approach, we demonstrate that the perplexity-word error rate correlation also stands when we look at local data points on the word level from a single model.

It should be noted that the correlation assumes a decent search, since bad and aggressive pruning could mask the effect of an improved language model, and that we must compare models under the *comparable* conditions. In particular, once we start considering the long context beyond sentence boundaries [Clarkson & Robinson 98], we must be aware that, first pruning becomes typically more aggressive, and second the improvements might not be uniform over positions: therefore extrapolating the same correlation law by mixing perplexities of models which make use of contexts beyond sentence boundaries with those which do not, typically is not successful. Being careful with these potential caveats, we can measure the quality of language models by the perplexity. Throughout this thesis, we generally find this correlation to be good, and will comment on it when it is not the case.

3.3.1 Corpus level Correlation Using Multiple Models

In the old rule of thumb of language modeling [Makhoul & Schwartz 95], practitioners tell us to expect *"5% relative improvements in terms of WER when we get 10% relative improvements in perplexity"*. Such an empirical rule turns out to be rather good, while the number *5%* depends on the task. For instance, we find that that number is rather close to 4% in the case of Quaero dataset (Appendix A.3) as illustrated by Figure 3.4. In Figure 3.4, 39 language models of different nature (10-gram feed-forward models, vanilla RNN, LSTM, and count language models) are used to generate data points by extending the work by [Sundermeyer & Ney$^+$ 15]. We obtain the correlation equation of $\log(\text{WER}) = 0.62 + 0.39 * \log(\text{PPL})$.

A similar experiment has been conducted for the LibriSpeech dataset (Appendix A.1). The result for the dev-clean dataset is shown in Figure 3.5. All count model data points are generated by applying pruning to the official 4-gram model distributed with the LibriSpeech dataset[13]. All count models are 4-grams. The LSTM and Transformer neural language models which are later developed in Chapter 4 (Table 4.17) are used. The correlation equation of $\log(\text{WER}) = -0.79 + 0.40 * \log(\text{PPL})$ is obtained.

[13] We thank Wei Zhou for running multiple decoding experiments with these count models.

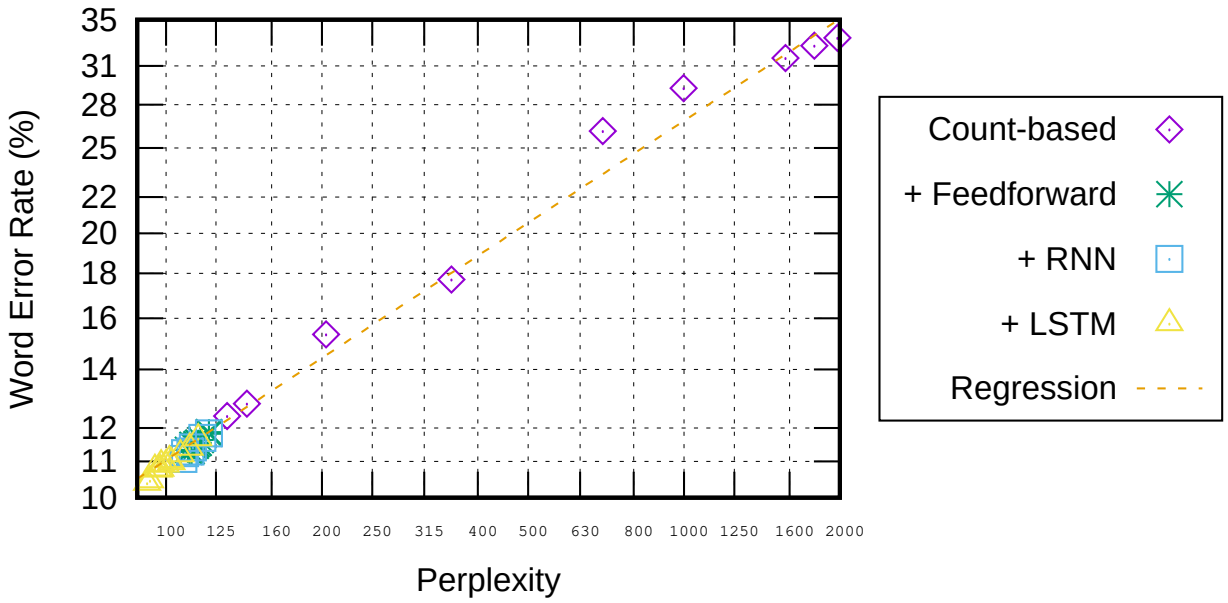

Figure 3.4: *Correlation between perplexity and word error rate using the preliminary ASR system for* ***Quaero*** *(A.3). Both axes are on the natural log scale. The regression has the equation:* $log(WER) = 0.62 + 0.39 * log(PPL)$.

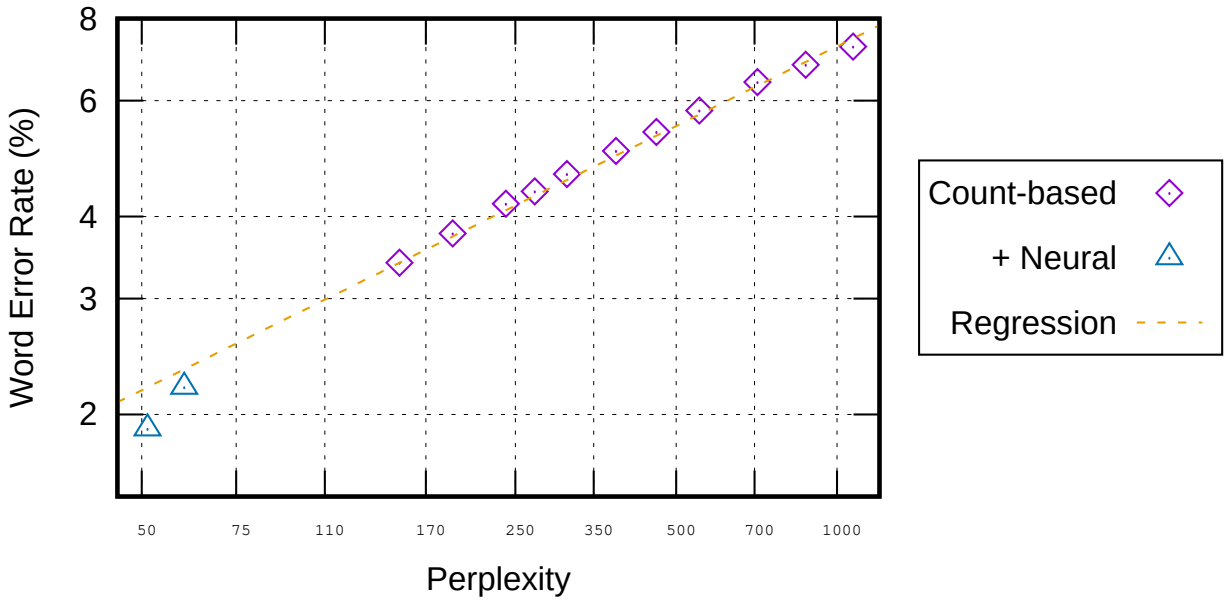

Figure 3.5: *Correlation between perplexity and word error rate using the hybrid NN-HMM ASR system for* ***LibriSpeech*** *on the dev-clean subset (A.1). Both axes are on the natural log scale. The regression has the equation:* $log(WER) = -0.79 + 0.40 * log(PPL)$.

3.3.2 Local Correlation Using One Model

While the perplexity and word error rate are both typically computed on the whole corpus, their computation is based on an *averaging* of their corresponding word-level quantities. Therefore, if we instead average within smaller partitions of the whole data, we can obtain multiple data points from a single recognition. In this section, we are interested in testing the correlation, by making use of such local perplexities and local word error rates, which we can generate from a single recognition run, as follows.

For perplexities, the local quantity at each word position in the transcription is the language model (log) probability. To smooth the statistics locally, we average the log probabilities over a centered, overlapping window at each word position, to obtain the final local probabilities. We truncate the left or right part of the window, for the positions close to the beginning and the end of a sentence.

For word error rates, we first obtain an error count for each word position in the transcription, by using the Levenshtein alignment with the recognition output. Each substitution and deletion error is assigned to the corresponding word position in the transcription. For an insertion error, we assign it to the word position following the error. We include the sentence end positions in the statistics.

We thereby obtain (log probability, error count)-pair values at each word position in the transcription. We then sort these pairs by log probability values, and group into bins of equal size (in practice, about 2000 word positions per bin). By averaging within each bin, we finally obtain the data points consisting of perplexity and word error rate.

Figure 3.6 illustrates the correlation using such data points generated using the linear interpolation between the best LSTM and 4-gram language for the same Quaero experiment as in the section above. A sliding window of +/- one word is used for smoothing of local perplexities. We empirically observe that the correlation also seems rather good using such statistics. The similar trend is also obtained for the TED-LIUM 2 dataset as illustrated in Figure 3.7.

3.4 Summary

In this preliminary chapter, we introduced basic concepts in language modeling for speech recognition, which are fundamental for other chapters in this thesis. We presented the core aspects in developing strong baseline LSTM language models; we discussed practical tuning of the model size and regularization (which is further put into practice in Chapter 4), training and evaluation consistency (further discussed in Chapter 7), domain adaptation (which is a crucial aspect for methods presented in Chapter 6), modeling unit as well as effect of small model extensions illustrated with the example with highway networks. We also introduced some preliminary ideas and models to apply attention in language modeling, which is to be contrasted with the successful Transformer architecture (in Chapter 4). Finally, we illustrated the correlation between language model perplexity and word error rate, which is a fundamental empirical result.

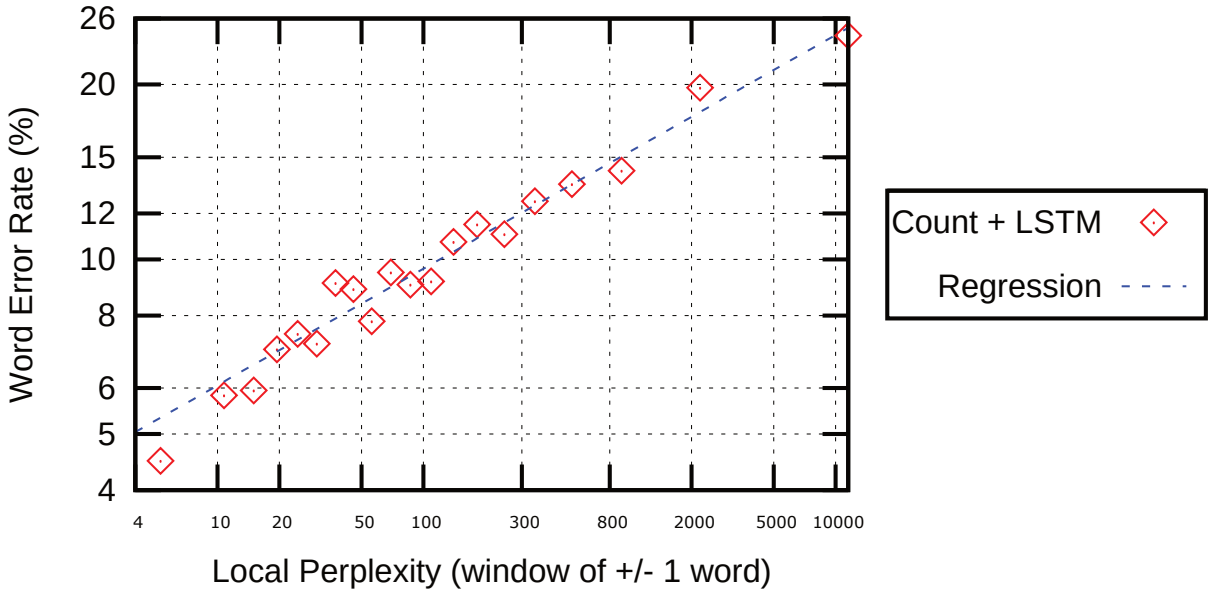

Figure 3.6: Correlation between perplexity and word error rate; using the preliminary ASR system for Quaero (A.3). Both axes are on the natural log scale. The regression has the equation: $\log(\text{WER}) = 1.34 + 0.20 * \log(\text{PPL})$.

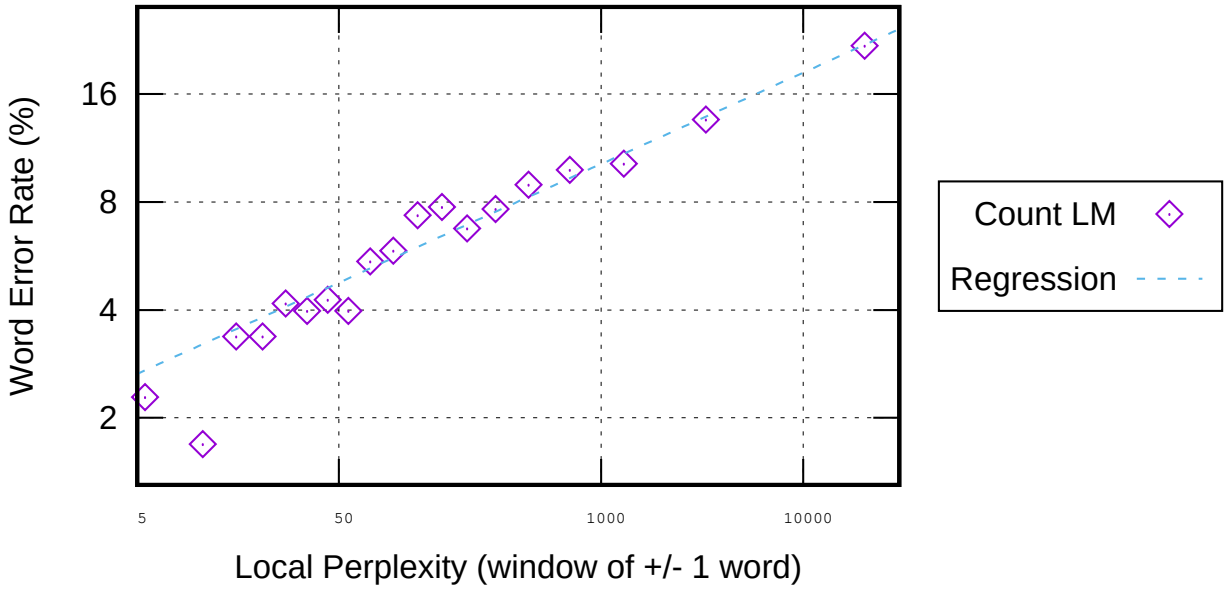

Figure 3.7: Correlation between perplexity and word error rate for TED-LIUM 2 (A.2) using the 4-gram count language model. Both axes are on the natural log scale. The regression has the equation: $\log(\text{WER}) = 1.76 + 0.25 * \log(\text{PPL})$.

4. State-of-the-art ASR Language Modeling with Transformers

Transformer encoder-decoder models [Vaswani & Shazeer$^+$ 17] have become popular in natural language processing. The Transformer architecture allows to successfully train a stack of *self-attention* layers [Cheng & Dong$^+$ 16, Lin & Feng$^+$ 17, Parikh & Täckström$^+$ 16] via residual connections [He & Zhang$^+$ 16a] and layer normalization [Ba & Kiros$^+$ 16]. The positional encodings [Vaswani & Shazeer$^+$ 17, Gehring & Auli$^+$ 17], typically based on sinusoidal functions, are used to provide the self-attention with the sequence order information. Across various applications, systematic improvements have been reported over the standard, multi-layer long short-term memory [Hochreiter & Schmidhuber 97] recurrent neural network based models. While originally designed as an encoder-decoder architecture in machine translation, the *encoder* (e.g., [Devlin & Chang$^+$ 19]) and the *decoder* (e.g., [Liu & Saleh$^+$ 18]) components are also separately used in corresponding problems depending on whether the problem disposes the whole sequence for prediction or not.

A number of recent works have also shown impressive performance in language modeling using the Transformer *decoder* component [Liu & Saleh$^+$ 18, Dai & Yang$^+$ 19, Al-Rfou & Choe$^+$ 19, Baevski & Auli 19, Radford & Narasimhan$^+$ 18, Radford & Wu$^+$ 19]. While the first application of Transformers for language modeling (for text generation) goes back to [Liu & Saleh$^+$ 18], Al-Rfou et al.'s work [Al-Rfou & Choe$^+$ 19] was the first to have started scaling up the Transformer for language modeling, by making it as deep as 64 layers, and has shown that it can be competitive with the state-of-the-art LSTM based models for character based language modeling tasks. More recently, OpenAI's GPT-2 model [Radford & Wu$^+$ 19] has shown further potential of larger and deeper Transformer language models.

The work presented in this chapter was motivated by the similar spirit of scaling up Transformers for language modeling, but with a specific purpose of pushing the state-of-the-art language modeling in automatic speech recognition.

In the following Sec. 4.1, we present the development of our deep Transformer language models with a successful application to large scale ASR. We revisit the parameter configurations of Transformers, originally engineered for the sequence-to-sequence problem, specifically for language modeling. We demonstrate that well configured Transformer language models outperform models based on the simple stack of LSTM RNN layers in terms of both perplexity and word error rate.

In Sec. 4.2, we reconsider and analyse the Transformer architecture for language modeling. In an autoregressive problem, as is the case for language modeling, where a new token is provided to the model at each time step, the amount of information the model has access to strictly increases from left to right at the lowest level of the network, which should provide some positional information by its own. We observe that deep Transformer language models without positional encoding automatically make use of such information, and even give slight improvements over models with positional encodings. In addition, by visualizing the attention weights, we reveal the functionality

of each layer, which we will relate to fundamental concepts in language modeling.

Another challenge with Transformers for language modeling in speech recognition is the memory requirement, as they are more demanding than an LSTM language model. Their memory requirement linearly increases in terms of number of tokens in the sequence. In Sec. 4.3, we propose a simple architectural re-organization of the Transformer layer to alleviate this problem.

Finally, we present an overview comparison between LSTM and Transformer language models across 6 datasets in Sec. 4.4.

4.1 Deep Transformers for Language Modeling

In this section, we follow the spirits of Al-Rfou et al.'s work [Al-Rfou & Choe+ 19] and Radford et al.'s work [Radford & Narasimhan+ 18, Radford & Wu+ 19] in investigating larger and deeper Transformers for language modeling, with the objective of improving automatic speech recognition.

4.1.1 Transformer Language Models

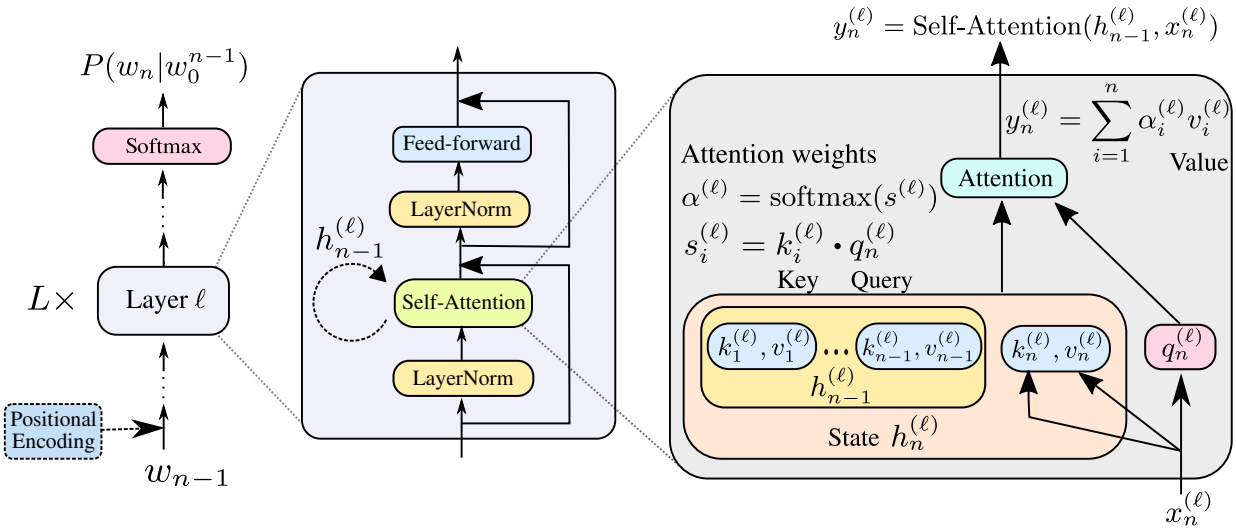

Figure 4.1: Illustration for Transformer language model components.

The Transformer language model is based on the *decoder component* of the Transformer architecture [Vaswani & Shazeer+ 17][1]. The model is depicted in Figure 4.1. Similar to previous works [Liu & Saleh+ 18, Radford & Narasimhan+ 18, Baevski & Auli 19, Al-Rfou & Choe+ 19, Dai & Yang+ 19, Radford & Wu+ 19], we define *layer* as a stack of two components: *self-attention* and *feed-forward*[2] modules.

[1] In principle, we could also consider using the *encoder* component for an autoregressive self-attention model which updates states at all predecessor positions for each new input. Such a model would be then much more computationally inefficient, but could also potentially be more powerful.

[2] Typically called *position-wise feed-forward* module [Vaswani & Shazeer+ 17]. Here we omit *position-wise* as it is obvious for autoregressive models.

The autoregressive *self-attention module* in the l-th layer transforms the input $z_n^{(l-1)}$ at position n as follows:

$$\begin{aligned} x_n^{(l)} &= \text{LayerNorm}_1(z_n^{(l-1)}) && (4.1)\\ q_n^{(l)}, k_n^{(l)}, v_n^{(l)} &= Qx_n^{(l)}, Kx_n^{(l)}, Vx_n^{(l)} && (4.2)\\ h_n^{(l)} &= \left(h_{n-1}^{(l)}, (k_n^{(l)}, v_n^{(l)})\right) && (4.3)\\ y_n^{(l)} &= \text{Attention}(h_n^{(l)}, q_n^{(l)}) && (4.4)\\ \tilde{y}_n^{(l)} &= z_n^{(l-1)} + W_0 y_n^{(l)} && (4.5) \end{aligned}$$

where Q, K, V, respectively denote query, key, value projection matrices, LayerNorm_1 denotes layer normalization [Ba & Kiros$^+$ 16], and W_0 denotes the projection matrix for the residual connection [He & Zhang$^+$ 16a]. We omit the layer index (l) on the parameters to avoid heavy notations. The sub-script for LayerNorm is added as each layer normalization (LayerNorm_1 in Eq. (4.1) and LayerNorm_2 in Eq. (4.9)) has its own scaling and bias parameters.

Attention in Eq. (4.4) denotes the *scaled multi-head dot product attention* [Vaswani & Shazeer$^+$ 17] which is an extension to the dot attention we have seen in the preliminary Chapter 3, Eqs. (3.26 - 3.28) in Sec. 3.2. We have $y_n^{(l)} = \text{Attention}(h_n^{(l)}, q_n^{(l)})$, where $h_n^{(l)} = \left((k_0^{(l)}, v_0^{(l)}), ..., (k_n^{(l)}, v_n^{(l)})\right)$. An H-head attention carries out H separate attention operations by splitting each of $k_i^{(l)}$, $v_i^{(l)}$, and $q_n^{(l)}$ vectors into H equally sized sub-vectors. We always choose H such that H is a factor of the key and query dimension d_{key}, as well as the dimension of the value vector. For example, for a key vector $k_i^{(l)} \in \mathbb{R}^{d_{\text{key}}}$ from the position i, $k_i^{(l)} = (k_{i,1}^{(l)}, ..., k_{i,H}^{(l)})$ where $k_{i,j}^{(l)} \in \mathbb{R}^{d_{\text{key}}/H}$ corresponds to the key vector for the head j. For each head j, $1 \leq j \leq H$, we compute:

$$s_{j,i}^{(l)} = k_{i,j}^{(l)} \bullet q_{n,j}^{(l)} \quad \text{for each position } i,\ 1 \leq i \leq n, \text{ where } \bullet \text{ denotes the dot product.} \quad (4.6)$$

$$\alpha_j^{(l)} = \text{softmax}\left(\sqrt{\frac{H}{d_{\text{key}}}}\, s_j^{(l)}\right) \quad \text{where } s_j^{(l)} = (s_{j,1}^{(l)}, .., s_{j,i}^{(l)}, .., s_{j,n}^{(l)}). \quad (4.7)$$

$$y_{n,j}^{(l)} = \sum_{i=1}^{n} \alpha_{i,j}^{(l)} v_{i,j}^{(l)} \quad \text{where } \alpha_{i,j}^{(l)} \text{ are the components of } \alpha_j^{(l)} = (\alpha_{1,j}^{(l)}, .., \alpha_{i,j}^{(l)}, .., \alpha_{n,j}^{(l)}). \quad (4.8)$$

Finally, all resulting vectors $y_{n,j}^{(l)}$ from different heads are concatenated to form the output $y_n^{(l)}$. The operations above from Eq. (4.2) to Eq. (4.4) which transforms $x_n^{(l)}$ into $y_n^{(l)}$ correspond to the self-attention operation $y_n^{(l)} = \text{SelfAttention}(h_{n-1}^{(l)}, x_n^{(l)})$ as illustrated in Fig. 4.1 for the single head case (obtained by removing the index j) and by omitting scaling before the softmax. After linear transformation and the residual connection as in Eq. (4.5), we obtain $\tilde{y}_n^{(l)}$. The output $\tilde{y}_n^{(l)}$ of the self-attention layer is then fed to the *feed-forward* module:

$$\begin{aligned} m_n^{(l)} &= \text{LayerNorm}_2(\tilde{y}_n^{(l)}) && (4.9)\\ z_n^{(l)} &= \tilde{y}_n^{(l)} + W_2\, \text{Activation}(W_1 m_n^{(l)}) && (4.10) \end{aligned}$$

where for Activation, the rectifier linear unit (ReLU) [Nair & Hinton 10], Gaussian error linear unit (GELU) [Hendrycks & Gimpel 18, Radford & Wu$^+$ 19], or gated linear unit (GLU) [Dauphin & Fan$^+$ 17] are investigated in this section. $W_1^{(l)}$ and $W_2^{(l)}$ denote weight matrices. Biases after linear transformations are omitted for clarity.

The final model is built by simply stacking these *layers* multiple times. We thus note that the components in the Transformer language model which learn the temporal dependencies are the self-attention modules.

The input of the network consists of the sum of the token embedding (word or BPE in this thesis) and the sinusoidal *positional encoding* as introduced in [Vaswani & Shazeer+ 17]. A sinusoidal positional encoding e_n representing the position n is a vector of dimension M (which is equal to the dimension of input token embeddings) where each component is computed by:

$$e_{n,2i} = \sin(n/10000^{2i/M}) \tag{4.11}$$

$$e_{n,2i+1} = \cos(n/10000^{2i/M}) \tag{4.12}$$

for $1 < i < \frac{M}{2}$.

Finally, the output softmax layer gives the probability distribution for the next token.

As can be seen in the equation Eq. (4.3) above, $h_n^{(l)}$ above, can be seen as *states* of the Transformer model (whose size, as opposed to the RNN states, linearly grows along the position dimension). During inference, these states are stored to avoid redundant computation. During training, the computation along the position dimension is parallelized for speedup.

4.1.2 Tuning Hyper-Parameters in Transformers

Hyper-parameters in Transformers. The Transformer architecture is a new *search space Odyssey* [Greff & Srivastava+ 17]. The exhaustive model hyper-parameters for a Transformer language model are the input token embedding size, the number of layers, the dimension of the residual connection, and for each layer the number of attention heads, the dimension of the key and query, the dimension of the value, and the dimension of the feed-forward layer.

In order to reduce this complexity, in our experiments, we use the same dimension for key, query and value, as well as the residual connection. We use the same dimensionality across all layers. Therefore, our models can be fully specified by the tuple (**number of layers L, feed-forward dimension $d_{\mathbf{ff}}$, residual dimension $d_{\mathbf{res}}$, number of heads H**).

We carry out our experiments on the LibriSpeech dataset for both word-level and BPE-level language modeling. For illustrating the hyper-parameter tuning, we first focus on the word-level one.

We train all models using the plain stochastic gradient descent and new-bob learning rate tuning on a single GPU. We define our training *sub-epoch* (for Newbob) as the 10-th of the full training data. All our implementations are based on the TensorFlow [Abadi & Barham+ 16] based open-source toolkit RETURNN [Zeyer & Alkhouli+ 18][3].

Given the amount of LibriSpeech training data (850 M running words), it is unreasonable to train all model variants until full convergence. However, we observe that the model performance at some earlier stage of the training is a good indicator of the performance after convergence. Therefore, we first carry out comparisons between models with different configuration at the equal, large enough, but reasonable number of updates.

Depth and width. The first set of comparison investigates the effect of depth and width. The perplexity results can be found in Table 4.1. The training perplexities are computed during the final sub-epoch of the corresponding training. As we do not make use of dropout, these numbers are comparable to the development perplexities, up to parameter updates during the last sub-epoch. All models in the table use 8 attention heads. Other parameters are specified in the table. The table is organized in three parts:

The upper part of Table 4.1 shows the effect of number of layers; we observe that increasing number of layers (therefore the number of parameters) from 1 to 42 gradually improves the

[3]Training configuration files and trained models for this section are available at `https://github.com/rwth-i6/returnn-experiments/tree/master/2019-lm-transformers`.

perplexity. This result alone only tells that the model seems to require more parameters for this dataset, and that training of the deep models seems to work without particular effort.

In the middle part of Table 4.1 , we vary both the number of layers, feed-forward dimension, and the residual dimension. First of all, the 12-layer ($L = 12, d_{\text{ff}} = 4096, d_{\text{res}} = 512, H = 8$) model outperforms the 6-layer ($L = 6, d_{\text{ff}} = 8192, d_{\text{res}} = 512, H = 8$) model, while having similar number of parameters, which seems to indicate that the depth effectively benefits Transformer language models. We also train an extreme model which has only 2 layers with wide dimensions ($L = 2, d_{\text{ff}} = 8192, d_{\text{res}} = 2048, H = 8$). The number of parameters in fact blows up because of the large value of d_{res} which results in a large matrix in the output softmax layer with 200 K vocabulary[4]. We observe that such wide but shallow models do not perform well. Since the softmax bottleneck dimension typically needs to be large for the best performance [Yang & Dai$^+$ 18], we also train a ($L = 12, d_{\text{ff}} = 2048, d_{\text{res}} = 512, H = 8$) model where we insert an additional projection layer with a large dimension of 2048 before the output layer; no improvement was obtained though.

Finally, the lower part of Table 4.1 shows deeper models with a smaller input dimension, which gives further improvements.

Table 4.1: *Perplexity on word level LibriSpeech* ***after 2.5 epoch*** *(25 sub-epochs in our setup; 6.5 M updates). The number of heads* H is 8 for all models below.

Dim. Input embedding	L	d_{ff}	d_{res}	Params. in M	Perplexity	
					Train	Dev
512	1	2048	512	208	108.3	104.9
	6			224	75.7	74.3
	12			243	67.6	67.1
	24			281	62.2	62.3
	32			306	60.1	60.6
	42			338	**59.0**	**59.6**
512	2	8192	2048	536	73.1	73.8
	6		512	262	66.7	66.7
	12	4096		268	**63.5**	**63.8**
	4	16384		277	67.6	67.4
		32768		344	65.4	68.4
128	64	2048	512	330	56.3	57.6
	80			380	53.1	55.5
	96			431	51.9	54.9
	112			481	**51.5**	**54.5**

Number of heads. Table 4.2 shows the effect of number of attention heads. 16 heads which is the largest number we try in this setup give the best performance.

Activation function. In addition, we examine the type of activation function (Table 4.3). As opposed to previous work on feed-forward language models using GLUs [Dauphin & Fan$^+$ 17, Irie & Lei$^+$ 18b], we did not observe faster convergence. As we observed that the impact of choice of activation functions on the perplexity is overall limited, all our other models use the standard ReLU.

[4] We note that this is also the reason why the number of parameters of our baseline LSTM language models in Table 3.3 in the preliminary is relatively high.

Table 4.2: *Effect of number of heads. Perplexity on word level LibriSpeech* ***after 2.5 epoch*** *for* $(L = 12, d_{ff} = 2048, d_{res} = 512, H)$.

H	Params. in M	Perplexity	
		Train	Dev
1	243	71.9	70.8
4		69.1	68.6
8		67.6	67.1
16		**66.9**	**66.6**

Table 4.3: *Effect of activation functions. Perplexity on word level LibriSpeech* ***after 1 epoch*** *(10 sub-epochs in our setup) for* $(L = 24, d_{ff} = 2048, d_{res} = 512, H = 8)$.

Activation	Perplexity	
	Train	Dev
ReLU [Nair & Hinton 10, Vaswani & Shazeer$^+$ 17]	76.4	72.5
GLU [Dauphin & Fan$^+$ 17]	76.5	72.8
GELU [Hendrycks & Gimpel 18, Radford & Wu$^+$ 19]	**75.7**	**72.2**

Final models. Finally, we train models with the best configurations until convergence. Table 4.4 shows the perplexities of the fully converged models. These perplexities obtained by the Transformer models are better than those obtained by our LSTM based models. It is important to note here that we did not apply any regularization on models as almost no overfitting was observed in the range of model sizes we experimented with. We emphasize again the possibilities to still improve our models simply by scaling up their size and using regularization, as we discussed in the preliminary Sec. 3.1.2.

Table 4.4: *Final perplexities on LibriSpeech* ***after full convergence****. The baseline 4-gram and LSTM numbers are taken from Table 3.3.* d_{res} *is 512 for all Transformer models.*

Model	Input Emb.	L	d_{ff}	Params. in M	Perplexity	
					Dev	Test
4-gram	-			230	146.2	151.8
LSTM	-			1048	60.2	63.2
Transformer	512	12	4096	268	59.9	62.3
		24	2048	281	58.0	60.7
		32		306	56.6	59.5
		42		338	**55.0**	**57.7**
	128	80		380	53.5	56.3
		96		431	53.2	55.9
		112		481	**52.5**	**55.2**

4.1.3 Residual vs. Highway Connection

As has been already aforementioned in Sec. 3.1.6, we also note that the highway connection is an alternative method to residual connections to train deep neural networks. We carried out a brief ablation study in which we replace residual connections in the Transformer by highway connections. We conducted experiments for the word-level 24-layer model.

First of all, as noted in [Srivastava & Greff$^+$ 15a], we found the bias initialization in the gate function to be crucial for the highway based model. Table 4.5 summarizes the corresponding effect.

Finally, as shown in Table 4.6, we found the residual connection to work better than the highway connection in our Transformer language modeling setup.

Table 4.5: *Effect of gate bias initialization. Perplexity on the LibriSpeech Dev set after 1 sub-epoch for* $(L = 24, d_{ff} = 2048, d_{ff} = 512, H = 8)$ *with highway connections.*

Bias Init.	Perplexity
0	973.1
1	194.4
5	**134.8**
10	149.7
50	312.6

Table 4.6: *Residual connection vs. Highway connection in Transformer models* $(L = 24, d_{ff} = 2048, d_{res} = 512, H = 8)$*. Perplexity after convergence.*

Skip Connection Type	Params. in M.	Perplexity		
		Train	Dev	Eval
Residual	281	55.6	**58.0**	**60.7**
Highway	306	68.7	68.1	71.4

Layer norm and residual connections. We tried to train multiple models without either residual connections or layer normalization, but without success. As reported in the previous works on Transformers, we thus confirm that both layer normalization and residual connections are needed for these models for stable training. Also, following [Radford & Wu$^+$ 19], we tried reorganizing the *feed-forward module* to insert one additional pre-activation layer normalization [He & Zhang$^+$ 16b] and one more activation function. However, we did not observe any improvement. The original Transformers anyway do not have any activation on the residual path throughout the whole network.

However, in a future work, it would be interesting to investigate better initialization techniques such as [Zhang & Dauphin$^+$ 19, Dauphin & Schoenholz 19], also for these deep language models, which would allow us to get rid of layer normalization operations, therefore to reduce these extra computation for a training helper.

4.1.4 Parameter Tying

Dehghani et al. [Dehghani & Gouws+ 19] has reported Universal Transformers to perform particularly well for language modeling. This motivates us to experiment with parameter sharing across layers. For such models to have comparable number of parameters with the standard deep Transformers, the dimensions in each layer must be increased, which results in slower training; here we simply investigate the effect of number of recurrence.

Table 4.7 shows the perplexity results. First of all, we observe that the model performance is behind that of the standard Transformers (Table 4.1). However, we note that here the direct comparison is not as straightforward as between the standard Transformers. In fact, we observe that the training hyper-parameters tuned for the standard Transformers can not be directly applied to Universal Transformers. Specifically, we find it crucial to reduce the gradient norm clipping threshold from 1 to 0.1. This smaller clipping threshold is potentially slowing down the convergence.

Second, we can clearly observe that increasing the number of layers from 3 to 12 consistently improves the perplexity. This improvement without additional parameters motivates future work to investigate further parameter sharing strategies for Transformers. Such a model can be interesting, especially in an overfitting scenario, as a method for increasing the modeling power without increasing the number of parameter of the model.

Table 4.7: *Perplexity on LibriSpeech* ***after 2.5 epoch*** *for* $(L, d_{ff} = 8192, d_{res} = 1024, H = 16)$ *models with shared parameters across all layers.*

L	Params. in M	Perplexity	
		Train	Dev
3	329	82.6	79.9
6		76.7	74.6
12		**74.2**	**72.1**

4.1.5 ASR Experiments

Lattice rescoring results. We apply our word-level Transformer language models to the baseline NN-HMM hybrid speech recognition system (A.1) by lattice rescoring (Sec. 1.2.2). The standard push-forward lattice rescoring algorithm [Sundermeyer & Tüske+ 14] for long-span language models can be directly applied to self-attention based models. The only modifications from the RNN version is to define the "state" as all hidden states $h_n^{(l)}$ in Eq. (4.3) in all layers from all predecessor positions and the current position index (n; for position encoding). Table 4.8 shows the WERs and perplexities (PPL). We obtain consistent improvements in terms of WER over the LSTM baselines.

Table 4.8: *WERs (%) for **hybrid NN-HMM** systems on LibriSpeech. The 4-gram model is used in the first pass to generate lattices for rescoring. The row "Lattice" shows oracle WERs of the lattices.*

LM	L	Param. in M	dev				test			
			clean		other		clean		other	
			PPL	WER	PPL	WER	PPL	WER	PPL	WER
4-gram	-	230	151.7	3.4	140.6	8.3	158.1	3.8	145.7	8.8
Lattice	-	-	-	1.0	-	2.3	-	1.3	-	2.6
LSTM	2	1048	60.2	2.3	60.2	5.4	64.8	2.6	61.7	5.9
Transformer	24	281	57.8	2.2	58.3	**5.2**	62.2	**2.5**	59.4	5.7
	42	338	54.5	**2.1**	55.5	5.2	59.1	**2.5**	56.4	5.7
	96	431	52.7	**2.1**	53.7	**5.1**	57.3	**2.5**	54.5	5.7
	112	481	**52.0**	**2.1**	**53.0**	5.2	**56.4**	**2.5**	**53.9**	**5.6**

End-to-End ASR shallow fusion results. We also trained 10 K BPE-level Transformer language models to be combined with an attention-based encoder-decoder speech model (A.1) by shallow fusion [Gülçehre & Firat+ 17, Toshniwal & Kannan+ 18] (Sec. 1.2.3). The 10 K BPE level training data has a longer average length of 24 tokens per sentence with the longest sentence length of 1343, which is still manageable without any truncation for self-attention. We use the Transformer architecture of (24, 4096, 1024, 8). The LSTM model has 4 layers with 2048 nodes. Table 4.9 shows both perplexities and WERs. Following [Hannun & Lee+ 19], we introduce an end-of-sentence penalty in shallow fusion to benefit from a large beam size of 64. Again, we obtain consistent improvements over the LSTM baseline. These results are better than previously reported WERs [Hannun & Lee+ 19, Zeghidour & Xu+ 18, Irie & Prabhavalkar+ 19a] for end-to-end models without data augmentation [Park & Chan+ 19].

We also note that this LSTM configuration has been used in a number of follow-up works from other teams reporting good WERs on LibriSpeech [Karita & Chen+ 19, Han & Prieto+ 19].

Table 4.9: *WERs (%) for **attention-based models** on LibriSpeech. Perplexities are on the 10 K BPE level.*

LM	Beam	dev				test			
		clean		other		clean		other	
		PPL	WER	PPL	WER	PPL	WER	PPL	WER
None	12	-	4.3	-	12.9	-	4.4	-	13.5
LSTM	64	43.7	2.9	46.4	8.9	47.1	3.2	47.2	9.9
Transformer		**35.9**	**2.6**	**38.9**	**8.4**	**38.8**	**2.8**	**39.0**	**9.3**

4.1.6 Conclusion

In this section 4.1, we investigated Transformer models for language modeling in speech recognition. We carried out model tuning specifically for the task of language modeling; on the LibriSpeech dataset, the best models ended up being very deep, having about 100 layers. The final model gave about 12% relative improvements in perplexity and 4-10% relative improvements in WER over the well tuned LSTM baseline. We conducted experiments for both word and BPE level language modeling and applied them to hybrid NN-HMM and end-to-end ASR systems.

The evaluation of Transformer language models is extended in Sec. 4.4, in which this good performance is confirmed across more datasets (Swichboard, AMI, Quaero, TED-LIUM 2).

4.2 Analysis for Better Understanding Transformer Language Models

In this section, we analyse the Transformer language model. The first focus of analysis is the positional encoding, which is a crucial component in the original Transformer. In fact, the design of the positional encoding itself has been an active research topic [Gehring & Auli$^+$ 17, Shaw & Uszkoreit$^+$ 18, Sperber & Niehues$^+$ 18, Salazar & Kirchhoff$^+$ 19, Dai & Yang$^+$ 19, Wang & Zhao$^+$ 20]. For example, relative positional encoding has been considered for general purposes in [Shaw & Uszkoreit$^+$ 18]. An improved variant of relative positional encoding has been proposed in [Dai & Yang$^+$ 19] in the context of Transformer-XL models, for which the use of relative positional encoding is natural by construction, as the model processes a sequence, segment by segment, with a fixed segment length, and by only accessing the current and the predecessor segments (further discussion can be found in Chapter 7, Sec. 7.1). In [Wang & Zhao$^+$ 20], instead of considering the combination of two independent word embedding and positional encoding vectors by a simple element-wise addition, the word embedding is directly defined as a function of the position. Slight improvements by such a modification have been reported in character-level language modeling.

Previous works in standard language modeling with Transformers systematically use positional encoding, typically either a jointly learned one or the sinusoidal one (both cases are reported to give similar performance in [Al-Rfou & Choe$^+$ 19]). We show that the deep autoregressive self-attention models do not require any explicit input embeddings for positions to give the best performance.

Second, attention weights are easier to be visualized than the hidden states in RNNs, which gives opportunity for analysis by visualization. In particular, we focus on the comparison of the models with and without positional encoding, in the first layer. Finally, we investigate the behavior of each layer in the deep Transformer language models.

4.2.1 Transformer Language Models Without Positional Encoding

In the autoregressive problem where a new token is provided to the model at each time step, the amount of information the model has access to strictly increases from left to right at the lowest level of the network. The deeper layers should be able to recognize this structure which should provide the model with some positional information by its own. To validate this hypothesis, we train models without any positional encoding. The perplexity comparisons are shown in Table 4.10. We observe that they give better perplexities than the models with sinusoidal positional encoding. Lower training perplexities for the equal number of parameters indicate that the model without positional encoding simply learns better. In order to confirm the behavior of these models, we visualize the attention weights in the next section.

Table 4.10: *Effect of sinusoidal positional encoding. Perplexity* ***after 5 epochs*** *(13 M updates; full convergence) for* $(L, d_{ff} = 2048, d_{res} = 512, H = 8)$ *models.*

L	Position. encoding	Params. in M.	Perplexity		
			Train	Dev	Test
12	Sinusoidal	243	61.8	63.1	66.1
	None		58.0	**60.5**	**63.4**
24	Sinusoidal	281	55.6	58.0	60.8
	None		52.7	**56.6**	**59.2**
42	Sinusoidal	338	51.2	55.0	57.7
	None		50.5	**54.2**	**56.8**

4.2.2 Identifying 4 Functional Groups of Layers

First layer. The attention in the first layer is the most straightforward for interpretation because the feature at each position exactly corresponds to the word at the position (while deeper layers can potentially shuffle the feature content). The attention weights in the first layer of 24-layer Transformer language models with and without positional encodings are visualized in Figures 4.2 to 4.6. We observed that the first layer of the model with positional encoding (Figure 4.2) learns to create *n-gram features* (roughly 2 or 3-gram), which indicates that the positional information is directly used. In contrast, the first layer of the model without positional encoding learns to focus on the new input token as can be seen as the diagonal in Figure 4.3, which demonstrates that the model is aware of the position of the new input. Interestingly, we also see that it ignores some functional words such as "the", "and", "to" which might be modeled by some off-set values, therefore attending to the beginning of sentence token instead.

Later in [Zeyer & Bahar$^+$ 19], we also successfully trained Transformer based encoder decoder attention speech recognition models without positional encoding in the *decoder* component.

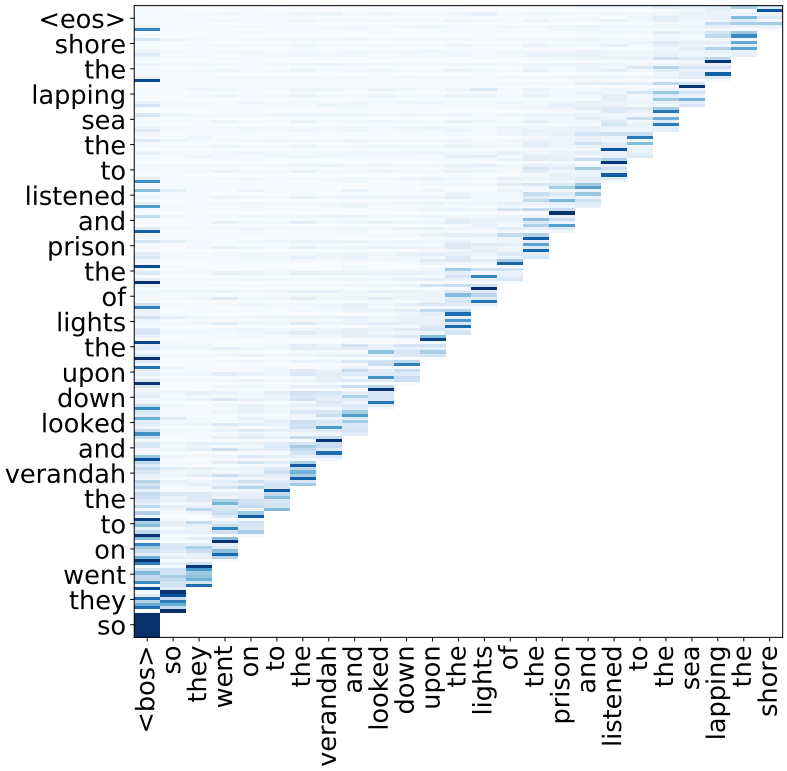

Figure 4.2: *Attention weights in the **first layer** for the model **with** positional encoding. The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

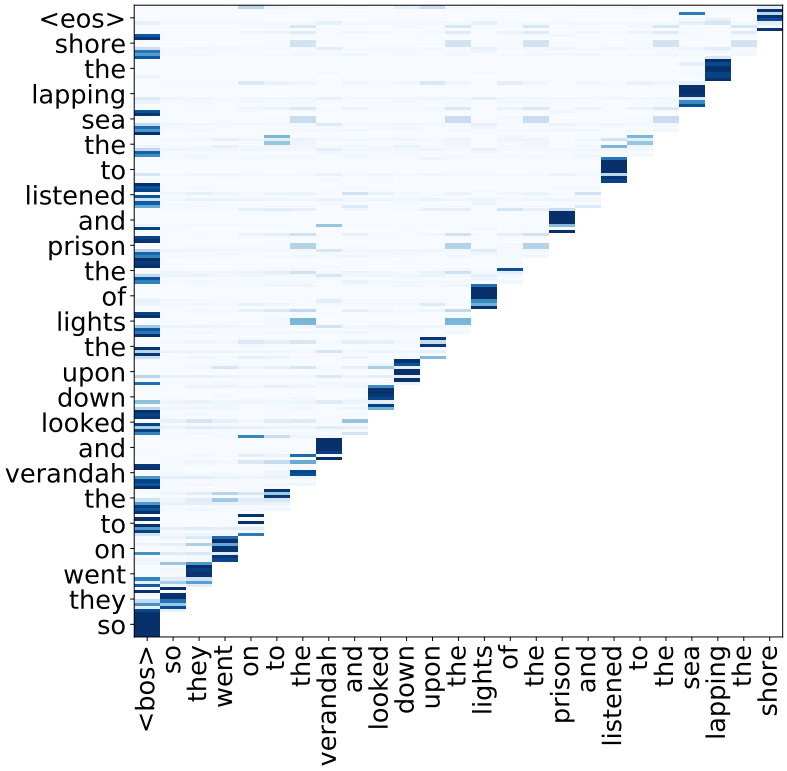

Figure 4.3: *Attention weights in the **first layer** for the model **without** positional encoding. The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

Other layers. The natural next step after the visualization of attention in the first layer is to investigate what other layers are doing. We observe that the behaviors of other layers are rather similar for both Transformer models with and without positional encoding. We find 3 categories of layers in the other 23 layers. The second and third layers are *"blur"* layers as shown in Figure 4.4, which seems to roughly average over all positions, similar to the bag-of-words, while we can also see that some heads focus on *difficult* words, here "verandah". Layers 4 to 9 are *window* layers which focus on the local n-gram context. A representative example is show in Figure 4.5. Finally, we find the top layers 10 to 24 to be more *structured*, attending to some specific patterns. An example is shown in Figure 4.6. We can relate these layers to the old max-entropy language models [Rosenfeld 96] where these features (such as word triggers) were manually designed. This trend will be later confirmed in chapter 7, when we consider the same model for translation language modeling (Sec. 7.2)

We found that much deeper models (such as our 96-layer model) also present this same 4-group layer structure (input, blurring, windowing, and structured layers). We observed that the deeper models contain roughly the same number of blur and window layers as the 24-layer model, but they have much more structured top layers.

We note that in the context of masked language modeling with BERT [Devlin & Chang$^+$ 19], [van Aken & Winter$^+$ 19] has also conducted layer-wise analysis in a similar spirit.

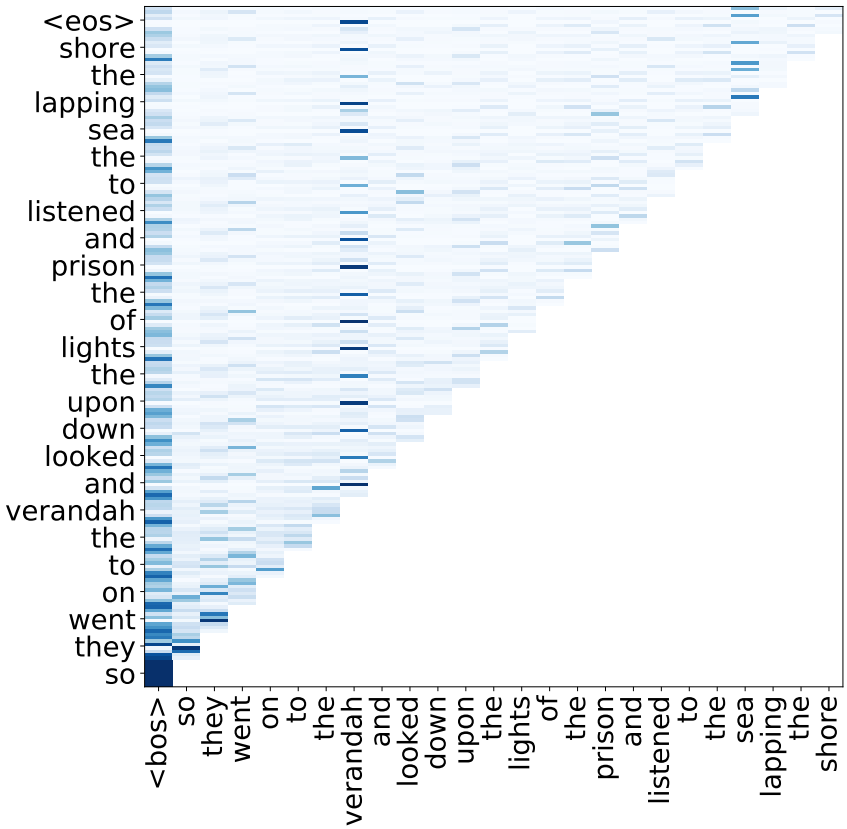

Figure 4.4: *Attention weights in the second layer representing the "blur"* ***bottom layers (2-3)*** *for the model without positional encoding.* ***These layers seem to carry out averaging over all positions, thus collecting global information. Some heads focus on difficult words, here "verandah".*** *The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

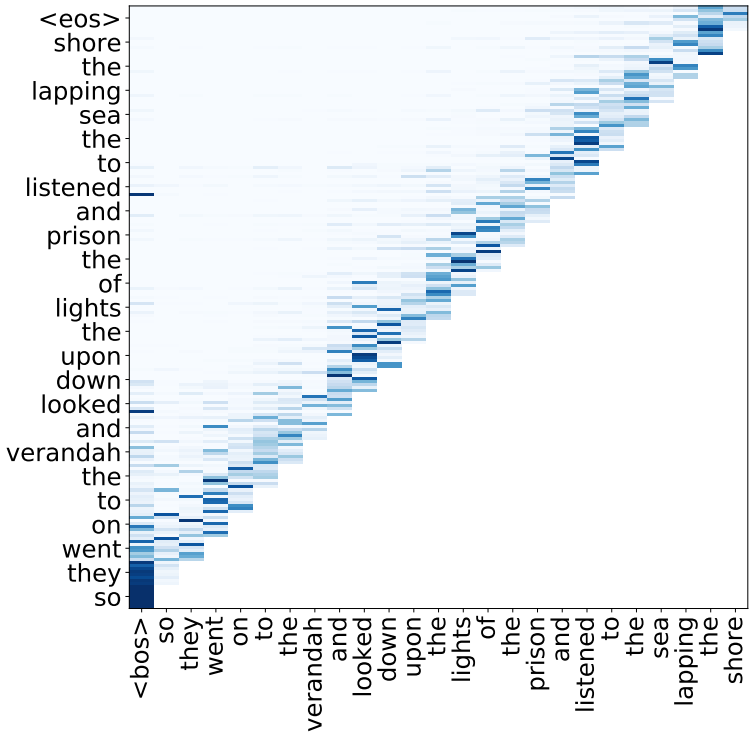

Figure 4.5: *Attention weights in the 5th layer representing the "window"* ***mid layers (4-9)*** *for the model without positional encoding.* ***These layers focus on the local n-gram.*** *The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

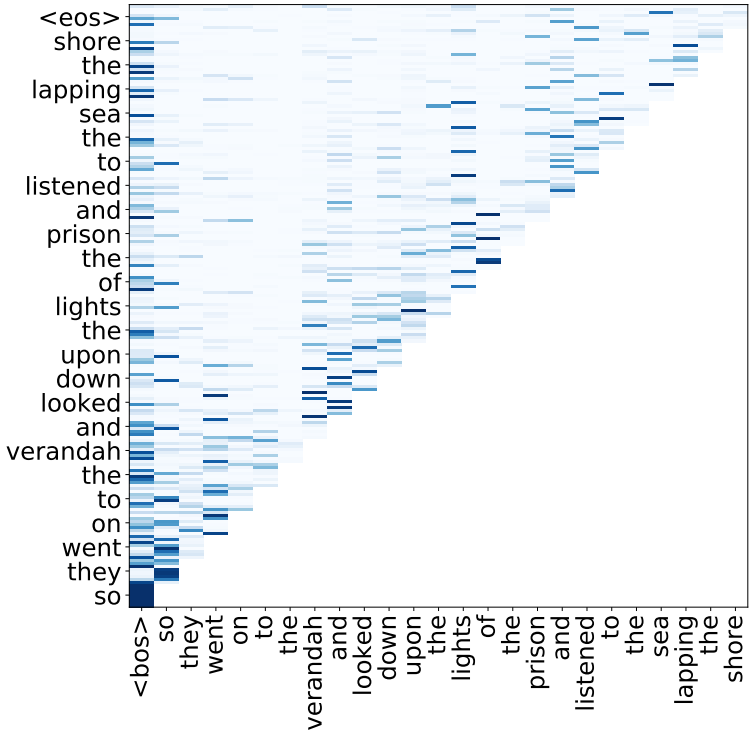

Figure 4.6: *Attention weights in 24th layer representing the "structured"* ***top layers (10-24)*** *for the model without positional encoding.* ***It seems to be some feature detector attending to some specific patterns.*** *The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

4.3 Alternative Architecture for More Memory Efficient Search

In the previous section 4.1, we have shown that deep Transformer language models can be successfully applied for automatic speech recognition. However, memory requirements of such large and deep Transformer language models at evaluation time become very demanding because each self-attention sub-layer in the model stores key and value vectors for all predecessor positions. This is a practical issue, since search algorithms (including lattice rescoring in hybrid NN-HMM based ASR and shallow fusion in end-to-end speech recognition) typically store these large *states* for a large number of hypotheses.

Interestingly, the only hyper-parameter in the original Transformer which can increase the number of model parameters (therefore potentially the model capacity) without affecting the state size is the feed-forward inner dimension. A larger key or value dimension obviously increases the state size. More layers increases the number of self-attention sub-layers (therefore the state size) as each layer contains one self-attention and one feed-forward sub-layer.

A natural question which arises out of this observation is whether we can put more parameters in the feed-forward module more efficiently. In this section, we investigate the following modifications with the goal of achieving a smaller state but still powerful Transformer: First, we introduce an extra hyper-parameter to specify the number of feed-forward sub-layers in each Transformer layer. This means that we replace the feed-forward module by a deep neural network (DNN) with residual connections, which could allow us to increase the model capacity efficiently, independent of the state size. Second, we also explore sharing key and value projection matrices which would allow Transformers to only store key vectors as its states.

A number of previous works [Lan & Chen$^+$ 19, Kitaev & Kaiser$^+$ 20] have focused on reducing the *model size* of the Transformer. We note that our goal is orthogonal. We are primarily interested in reducing the *state size* of Transformers. Notable previous works have proposed limited state size Transformers which makes use of some segment level recurrence such as Transformer-XL [Dai & Yang$^+$ 19] or Compressive Transformer [Rae & Potapenko$^+$ 20]. Our method can be applied in combination with these techniques. Some quantization (e.g. [Kumar & Nirschl$^+$ 17] for model compression) can be an alternative solution for reducing the state size. In this section, we tackle this problem from the modeling perspective.

Sharing query and key matrices has been investigated in [Kitaev & Kaiser$^+$ 20]. However, that does not help in reducing the state size. [Lample & Sablayrolles$^+$ 19] replaces some feed-forward layers by a more powerful but efficient product-key memory layer, and they also effectively managed to reduce the number of self-attention layers; in principle, our work also follows a similar spirit since we also replace the feed-forward sub-layer by a more powerful DNN.

4.3.1 Transformer Language Model with Reduced State Size

States size analysis in standard Transformer language model. As has been previously defined with Eq. (4.3) in Sec. 4.1, each self-attention module in L-layer Transformer language model at position n stores the **state vector** $h_n^{(l)}$:

$$h_n^{(l)} \quad = \quad \left(h_{n-1}^{(l)}, (k_n^{(l)}, v_n^{(l)})\right)$$

As in Sec. 4.1, we use the same dimension for key, value, and query dimensions, as well as for the residual connection, which we denote as d_{kv}. Its total state size is thus $2 \times L \times n \times d_{\text{kv}}$ which not only grows with the position n, but when we make the model deeper (larger L) or wider via self-attention dimensions (larger d_{kv}).

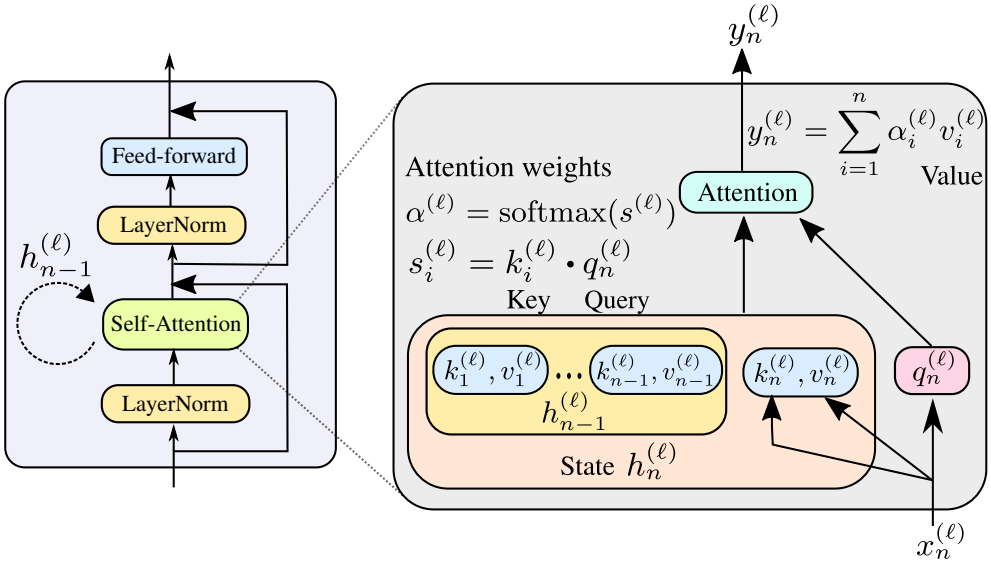

Figure 4.7: Illustration for the standard Transformer layer.

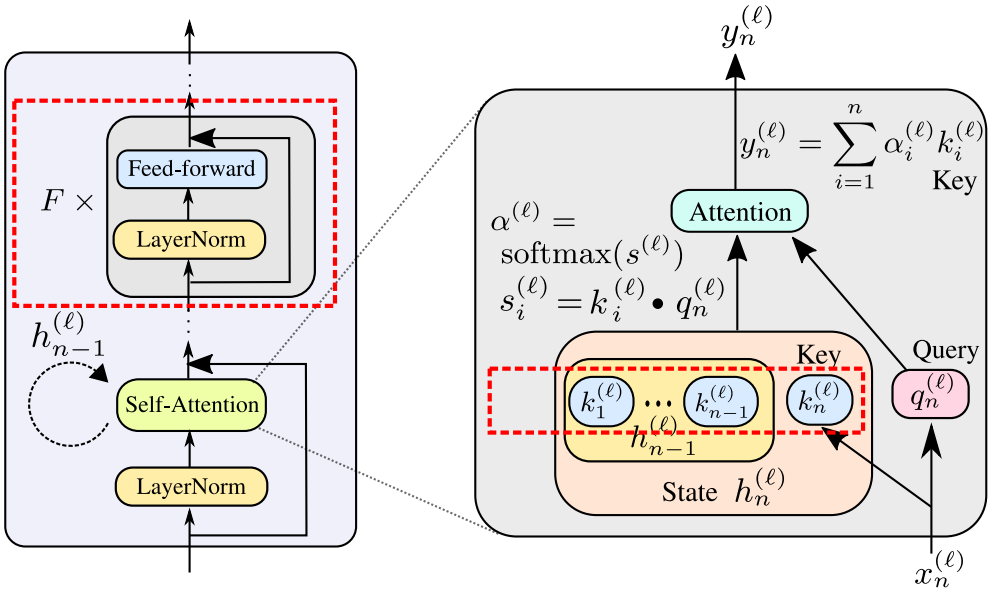

Figure 4.8: Illustration for the modified Transformer layer.

Modified Transformer for smaller state size. For deep Transformer language models, the size of state vectors $(h_n^{(1)}, .., h_n^{(l)}, .., h_n^{(L)})$ can be potentially very large, which is inconvenient, especially for ASR applications, for search where such states must be stored for a large number of hypotheses.

At the same time, we need to provide the model with a large number of parameters for a good performance. The only model hyper-parameter in the original Transformer which can increase the number of model parameters, but does not affect the state size is the feed-forward inner dimension d_{ff}. In order to **isolate increase in the model size from the total state size** in a Transformer, we make the feed-forward component in each Transformer layer deeper: using $F^{(\ell)}$ layers for the layer ℓ as indicated in Figure 4.8 (to be contrasted with the standard Transformer layer illustrated in Figure 4.7). In experiments, we use the same number F for all layers.

We therefore propose a modified Transformer layer design which:

- Defines the Transformer layer as one self-attention sub-layer plus F feed-forward sub-layers

(*self-attention-DNN*). Each sub-layer uses the layer normalization and the residual connection as in Eqs. (4.9, 4.10).

- Shares key and value weight matrices $K^{(l)}$ and $V^{(l)}$ (*shared-KV*), and only store the key vectors as the state:

$$q_n^{(l)}, k_n^{(l)} = Q^{(l)} x_n^{(l)}, K^{(l)} x_n^{(l)} \tag{4.13}$$

$$h_n^{(l)} = \left(h_{n-1}^{(l)}, k_n^{(l)}\right) \tag{4.14}$$

With this model, we aim to increase the number of feed-forward sub-layers F while reducing the number of Transformer layers L, as long as it preserves the model capacity. Sharing K and V is an extra option for further reduction of state size. While we evaluate this model for language modeling, this can be applied to any Transformer models.

4.3.2 Experimental Setups

We present our main results on the TED-LIUM release 2 (200h) dataset [Rousseau & Deléglise$^+$ 14] (Sec. A.2). TED-LIUM is a medium-size publicly available dataset (270M running words) and sentences are relatively long (20 words on average for the development and evaluation sets). The word level vocabulary size is 152 K. Our experiments have been conducted using the TensorFlow [Abadi & Barham$^+$ 16] based open-source toolkit RETURNN [Zeyer & Alkhouli$^+$ 18][5].

Baseline LSTM and Transformer language models. In order to exploit the mutli-corpus TED-LIUM training data (Appendix 4.3.2), we train both LSTM and Transformer language models in two steps. We first pre-train the model on the whole training data until convergence. Then we fine-tune the model on the TED-LIUM 2 *transcriptions* (2 M words) and *common crawl* (16 M words) sub-sets which are the top-2 sets with the highest weights for 6-gram interpolation[6]. This corresponds exactly to the domain adaptation technique we introduced in the preliminary Chapter 3, Sec. 3.1.4.

The perplexities of the LSTM and standard Transformer models are presented in the lower part of Table 4.11. The input word embedding dimension is 128 for all models. The LSTM model has 4 layers with 2048 nodes, and we apply 20% dropout (which gave a tiny improvement). For Transformers, the number of attention heads H is always set to 12 and d_{kv} is set to 768 unless specified otherwise. With the number of layers L and the feed-forward inner dimension d_{ff}, all our standard Transformer models are fully specified: The model in Table 4.11 has 32 layers with $d_{\text{ff}} = 4096$. No positional encoding is used following the finding from the previous section ([Irie & Zeyer$^+$ 19a]). More than 15% relative improvement in perplexity is obtained by the Transformer over the LSTM baseline.

The results reported in this section has been published in [Irie & Gerstenberger$^+$ 20]. As an additional engineering improvement from the predecessor work [Irie & Zeyer$^+$ 19a], we also made use of two speed-up methods for training neural language models: the noise contrastive estimation [Gutmann & Hyvärinen 10] loss and a training speedup oriented batch construction. To successfully train a model with a good performance using the noise contrastive estimation loss, we found it crucial to initialize the bias of the softmax layer to $-\log(V)$ as recommended by [Devlin & Zbib$^+$ 14][7].

[5] The config files and models are available in `https://github.com/rwth-i6/returnn-experiments/tree/master/2020-lm-small-state-trafo`.

[6] We set this up before getting aware of the overlap problem in the original TED-LIUM dataset (see Appendix A.2). The interpolation weights for 6-gram models were: 22% for the *common crawl*, 16% for the TED-LIUM 2 *transcriptions* and 60% for the background model, while the weight next in size was only 0.9%.

[7] We thank Alexander Gerstenberger who made us aware of this trick.

Table 4.11: ***Perplexity*** *of the word-level (152K vocab)* ***baseline*** *models on* ***TED-LIUM 2****.*

Model	Params. in M	Perplexity	
		Dev	Test
4-gram	343	105.4	124.7
+ pruning	161	113.2	127.9
LSTM	450	73.5	71.3
Transformer	414	**62.0**	**60.7**

Also on the side of batch construction method, instead of fully randomizing sentences, we first sort them by the length, create bins (each bin containing as many sentences as the batch size; here 32), and shuffle the bins. When indicated, we make use of these two speed-up techniques. In our preliminary experiments, we found that both techniques can give a large speedup (up to a factor of four by accumulating both techniques) in training almost without loss of performance.

4.3.3 Effect of DNN Inside Transformer Layer

We introduce an extra hyper-parameter F to specify the number of feed-forward layers in each Transformer layer. Table 4.12 shows the perplexity results for TED-LIUM 2. The baseline numbers in the first block are copied from the baseline 32-layer standard Transformer model presented in Table 4.11, to highlight the hyper-parameters in the same format. The *8L-3F* model which contains only 8 layers with 3 feed-forward sub-layers per layer (therefore only 8 self-attention and 24 feed-forward sub-layers) achieves comparable performance with the 32-layer models with a smaller state size. The more extreme $3L$-$15F$ model performs worse than the standard 8-layer Transformer model while having a higher number of parameters.

Table 4.12: *Perplexity of the word-level (152K vocab) models on* ***TED-LIUM 2****.* $d_{kv} = 768$ *and* $H = 12$ *for all models. The models with* $F = 1$ *are standard Transformers.*

L	F	d_{ff}	State size per position	Params. in M	Perplexity	
					Dev	Test
8	1	4096	12,288	206	67.9	64.9
32		2048	49,152	313	63.3	61.5
		4096		414	**62.0**	**60.7**
3	15	2048	4,608	247	69.3	66.0
6	7		9,216	280	64.5	62.6
8	3	4096	12,288	338	63.4	61.7
12			18,432	379	62.2	61.0
16			24,576	464	**61.4**	**60.7**

We also carry out similar experiments on LibriSpeech. Table 4.13 presents the perplexity comparison. The *6L-7F model* (6 self-attention and 42 feed-forward sub-layers) is trained using speed-up tricks (Sec. 4.3.2). The model achieves a similar number of parameters as the 32-layer standard Transformer model (taken from the previous Sec. 4.1.2, Table 4.4), while only containing 6 self-attention layers. The proposed model gives similar perplexities with much smaller state size. We note that this 32-layer model (the same model used in the previous section) makes use of positional encoding (which is in favor for the new $6L$-$7F$ model); but $6L$-$7F$ is trained using the speed-up tricks (Sec. 4.3.2) which instead is in favor of 32-layer model.

Table 4.13: ***Perplexity*** *of the word-level (200K vocab) model on* ***LibriSpeech****. d_{kv} is* ***512*** *for all models. The numbers for the standard models are taken from Table 4.4.*

L	F	d_{ff}	H	NCE Train	State size per position	Params. in M	Perplexity Dev	Perplexity Test
32	1	2048	8	No	32,768	306	56.6	59.5
42					43,008	338	**54.2**	**56.8**
6	7	4096	16	No	6,144	307	55.5	58.1
				Yes			56.8	59.4

4.3.4 Effect of Tying Key and Value Matrices

Sharing K and V matrices is appealing in the context of building small state Transformers because it allows us to only store key (or value) vectors as states in Transformers which can directly reduce the state size by a factor of two. We therefore evaluate KV-sharing in Transformers with self-attention-DNN (Sec. 4.3.3) as an extra method for reducing the state size. Table 4.14 shows that this approach results in a degradation of up to 5% relative in perplexity. The $6L$-$7F$ model without shared-KV from Table 4.12 outperforms the $8L$-$3F$ model with shared-KV while having less parameters.

Interestingly, KV-sharing gives almost no degradation on the standard 32-layer model, while it has more (32) self-attention layers which are affected by this parameter sharing. It seems that when the model has only a few self-attention layers, they need to have the complete set of parameters to perform well.

Table 4.14: *Effect of sharing KV for both standard and small state Transformers.* ***Perplexity*** *on* ***TED-LIUM 2*** *(152K vocab).*

Shared-KV	L	F	State size per position	Params. in M	Perplexity Dev	Perplexity Test
No	32	1	49,152	414	**62.0**	**60.7**
Yes			24,576	395	62.7	61.2
No	8	3	12,288	338	**63.4**	**61.7**
Yes			6,144	333	66.3	63.9

4.3.5 ASR Experiments

We finally show lattice rescoring experiments on TED-LIUM 2 with the baseline NN-HMM system (A.2). Table 4.15 shows the WERs. The proposed small state $8L$-$3F$ Transformer model give comparable performance to the standard deep Transformers, with 4 times smaller memory requirement (concretely, the highest requirement for one lattice is reduced from 65 GB to 17 GB).

Table 4.15: *WERs on **TED-LIUM 2**. Perplexities are after **interpolation** with the 4-gram LM. Lattices are generated by either 4-gram or 4-gram + LSTM LMs in the **first pass**.*

Model	L	F	Dev		Eval	
			PPL	WER	PPL	WER
4-gram	-	-	113.2	6.8	127.9	7.3
+ LSTM	-	-	64.4	5.5	69.2	6.0
+ Transformer	32	1	55.3	**5.3**	60.1	5.9
	8	3	56.6	**5.3**	61.1	5.9
	16		**54.9**	**5.3**	**59.8**	**5.8**
4-gram + LSTM	-	-	64.4	5.5	69.2	6.1
+ Transformer	32	1	54.8	**5.1**	59.3	**5.6**
	8	3	56.0	5.2	60.1	5.7
	16		**54.4**	5.3	**58.9**	5.7

4.3.6 Conclusion

In this section, we demonstrated that the one-to-one ratio between the numbers of self-attention and feed-forward sub-layers in the standard Transformer is sub-optimal when we consider both the state size and performance of the model. We investigated a possibility to reduce the total number of self-attention sub-layers in the model, by increasing the number of feed-forward sub-layer in each Transformer layer instead. This allowed us to reduce the number of self-attention layers to a relatively small number such as 6 or 8, with only a marginal loss of performance. These small state Transformers directly reduced the memory requirement for the down-stream ASR application. Sharing key and value matrices in addition to this modification, only allowed us to halve the state size at the cost of loss of performance.

4.4 Comparing LSTM and Transformers Across Different Datasets

The previous sections in this chapter have shown large improvements by deep Transformer language models over the LSTM language models on two large publicly available datasets for language modeling in ASR: LibriSpeech and TED-LIUM 2.

Finally in this section, we extend the comparison of LSTM and Transformer language models on more datasets, and provide an overview in order to obtain a better picture of how this comparison extrapolates to different data conditions, and lead to some trend.

4.4.1 Performance Overview

Table 4.16 summarizes the comparison for 6 tasks. We observe that large improvements are obtained by Transformers over LSTM across all datasets, except on AMI and Switchboard datasets where improvements are marginal[8].

From these results, we obtain an empirical trend that improvements by Transformers over the well-tuned LSTM baseline models are more pronounced when the data is rather large and evaluation sentences are long. This is also in line with the recent trend showing the scalability of Transformer language models over LSTM based models [Radford & Wu$^+$ 19].

Table 4.16: *Perplexities and word error rates overview comparing LSTM and Transformer (Trafo) language models across different ASR datasets. A 4-gram Kneser-Ney language model is used to generate the lattices in all tasks except AMI for which 3-gram is used, and lattice rescoring is carried out using either the LSTM or Transformer language model, except for the LibriSpeech BPE level experiment which uses the attention based end-to-end system and shallow fusion. Except for the LibriSpeech experiments, the reported perplexities obtained by interpolating the rescoring neural language model with the n-gram language model. For LibriSpeech, Dev and Eval correspond to **dev-other** and **eval-other**. For Switchboard, numbers for **Switchboard** and **CallHome** parts of Hub5_00 set are presented as Dev and Eval for this table. "Train" indicates the number of tokens in the training data, and "Voc" indicates the vocabulary size.*

Dataset	Train [M]	Voc [K]	Avg. Sentence Length			Language Model	Dev		Eval	
			Train	Dev	Eval		PPL	WER	PPL	WER
Switchboard	27	30	10	11	8	LSTM	45.9	6.9	55.6	13.4
						Trafo	**43.0**	6.9	**53.5**	**13.3**
AMI	28	48	10	11	12	LSTM	56.1	17.2	58.0	15.6
						Trafo	**55.0**	**17.1**	**57.2**	**15.5**
Quaero	53	128	16	28	30	LSTM	81.4	9.0	82.6	7.7
						Trafo	**70.8**	**8.6**	**73.3**	**7.4**
TED-LIUM 2	270	152	18	36	25	LSTM	64.4	5.5	69.2	6.1
						Trafo	**55.3**	**5.3**	**60.1**	**5.9**
LibriSpeech	853	200	20	19	19	LSTM	60.2	5.4	61.7	5.9
						Trafo	**53.7**	**5.1**	**54.5**	**5.7**
BPE-level LibriSpeech	962	10	23	22	22	LSTM	46.4	8.9	47.2	9.9
						Trafo	**38.9**	**8.4**	**39.0**	**9.3**

[8] We still note that these WERs for the AMI dataset are the best numbers reported on this dataset, to the best of our knowledge. We thank Peter Vieting for having shared his unpublished baseline system [Vieting 19].

4.4.2 Combination of LSTM and Transformer Language Models

Finally, we conduct model combination of LSTM and Transformer language models. For that, we first generate lattices using a combination of 4-gram count and LSTM models in the first pass decoding[9] and we carry out lattice rescoring (Sec. 1.2.2) with the Transformer language model.

Table 4.17 presents the results for LibriSpeech, TED-LIUM 2, and Switchboard 300 h. As expected from the results in the previous section, large improvements of up to 10% relative in WER are obtained on LibriSpeech and TED-LIUM 2, while the improvements on Switchboard are rather limited. We note that the rescoring lattices generated by the LSTM language models (prefix trees as no recombination is done) seems to perform very well. These WERs obtained by the Transformer models are much better than in the case of rescoring 4-gram lattices in the previous section, while the perplexity improvements obtained by the interpolation with the LSTM model (together with the 4-gram model) are rather marginal. Therefore, the benefit of combination comes rather from the quality of the lattices (rather than the improvements in perplexity). We thus speculate that a combination with an extra LSTM language model would not bring much improvements, if in a future work, Transformer language models are used in the first pass decoding.

The WERs for TED-LIUM 2 and Switchboard 300 h presented here are the current best reported numbers in the literature, as of the time of writing[10]. The LibriSpeech numbers were state of the art at the time of writing [Lüscher & Beck$^+$ 19], by a large margin from the previous best results [Han & Chandrashekaran$^+$ 17], and they still remain competitive with more recent works which makes use of improved acoustic models based on Transformers [Han & Prieto$^+$ 19, Wang & Mohamed$^+$ 19, Synnaeve & Xu$^+$ 19].

Table 4.17: *Perplexities and word error rates for model combination between LSTM and Transformer language models across **standard ASR datasets**. For Switchboard, numbers for **Switchboard** and **CallHome** parts of Hub5_00 set are presented as Dev and Eval for this table. "Train" indicates the number of words in the training data, and "Voc" indicates the vocabulary size.*

Dataset		Train [M]	Voc [K]	Language Model	Dev		Eval	
					PPL	WER	PPL	WER
Switchboard		27	30	4-gram	68.8	8.1	80.5	15.4
				+ LSTM	45.2	6.7	55.0	13.5
				+ Transformer	**41.5**	**6.6**	**51.5**	**13.1**
TED-LIUM 2		270	152	4-gram	113.2	7.1	127.9	7.7
				+ LSTM	64.4	5.7	69.2	6.0
				+ Transformer	**54.8**	**5.2**	**59.2**	**5.7**
LibriSpeech	clean	853	200	4-gram	151.7	3.4	158.1	3.8
				+ LSTM	60.0	2.2	64.4	2.6
				+ Transformer	**51.2**	**1.9**	**55.5**	**2.3**
	other			4-gram	158.1	8.3	145.7	8.8
				+ LSTM	59.9	5.1	61.3	5.5
				+ Transformer	**52.2**	**4.5**	**53.0**	**5.0**

[9] We thank Eugen Beck and Wei Zhou for having run the corresponding experiments and provided us with the lattices. We note that the differences in baseline LSTM results are due to the first pass decoding (this section) and lattice rescoring (previous section).

[10] The previous best numbers were reported in [Han & Chandrashekaran$^+$ 17] for TED-LIUM 2, and for Switchboard 300 h, we carried out rescoring on top of the state-of-the-art system reported in [Kitza & Golik$^+$ 19].

4.5 Summary

In this chapter, we successfully applied the recently proposed Transformer model to language modeling in speech recognition. While its effectiveness had been already demonstrated on machine translation and other natural processing tasks, we considered a number of specific aspects to make it successful in language modeling for speech recognition. By searching for the best configuration, we obtained particularly deep Transformer models. We also demonstrated that the standard positional encoding is not needed for the task of language modeling. Finally, we proposed modifications to the Transformer layer, such that we can increase the capacity of Transformer language models, while keeping the state size practical for search in speech recognition.

Also, in Sec. 4.2, we visualized the attention weights in all layers of deep Transformer language models, and identified that there are only 4 groups of layers (checked for 24 and 96-layer models), from the bottom: one input layer, a few blur layers, a few window layers, and many structured layers. Later in Chapter 7, Sec. 7.2, we confirm this finding using translation language models; where we show more explicit demonstration that only top layers learn structured attention.

Finally, at the end of this chapter, we carried out more comparisons with the state-of-the-art LSTM language models with results across 6 standard tasks in speech recognition. We showed that in all cases, we obtain good improvements over the LSTM baseline, and observed that the improvements were particularly large for large datasets with long average sequence lengths.

5. Knowledge Distillation for Language Modeling

The use of neural networks enables neural language modeling to benefit from advances in deep learning techniques which have been developed for neural networks for general purposes.

In this chapter, we are interested in applying one example of such techniques, called *knowledge distillation* [Hinton & Vinyals$^+$ 14] or *teacher student learning* [Ba & Caruana 14] to language modeling in speech recognition. The general idea of such a technique is based on the experimental evidences: training a neural network to predict the output from an already trained model as a *soft target label* can result in a better performance, than predicting the sparse hard label as in the standard cross-entropy loss. Such an approach is typically used to transfer the performance of some complex model (because of the model architecture or by some ensembling) to a *simpler* model.

The origin of the approach has its roots in the model compression by Bucă et al.'s [Buciluǎ & Caruana$^+$ 06], which consists in labelling the unlabelled data by using a large ensemble of neural networks to generate data to train a single model. Such an idea of transferring the power of a large model or an ensemble into a single model has been extended with the use of the *soft label* in the works by Ba and Caruana [Ba & Caruana 14] as *student teacher learning* and by Hinton et al. [Hinton & Vinyals$^+$ 14] as *knowledge distillation*. The technique has been used in multiple contexts of acoustic modeling [Li & Zhao$^+$ 14, Cui & Kingsbury$^+$ 17, Lu & Guo$^+$ 17, Watanabe & Hori$^+$ 17, Wong & Gales 16]. In [Chan & Ke$^+$ 15, Geras & Mohamed$^+$ 16], the transfer from an RNN was successfully used to improve feed-forward acoustic models. Early applications in language processing tasks include the machine translation [Kim & Rush 16a] and parsing [Kuncoro & Ballesteros$^+$ 16].

While applying such a method to language modeling seems straightforward at first sight, we need some small tweaks when we work with large vocabulary neural language models. The following Sec. 5.1 first introduces these knowledge distillation techniques for large vocabulary scenarios.

Then, in Sec. 5.2, we explore two application scenarios where knowledge distillation is used to transfer performance from a neural language model to another one which is originally weaker, but has a better property for search. In the first case, we simply try to transfer the performance of a powerful Transformer language model (Chapter 4) into an LSTM-RNN language model which requires much less memory at the evaluation time. In the second case, we first develop an independent motivation for improving n-gram feed-forward neural language models by letting them learn to recover the truncated context, which can be in fact formulated as a knowledge distillation problem, from an LSTM language model to an n-gram feed-forward language model.

Also importantly, this chapter serves to introduce knowledge distillation in language modeling, which will be the core technique in one of our methods for building a domain robust language model later in Chapter 6.

5.1 Knowledge Distillation for Large Vocabulary Language Models

We first present the direct application of knowledge distillation (KD) to language modeling. For a distillation from a teacher $p_T(w|h)$ to a student language model $p_\theta(w|h)$ with its parameters θ and a vocabulary V, we optimize θ to minimize the Kullback-Leibler divergence of the student model's output distribution $p_\theta(w|h)$ and that of the teacher model $p_T(w|h)$:

$$KL(p_T|p_\theta) = \sum_{n=1}^{N} \sum_{w \in V} p_T(w|h_n) \log \left(\frac{p_T(w|h_n)}{p_\theta(w|h_n)} \right) \tag{5.1}$$

which is equivalent to minimizing the following cross entropy (CE) between the student and teacher distributions:

$$\mathcal{L}_{\text{KD}}(\theta) = -\sum_{n=1}^{N} \sum_{w \in V} \ell_{\text{KD}}(h_n, w; \theta) \tag{5.2}$$

where we can introduce the notation:

$$\ell_{\text{KD}}(h, w; \theta) = p_T(w|h) \log p_\theta(w|h) \tag{5.3}$$

which allows simpler connections to other losses introduced later in this section.

In practice, this distillation loss is interpolated with the standard cross-entropy loss using an interpolation weight λ. The final objective function is therefore an interpolation as follows:

$$\lambda \mathcal{L}_{\text{KD}}(\theta) + (1 - \lambda) \mathcal{L}(\theta) \tag{5.4}$$

where $\mathcal{L}(\theta)$ is the standard cross-entropy loss, which is proportional to log perplexity:

$$\mathcal{L}(\theta) = -\sum_{n=1}^{N} \sum_{w \in V} \delta_{w,w_n} \log(p_\theta(w|h_n)) = -\sum_{n=1}^{N} \log(p_\theta(w_n|h_n)) \tag{5.5}$$

and the interpolation weight λ can be tuned to optimize the validation perplexity.

This approach assumes that we make use of the full softmax for training. When large vocabulary word-level language models are trained using some method to avoid the full softmax, the corresponding distillation loss must also be adapted accordingly. We consider both noise contrastive estimation [Gutmann & Hyvärinen 10, Mnih & Teh 12, Ma & Collins 18] and sampled softmax methods [Jean & Cho$^+$ 15] (in Sec. 5.1.1), as well as the class based factorized output [Goodman 01] (in Sec. 5.1.2).

5.1.1 Distillation with Sampling based Losses

This section presents distillation loss functions for training with sampled softmax and noise contrastive estimation. For further analytical discussions on these losses, we refer to [Gerstenberger 20]. These losses will be used in Chapter 6 where we will be discussing larger scale neural language models.

We already note here that the log uniform distribution from the frequency sorted vocabulary is used as the sampling distribution in all experiments using these sampling based losses.

Sampled softmax In the sampled softmax loss [Jean & Cho$^+$ 15], the normalization term in the softmax is computed based on a subset of words sampled for each batch from a noise distribution. Therefore, we can directly obtain the distillation loss by replacing $p_T(w|h)$ and $p_\theta(w|h)$ in Eq. (5.3) with the corresponding sampled softmax probabilities, making sure to use the same samples for teacher and student.

Noise contrastive estimation While sampled softmax only allows speed up in training, use of the noise contrastive estimation loss [Gutmann & Hyvärinen 10, Mnih & Teh 12, Ma & Collins 18] allows to achieve both faster training and a self-normalized final model. The self-normalization property makes evaluation (in particular rescoring in speech recognition) faster, because it would allow us to only compute the exponential of the logit for the target token in order to obtain the score for the target token, without having to compute the normalization term of the full softmax (which requires to compute these exponential terms for all words in the vocabulary).

The NCE loss trains the model to discriminate noise samples drawn from a noise distribution q from true data by logistic regression:

$$\mathcal{L}_{\text{NCE}}(\theta) = -\sum_{n=1}^{N} \ell_{\text{NCE}}(h_n, w_n; \theta) \tag{5.6}$$

where:

$$\ell_{\text{NCE}}(h_n, w_n; \theta) = \log g_\theta(w_n, h_n) + \sum_{\tilde{w} \in D_q^{(n)}} \log\left(1 - g_\theta(\tilde{w}, h_n)\right) \tag{5.7}$$

and the sigmoid function σ is used to obtain $g_\theta(w, h) = \sigma(s_\theta(w, h) - \log q(w|h))$ with $s_\theta(w, h)$ the logits of the model, and $D_q^{(n)}$ denotes the set of words sampled from a noise distribution q at position n.

For knowledge distillation, we similarly introduce the quantity $g_T(w, h)$ for the teacher model and obtain the following function which computes for each data point (h_n, w_n):

$$\ell_{\text{KD-NCE}}(h_n, w_n; \theta) = \sum_{\tilde{w} \in D_q^{(n)} \cup \{w_n\}} \Big(g_T(\tilde{w}, h_n) \log g_\theta(\tilde{w}, h_n) + (1 - g_T(\tilde{w}, h_n)) \log\left(1 - g_\theta(\tilde{w}, h_n)\right) \Big) \tag{5.8}$$

We thus obtain the distillation loss:

$$\mathcal{L}_{\text{KD-NCE}}(\theta) = -\sum_{n=1}^{N} \ell_{\text{KD-NCE}}(h_n, w_n; \theta) \tag{5.9}$$

In order to obtain a self-normalized student model, the teacher models are also pre-trained using the NCE loss. For further discussion, we refer to [Gerstenberger 20].

5.1.2 Class based Language Modeling Case

When neural language models have a class-based factorized output [Brown & Desouza$^+$ 92], the output distribution is computed using two softmax outputs:

$$p_\theta(w_n|h_n) = p_\theta(w_n|h_n, c(w_n)) \cdot p_\theta(c(w_n)|h_n) \tag{5.10}$$

where $c(.)$ defines the function which maps a word w to its word class $c(w)$, Eq. (5.2) can be specifically adapted for the neural language models with the class factorized outputs. This method was the mainstream approach for speeding up training, in the early stage of development of neural language modeling, when the training was still mainly done on the CPU [Mikolov & Kombrink$^+$ 11]. We refer to [Botros & Irie$^+$ 15] for a study on neural language models with the class factorized output.

For knowledge distillation, instead of directly substituting the class factorization of Eq. (5.10) into both p_θ and p_T in Eq. (5.3), we opt for minimizing the cross entropy on the word part and

class part distributions separately, which gives the following objective function:

$$\mathcal{L}_{\text{KD-Class}}(\theta) = -\sum_{n=1}^{N}\Bigg(\sum_{c\in C} p_T(c|h_n)\log(p_\theta(c|h_n)) + \sum_{u\in c(w_n)} p_T(u|h_n, c(w_n))\log(p_\theta(u|h_n, c(w_n)))\Bigg) \tag{5.11}$$

where C denotes the set of word classes.

5.1.3 Distillation with Mean Squared Error Between Hidden States

As an alternative to the previous methods which carry out distillation at the output layer, here, we consider an objective function based on the mean squared error between the final hidden layer of the teacher model $y_n^{(T)}$ and the final hidden layer in the student language model $y_n(\theta)$:

$$\mathcal{L}_{\text{KD-MSE}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} ||y_n^{(T)} - y_n(\theta)||_2^2 \tag{5.12}$$

The only constraint of the method is that the computation in Eq. (5.12) requires the teacher and the student to have the same dimension at the penultimate layer.

5.2 Application Scenarios

5.2.1 Distillation from Transformer to LSTM

In this section, we study one basic application of knowledge distillation for language modeling: distillation from a powerful Transformer language model to an LSTM language model which requires less memory at the evaluation time. As has been already pointed out in Sec. 4.3 (Chapter 4), the memory requirement of deep Transformer language models are very demanding at the evaluation time, as opposed to LSTM language models which only requires a fixed-size memory for any arbitrary sequence length. Therefore, there is an engineering interest in testing the potential of such a performance transfer. The question is whether we can transfer performance from a *growing state* Transformer to a *fixed size state* LSTM RNN.

We conducted experiments on TED-LIUM 2 (A.2). Table 5.1 shows the perplexities. We obtain improvements over the baseline LSTM while the performance of the student LSTM model still do not match that of the Transformer teacher model.

Table 5.1: *Results of **knowledge distillation**. **Perplexities** for the word-level **TED-LIUM 2**.*

Model		State size for n tokens	Params. in M	Perplexity	
				Dev	Test
Baseline	LSTM	16,384	450	73.5	71.3
Teacher	Transformer	$n\times$49,152	414	**62.0**	**60.7**
Student	LSTM	16,384	450	66.1	63.0

5.2.2 Distillation from LSTM to N-gram Feed-forward Models?

The main focus of neural language modeling today in general, is the long-span language modeling, including LSTM and Transformer language models, and also long context n-gram convolutional language models [Dauphin & Fan$^+$ 17]. In this section, we are interested in potential

improvements in short n-gram feed-forward language models. In fact, n-gram feed-forward language models are interesting when n is small, because it has the potential to be directly integrated into the traditional decoding algorithm (designed for short n-gram language models) for conventional NN-HMM hybrid ASR system [Schwenk & Gauvain 02, Huang & Sethy$^+$ 17]. However, in practice, a long n-gram context (over 20 words) is needed for a feed-forward language model to be competitive [Tüske & Irie$^+$ 16, Dauphin & Fan$^+$ 17] with LSTM-RNN language models.

Original motivation. It should be noted that language models based on the n-gram context does not know that the input it sees is only a truncated portion of the full context. We can consider training the model such that it has a chance to recover the truncated part of the context. If a well trained LSTM language model is available, it can compress the full context into a vector which can be paired to its truncated n-gram context. Learning such pairs is a vector to vector mapping problem suitable for a neural network. We explore it as a sub-task to train n-gram feed-forward language models.

Such an approach is in fact a form of knowledge distillation. We compare two approaches. The first approach is the standard transfer based on the Kullback-Leibler divergence of the output distribution of the feed-forward model from that of the LSTM using the class based output (Sec. 5.1.2). In the second approach, we alternatively minimize the mean squared error between the hidden state of the LSTM and that of the n-gram feed-forward model (Sec. 5.1.3).

We therefore consider n-gram feed-forward neural network as the *student* and LSTM-RNN as the *teacher*. The feed-forward models are based on the fully connected multi-layer perceptron (MLP) or convolutional neural network (CNN) are considered. All neural language models in this section use an output layer factorized using word classes. We consider two context sizes for the student model: 5-gram and 10-gram. The focus of this section is to evaluate the potential of knowledge transfer to improve neural language models with a short (5-gram) and a medium (10-gram) context length. We carry out experiments on Switchboard (A.4).

Baseline Neural Language Model Setups. The teacher LSTM-RNN language model consists of one projection layer of 600 nodes, one LSTM layer of 600 nodes, and an output layer[1]. The output layer is factorized using 200 word classes trained using the exchange algorithm with the bigram two-sided criterion [Kneser & Ney 91]. We use this LSTM model as the teacher model for all experiments. The student feed-forward language models has one projection layer with 100 nodes for each word, two non-linear layers, and the output layer. The dimension of the final hidden layer is set to 600 since it is tied with that of the teacher (as discussed in Sec. 5.1.3, this is a requirement for the MSE case). The dimensions of the other hidden layers are optimized to be among 600, 1000, 1200 and 1500 for baseline models as well as when knowledge distillation is used (depending on cases, either 1000 or 1200 were found to work the best). We use the gated linear unit (GLU) activation function [Dauphin & Fan$^+$ 17] instead of the sigmoid, which transforms the input vector x_t to the output vector $y_t^{(\text{GLU})}$ by using the weight matrices A, B and bias vectors c, d as:

$$y_t^{(\text{GLU})} = (Ax_t + c) \odot \sigma(Bx_t + d) \tag{5.13}$$

Similar to what was reported by [Dauphin & Fan$^+$ 17], we also found that the GLU converges faster than the standard sigmoid layer (as opposed to the case we experienced with deep Transformers of Chapter 4). In addition, we observed that a sigmoid model can also reach the same level of

[1]We note that better LSTM models are used in Switchboard experiments in other chapters, which were conducted after those presented in this chapter. We note however that these LSTM baselines in this section are much better compared with the n-gram feed-forward baseline models. Therefore, in the scope of experiments in this section, we consider these LSTM models to be good enough baselines.

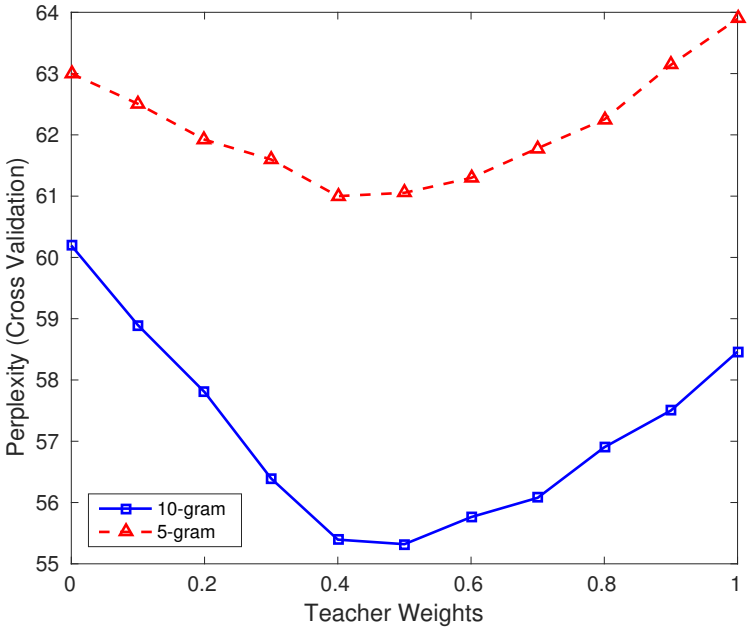

Figure 5.1: *Effect of the teacher weight λ in Eq.* (5.4) *on the Switchboard cross validation set.*

perplexity, but it requires more epochs. All neural networks are trained with stochastic gradient using Newbob learning rate scheduling. Batch size of 64 and 8 are respectively used to train the feed-forward models and the LSTM model. We construct training sequences by concatenating sentences until that we get a sequence with more than 100 words. For these experiments, all neural language models were implemented using the toolkit `rwthlm` [Sundermeyer & Schlüter+ 14].

Results for distillation with class based output. We searched for the optimal value of the interpolation weight λ for the distillation loss in Eq. (5.4) between 0 and 1. The cross validation perplexity results in Fig. 5.1 show that the optimal weights were 0.5 and 0.4 respectively for the 10-gram and the 5-gram. It should be noted that the pure knowledge distillation case λ=1 is better than the baseline case λ=0 for the 10-gram, while such is not the case for the 5-gram. Table 5.2 shows the perplexity results. We compute perplexities on the development set (Hub5_00) and the evaluation set (Hub5e_01) without using any context across the sentence boundaries such that they are consistent with the speech recognition setup in this section. For the cross-validation (CV) set, we report the perplexities using the context across sentence boundaries by concatenating multiple sentences as is the case during training[2]. We observe consistent improvements by distillation for both the 10-gram and 5-gram cases. We note that, given the short average sentence lengths in the Switchboard data (Appendix A.4), the baseline perplexities are close between the 5-gram and the 10-gram.

Larger improvements by knowledge distillation can be observed when we consider longer sequences at evaluation time. In Table 5.3 as an exception, we report perplexities computed by using contexts across sentence boundaries on Hub5_00 and Hub5e_01. We use the same sentence concatenation (of up to 100 words, without splitting sentences) as for training . We observe

[2] These experiments were conducted before getting aware of the large impact of the training and evaluation inconsistency in LSTM language models. The findings in these experiments were one of the motivations which lead us to carry out the corresponding systematic comparison which is a subject in Chapter 7.

up to 8% relative improvements for the 10-gram case. Such improvements are also potentially interesting for cross-utterance speech recognition (which is the subject of Chapter 7).

Table 5.2: *Perplexity results of **knowledge distillation** based on the class based output.*

LM		Distillation	CV	Hub5_00	Hub5e_01	#Param.
4-gram Kneser-Ney		-	75.9	74.6	65.3	7M
LSTM		-	52.2	60.8	52.4	39M
Feed-forward	5-gram	No	64.1	64.9	57.0	24M
		Yes	**61.0**	**62.4**	**54.9**	
	10-gram	No	60.9	64.2	55.4	25M
		Yes	**55.3**	**59.0**	**51.4**	

Table 5.3: *Perplexity results on Switchboard of knowledge distillation based on class based output, using **contexts across sentence boundaries (up to 100 words)**.*

LM		Distillation	Hub5_00	Hub5e_01
LSTM		-	52.2	46.0
Feed-forward	5-gram	No	62.2	54.1
		Yes	**59.5**	**51.9**
	10-gram	No	60.3	51.8
		Yes	**54.7**	**47.6**

Results for MSE based distillation. The mean square error objective based distillation (Sec. 5.1.3) in this setup aims to fit the the GLU output (Eq. 5.13) in the student model to the LSTM state. Instead, we can rather use the gated tangent unit (GTU) for the final hidden layer of the student model:

$$y_t^{(\mathrm{GTU})} = \tanh(Ax_t + c) \odot \sigma(Bx_t + d) \tag{5.14}$$

which is used in the LSTM. For all other layers, we use the GLU which we found to achieve slightly better development perplexity than the GTU in our preliminary experiments. Table 5.4 shows that the GTU effectively gives slightly better perplexities than the GLU. Though the distillation using MSE improves both 5-gram and 10-gram baseline models, the variant with the class based output (Table 5.2) gives better perplexities.

MLP vs CNN as the student model. Finally, we compare the standard MLP and the convolution based n-gram models as student models. We carry out experiments for the 5-gram case. We use the CNN which consists of 4 convolutional layers (200 filters for each layer, with the filter size 2 and the word dimension of 100) followed by one fully connected layer of dimension 600. The GLU activation is used in all layers. For distillation, we only evaluate the class based variant, which gave better perplexity than the MSE based one in the experiments above. The results are shown in Table 5.5. We first observed that the baseline CNN gives slightly better baseline perplexity than the MLP case. However, the gap disappears after distillation.

Table 5.4: *Perplexity results for MSE based distillation using the gated linear unit (GLU) or the gated tangent unit (GTU) in the final hidden layer. The baseline perplexities are copied from Table 5.2 for easy comparison.*

Context	Distillation	Activation	CV	Hub5_00	Hub5e_01
5-gram	No	-	64.1	64.9	57.0
	Yes	GLU	63.0	64.3	56.4
		GTU	**61.4**	**63.4**	**55.4**
10-gram	No	-	60.9	64.2	55.4
	Yes	GLU	57.9	61.8	53.7
		GTU	**56.4**	**60.5**	**52.6**

Table 5.5: *MLP vs. CNN with class output based distillation. The best perplexities for the MLP are copied from Table 5.2 for easy comparison. All modes are **5-grams**.*

Model	Distillation	CV	Hub5_00	Hub5e_01	#Params.
MLP	No	64.1	64.9	57.0	24 M
	Yes	**61.0**	**62.4**	**54.9**	
CNN	No	62.4	64.1	55.9	22 M
	Yes	**61.1**	**62.6**	**55.0**	

ASR and Lattice Rescoring Experiments. We carry out speech recognition experiments with the *preliminary* ASR system for Switchboard (Appendix A.4). Table 5.6 shows the word error rate (WER) results. For the 10-gram case, we observed significant improvements in WER from both class output and MSE based distillation on all subsets. Improvements in WER of up to 4% relative are obtained and the performance is competitive to the LSTM. In contrast, for the 5-gram case, the benefit from knowledge distillation is rather marginal. We conclude that a large enough n-gram context is needed for a feed-forward model to benefit from an LSTM teacher language model in knowledge distillation.

Table 5.6: *WER results on Switchboard. All results are reported after **interpolation** with the baseline count model.*

Model		Distillation Type	Hub5_00 (Dev)				Hub5e_01 (Eval)	
			CH		SWB			
			PPL	WER	PPL	WER	PPL	WER
4-gram Kneser-Ney		-	80.5	19.2	68.8	10.5	65.3	15.0
LSTM		-	63.1	17.5	52.4	**9.2**	49.7	**13.3**
Feed-forward	5-gram		65.8	17.8	56.8	9.6	53.8	13.9
		Class	**64.7**	17.8	**55.8**	**9.5**	52.9	**13.7**
		MSE	65.2	**17.6**	56.2	**9.5**	53.1	13.8
	10-gram		65.0	17.7	55.0	9.5	52.0	13.8
		Class	**62.0**	**17.4**	**52.3**	**9.2**	**49.7**	13.4
		MSE	63.1	17.5	52.7	**9.2**	50.2	**13.3**

5.3 Summary

In this chapter, we introduced knowledge distillation for large vocabulary language modeling. We have shown some of its application in improving language models which have some structure or property which is convenient for search, but whose performance lags behind models with a better architecture, namely transfer from LSTM to n-gram feed-forward language models, and from Transformer to LSTM language models. The techniques introduced in this chapter will be a core component for the second approach for domain robust language modeling in Chapter 6 when we transfer the performance of multiple domain expert models into a single model.

6. Domain Robust Language Modeling

Domain match is a key of success in data driven statistical approaches in general. Language modeling in automatic speech recognition is not an exception. That makes simple domain adaptation effective, which we have illustrated in Sec. 3.1.4 in the preliminary chapter.

However, the situation becomes completely different, once we are interested in a multi-target domain problem where we wish to build a single model which performs well across different target domains. In such a case, a simple domain adaptation strategy does not suffice anymore, since adaptation into one domain typically results in degradation in other domains.

In fact, despite advances in neural language modeling which cover most of the topics relevant for language modeling in speech recognition, there is no solution in the literature, other than naively training a large model on the all available data, for building a neural language model for such a problem. While we could hope that a large neural network could capture all local distributions, this approach can be proved in practice to be sub-optimal, since domain adaptation can further improve the model. We would ideally want to adapt to all domains. This contrasts with the case of conventional n-gram count models, in which domain specific language models [Kneser & Steinbiss 93, Iyer & Ostendorf 99] can be combined by Bayesian interpolation [Allauzen & Riley 11] to build a target domain independent mixture model.

Also, more generally, obtaining a good neural language model on a large scale multi-domain dataset still remains a difficult task in practice. This is conceptually pity for language modeling which is arguably the most data abundant task of any machine learning task, as it does not require human labeling for training. Larger data typically implies more diversity in the data distribution, which we should ideally exploit to improve language modeling.

In this chapter, we introduce *domain robust neural language modeling*, which aims at building a single neural language model which performs well across different domains at test time. In particular, we are interested in the scenario where we do not have access to any domain information about the test data.

We investigate two approaches. First, in Sec. 6.1, we propose to build a large *mixture of experts* model [Jacobs & Jordan$^+$ 91, Hampshire & Waibel 92, Tani & Nolfi 99, Shazeer & Mirhoseini$^+$ 17] where all components are parametrized by recurrent neural networks. In the following Sec. 6.2, in contrast, we propose training methods for building neural language models which are not only domain robust, but reasonable in model size and fast for evaluation by using *knowledge distillation*.

By the large scale nature of the problem involved, experiments are conducted on datasets provided by industries, namely Google and AppTek. As the domain information of the datasets itself is at the center of the problem, we include descriptions and discussion on the dataset in the corresponding sections (instead of having them in the appendix).

6.1 Recurrent Adaptive Mixture Models

In this section, we present a new architecture and a training strategy for an adaptive mixture of experts with applications to domain robust language modeling. The proposed model is designed to benefit from the scenario where the training data are available in diverse domains as is the case for YouTube speech recognition. The model we propose has two main components: an ensemble of parallel long short-term memory expert layers for each domain, and another LSTM based network which generates state dependent mixture weights for combining expert LSTM states by linear interpolation. We note that these expert components can also be parametrized by Transformers, together with a simple LSTM based mixer network.

We refer to the resulting model as a recurrent adaptive mixture model (RADMM) of domain experts. We train our model on 4.4 B words from YouTube speech recognition data [Liao & McDermott$^+$ 13]. In the YouTube speech recognition dataset, each video is tagged with one of 17 categories. Motivated by this data diversity, we design RADMM to be a model which can integrate the diversity of the data into a single neural language model. We present such a model together with a multi-stage training strategy. We evaluate our model on the YouTube speech recognition test set containing various domains, without using any domain information at the evaluation time.

6.1.1 Recurrent Adaptive Mixture Model for Language Modeling

Model Description. The architecture of the recurrent adaptive mixture model (RADMM) based language model is shown in Fig. 6.1. The building blocks of the model are: one word embedding layer shared across experts, multiple layers of parallel LSTM domain *experts*, the *mixer* LSTM network and the single softmax output layer. These components are composed following the equations below which describe the forward pass of the model. The word vector x_t of the input one hot word vector w_t is first obtained by a lookup in the *input embedding* matrix W_{emb}:

$$x_t = W_{emb} w_t \tag{6.1}$$

Such a vector is fed to each domain *expert* LSTM_k for a domain id $k \in 1, .., K$ where K is the number of pre-defined domains,

$$h_t^{(k)}, c_t^{(k)} = \text{LSTM}_k(x_t, h_{t-1}^{(k)}, c_{t-1}^{(k)}) \tag{6.2}$$

where $h_t^{(k)}$ and $c_t^{(k)}$ respectively denote the output and the cell state of the LSTM expert of the domain k.
The same input word vector x_t is also fed to the *mixer* LSTM function:

$$h_t^{(\text{mixer})}, c_t^{(\text{mixer})} = \text{LSTM}_{\text{mixer}}(x_t, h_{t-1}^{(\text{mixer})}, c_{t-1}^{(\text{mixer})}) \tag{6.3}$$

which is followed by a fully connected layer with the softmax activation function to generate the mixture weights over K domains represented by a vector g_t:

$$g_t = \text{softmax}(W_{\text{mixer}} h_t^{(\text{mixer})} + b_{\text{mixer}}) \tag{6.4}$$

The corresponding scalar components $g_{t,k}$ for each domain k, are then used as the relevance weights to combine the K LSTM expert features by linear interpolation:

$$s_t = \sum_{k=1}^{K} g_{t,k} h_t^{(k)} \tag{6.5}$$

which is used as the final feature to generate the output word distribution:

$$p(.|w_0^t) = \text{softmax}(W_{\text{out}} s_t + b_{\text{out}}) \tag{6.6}$$

where $w_0^t = w_0, w_1, ..., w_t$ is the word history. We refer to the parameters W_{out} and b_{out} as *output parameters*.

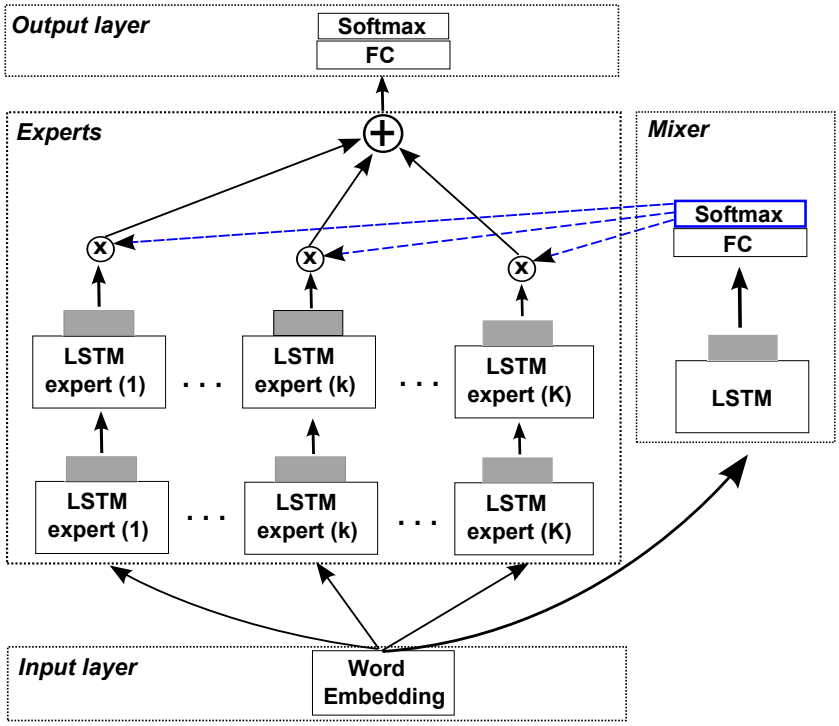

Figure 6.1: *Recurrent adaptive mixture model (RADMM) based neural language model.*

6.1.2 Training Strategy

Requirements. The role of the mixer is to generate the context dependent relevance weight for each expert. Therefore, training of the mixer requires that the experts are already well trained. Because of this constraint, the training should at least have two stages consisting of pre-training of the experts, then training of the mixer. We note that alternatively, a multi-task approach using the domain prediction loss could be considered to train the mixer. In this thesis, we train our model only using the language model perplexity as the objective function.

In addition, in order to reduce the memory requirement of the model, we tie the input word embedding across different domains (as shown in Fig. 6.1).

Finally, we experimentally found that it is necessary to initialize the final model with the *output parameters* shared across experts to train a good mixer which transfers the performance of the experts to the final model. This requires us to have the shared *input embedding* and the *output parameters* before training the experts, by training a background model beforehand. We thus end up with a 3-stage training strategy as described in the next paragraph.

3-stage training. The 3-stage training consists of the following steps. We update or freeze parameters in the 4 blocks (input layer, experts, mixer, and output layer) shown in Fig. 6.1 at different stages.

1. Train a background LSTM language model using all the data.

2. Take the input embedding and output parameters from the background model to initialize the experts. Keep these parameters constant and train each expert LSTM only using the respective domain data.

3. Take all expert LSTM parameters, input embedding and output parameters from previous stages to initialize the final mixture model. Keep all the experts and input embedding parameters constant and train the mixer LSTM on all the data while fine-tuning the output parameters.

After exploring other strategies, we found that by using this recipe, we can successfully transfer the performance of each expert on their respective domain to the single mixture model. We also include the background model as one of the experts in the mixture model.

6.1.3 YouTube Speech Recognition Dataset

The training data consist of 4.4 B running words from around 3.5 M YouTube video transcriptions. Each video is tagged with a user selected category. The distribution of the categories in the training data can be found in Table 6.1. In addition to these training data, we use 71 K words from transcriptions of an additional 125 videos as the validation data during the training of neural language models. We evaluate our model on the YouTube evaluation set of 250 K words from transcriptions of 296 videos. These datasets are the same as in [Kumar & Nirschl$^+$ 17].

Table 6.1: *YouTube training data split by categories. "Self weight" indicates the optimal interpolation weights for 5-gram count models trained on each domain when minimizing the perplexity on the subset of the validation set with the same domain (not all domains are in the validation set). 9 categories with the highest self weight are in* ***bold****.*

User selected category	Running words	%Total	Self weight
Autos & Vehicles	31 M	0.7%	6%
Comedy	30 M	0.7%	29%
Education	758 M	17.1%	77%
Entertainment	223 M	5.0%	19%
Film & Animation	103 M	2.3%	-
Gadgets & Games	79 M	1.8%	31%
Howto & Style	149 M	3.4%	48%
Movies	409 M	9.2%	31%
Music	51 M	1.2%	6%
News & Politics	344 M	7.8%	27%
Nonprofits & Activism	117 M	2.6%	-
People & Blogs	475 M	10.7%	31%
Pets & Animals	8 M	0.2%	29%
Science & Technology	175 M	3.9%	22%
Shows	1.3 **B**	29.7%	18%
Sports	61 M	1.3%	46%
Trailer	154 **K**	0.004%	-
Travel & Events	98 M	2.2%	4%
Total	4.4 **B**	100%	

Domain signals in the data. While the second and the third columns of Table 6.1 shows the diversity of the YouTube data, we can also check whether these user selected categories are relevant for language modeling in the respective category. For this purpose, we train separate 5-gram count models on each domain data. We then compute the interpolation weights that minimize the perplexity on the subset of the validation data corresponding to each category. In the last column of Table 6.1, the *self weights* indicate the interpolation weight of the domain language model for its own domain. We note that not all domains are present in this validation set. We can observe that the weights are high for most domains, which shows that the definition of domains based on the user selected categories is relevant. From this list, we choose 9 domains which give the greatest self weight, to train the domain experts.

Neural language model training setups. All experiments have been conducted at Google. All neural language models are trained on 32 GPUs using a batch size of 128 and unrolling the recurrence for 20 time steps. We use the Adagrad [Duchi & Hazan$^+$ 11] optimizer with an initial learning rate of 0.2. We use a vocabulary size of 133,008 words. In training, we use the sampled softmax by sampling 4092 words from the log uniform distribution sorted by the unigram frequency. All LSTMs used in this work have tied input and forget gate, as well as the recurrent projection as in [Sak & Senior$^+$ 14]. These setups are the same as those used in [Kumar & Nirschl$^+$ 17]. All our implementations of the neural language models are based on the TF-Slim library of TensorFlow [Abadi & Barham$^+$ 16]. In all models, we use the input word embedding size of 1024. The background model is a 2-layer LSTM with 2048 units per layer with 514 recurrent projection units.

Setups for the recurrent adaptive mixture model. In the second stage of the training (Sec. 6.1.2), we found that initializing all expert LSTMs with the parameters from the background model is helpful. Therefore, the dimensions of experts are the same as the background LSTM, except in the case of *Education*, where we get slight improvements by increasing the number of units to 4092 and train only on the *Education* data. This is reasonable given the high self weight and the amount of data in this domain shown in Table 6.1. The same recurrent projection size of 512 is used for all LSTMs. For the sampled softmax, we used the log uniform distribution based on the domain specific unigram frequency to train each expert. The mixer is a 1-layer LSTM with 1024 units and 512 recurrent projection units.

Text based experiments. Table 6.2 shows the perplexity on the validation set split by the categories. Table 6.2 has two parts: The upper part shows the perplexities on the domains for which the expert models are trained. We first notice that on some domains such as *Gadgets*, the RADMM does not achieve the performance of the domain specific expert model although it outperforms the background model. However, overall, we can observe that the performance of the different expert models is well transferred to the single RADMM which does not use any explicit domain information at the evaluation time. In addition, the lower part of Table 6.2 shows that the RADMM also gives better perplexities of up to 9% relative on domains on which no expert model is trained. Overall on the full validation set and the evaluation set, the improvements in perplexities of respectively 7% and 12% relative are obtained.

Table 6.2: *Perplexity overview for the YouTube dataset. The validation perplexities are split by categories. Background and RADMM are* ***single*** *models while Experts are* ***one model per category****.*

User selected category	Background	Experts	RADMM
Comedy	111.3	104.5	107.0
Education	93.7	72.6	78.9
Gadgets & Games	94.9	74.5	86.0
Howto & Style	98.8	81.5	88.6
Movies	145.9	143.4	142.7
News & Politics	155.0	141.6	141.4
People & Blogs	129.0	126.2	121.8
Pets & Animals	98.9	94.5	94.1
Sports	156.0	130.2	140.5
Autos & Vehicles	159.9	-	146.3
Entertainment	139.5	-	132.3
Music	136.8	-	130.7
Science & Technology	112.3	-	104.9
Shows	128.9	-	124.1
Travel & Events	92.3	-	88.4
None	130.9	-	123.9
Full validation set	118.2	-	**109.9**
Evaluation set	61.6	-	**54.0**

6.1.4 Effectiveness of the Mixer

We can examine whether the mixer function is making reasonable decisions. Four example sentences from the validation set are shown in Figures 6.2-6.5. The experts' domain are indicated on the left and input words are shown on the top. The beginning is the same for all cases. Since there is no context, the mixer chooses to mainly use the background model. Figure 6.2 is a sentence from *News* and the *News* expert is activated. Now if we look at Figure 6.3, the sentence is again from *News*, though, *People* and *Education* experts are used instead of *News*. This shows some fuzziness of the user selected categories. The domains suggested by the mixer for this sentence are also reasonable. Figure 6.4 shows an example from the category *Howto & Style*, where the model selected to use both *Education* and *Howto & Style* experts. In the example in Figure 6.5, the sentence is clearly from the category *Gadgets & Games*. We observe that while the *Education* expert is used at the beginning of the sentence, the word *game* triggers both *Gadgets* and *Sport* experts, which is also meaningful. This example also shows that the RADMM is robust to domain transitions.

6.1.5 ASR Experiments

We apply the neural language model in the second pass lattice rescoring. The lattices are generated by decoding with the first pass 5-gram count model with about 50 M n-grams and a vocabulary size of 947 K. The phone-level CTC based acoustic model described in [Soltau & Liao$^+$ 17] is used (A.6). We use the push-forward algorithm [Auli & Galley$^+$ 13, Sundermeyer & Tüske$^+$ 14] for lattice rescoring using a strong pruning by keeping only the best hypothesis per node [Kumar & Nirschl$^+$ 17]. We only use the second pass language model scores as the linear interpolation with the first pass LM scores did not improve the word error rate. The results can be found in Table 6.3. Despite this strong pruning during the rescoring, the word error rate improves from 12.3% to 12.1%, which is large in this pruning condition, since it corresponds to 29 % of the improvements obtained by the background LSTM model from the 5-gram baseline. Given the improvement in terms of perplexity, there is still potential for improvements of WER by improving the search strategy during the rescoring, at the cost of some higher computation time.

Table 6.3: *WER results on the YouTube eval set. Perplexities computed on the second pass 133 K vocabulary.*

Language model	Perplexity	WER
5-gram count	-	13.0 %
Background LSTM	61.6	12.3 %
RADMM	**54.0**	**12.1** %

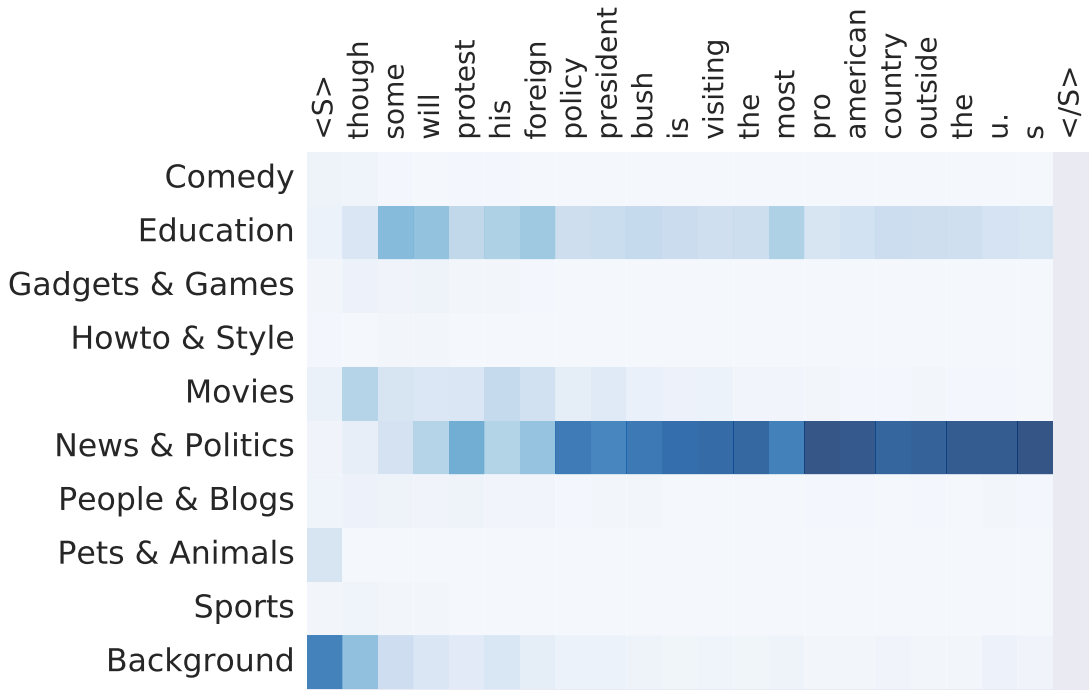

Figure 6.2: *Example 1: Category* ***News & Politics****. The x-axis corresponds to the input words. The y-axis shows the expert domains.*

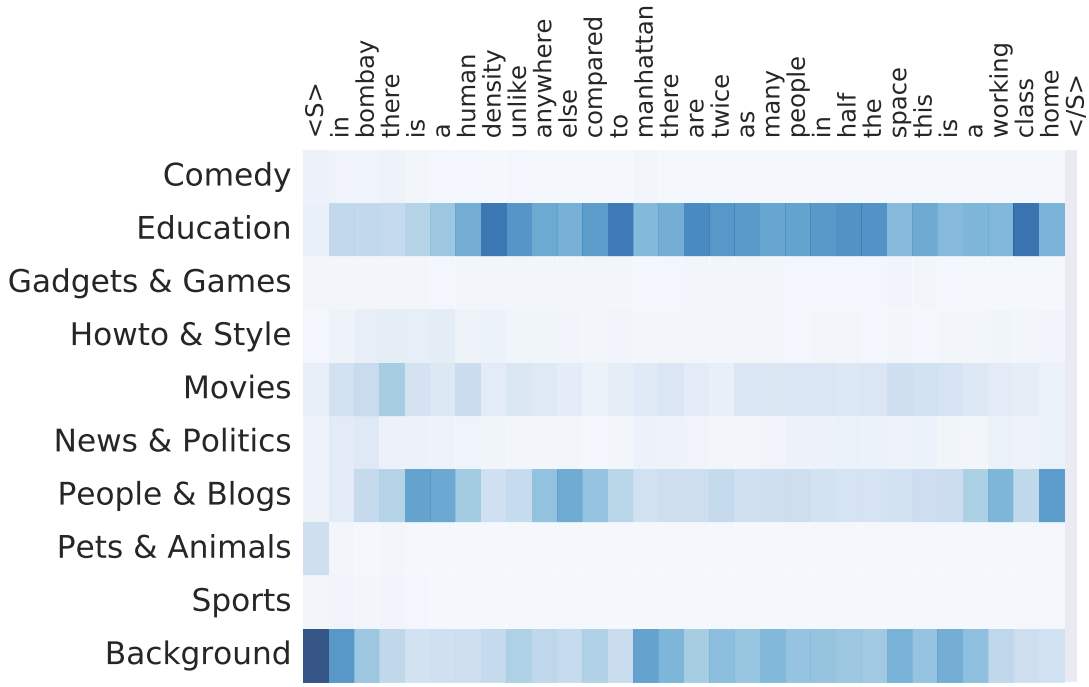

Figure 6.3: *Example 2: Category* ***News & Politics****. The x-axis corresponds to the input words. The y-axis shows the expert domains.*

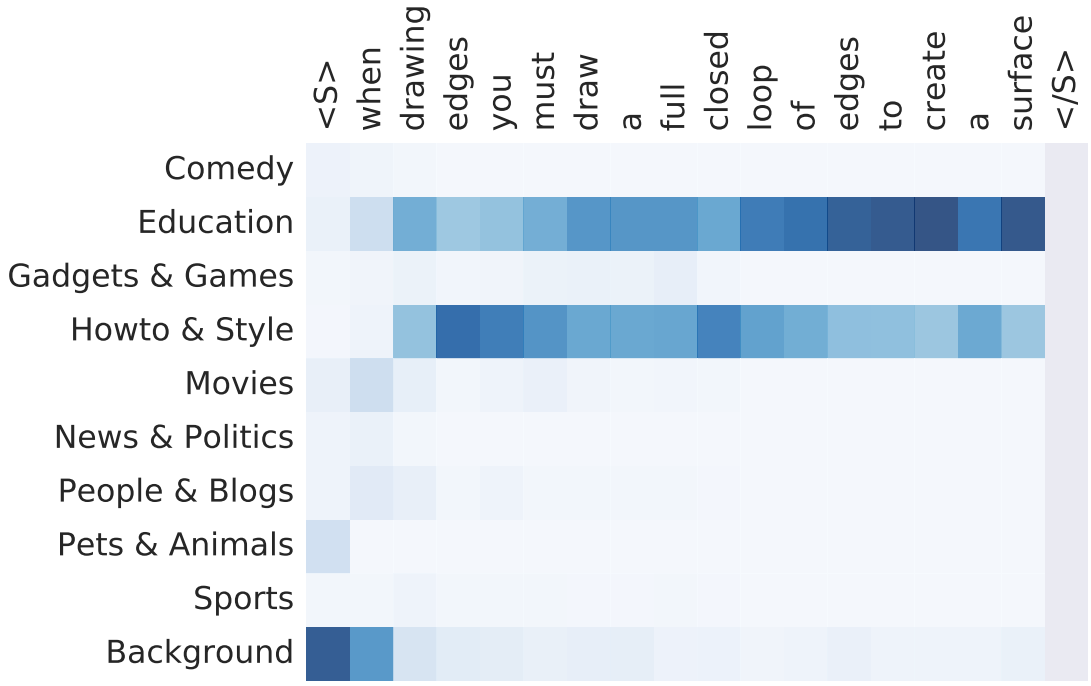

Figure 6.4: *Example 3: Category* ***Howto & Style****. The x-axis corresponds to the input words. The y-axis shows the expert domains.*

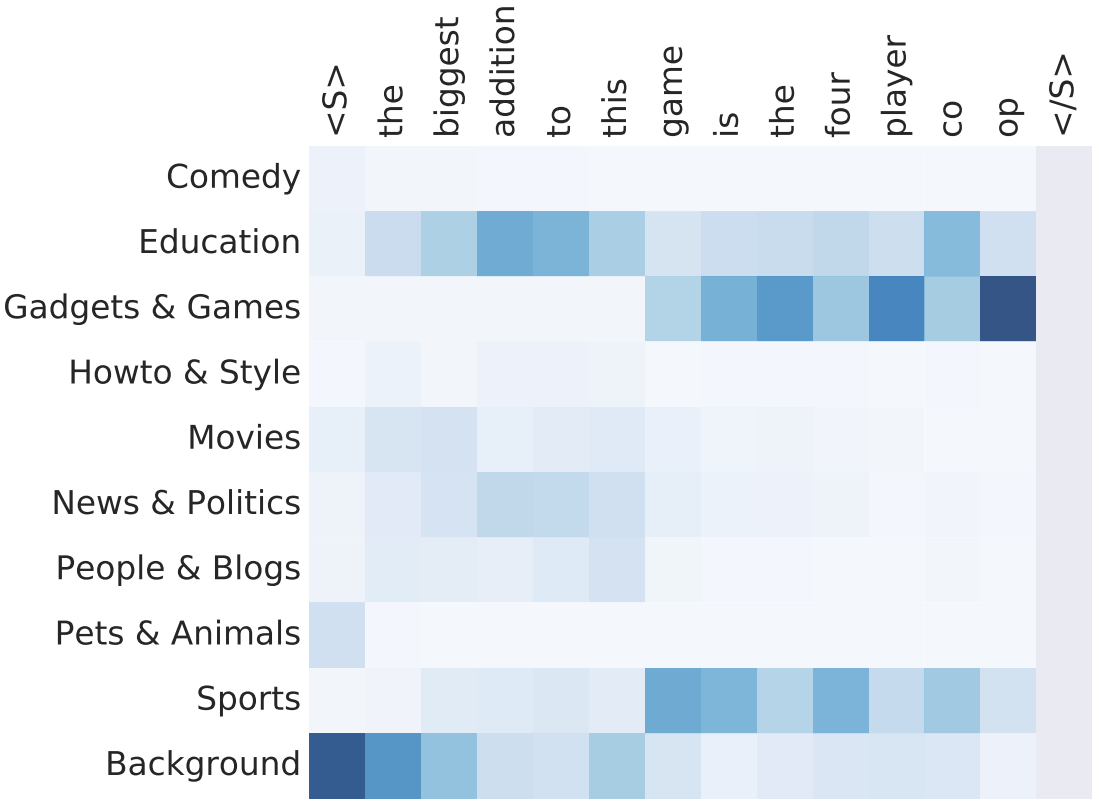

Figure 6.5: *Example 4: Category* ***Gadgets & Games****. The x-axis corresponds to the input words. The y-axis shows the expert domains.*

6.1.6 Scaling Up Further

We note that the mixture model has more parameters than the background model by construction, since it includes the background model as one of its experts. For the RADMM to have a comparable number of parameters as the background LSTM, we would require each of the experts to have very few parameters, thus, decreasing the modeling capacity. Instead, we investigate how our mixture model can scale up when we increase the size of the background model as well as that of all experts. Table 6.4 shows the perplexity results of the models with 8192 units in all expert LSTMs. All other dimensions remain the same. In this experiment, we initialize all experts using the background model. We observe that we still get 6% relative improvements in terms of perplexity on the evaluation set. While simply increasing the LSTM size of a background model has limits. In fact, we could not achieve a better perplexity by scaling up the background model to 16,384 units per layer: the best background perplexity we achieved is 110.0 on the validation set. We believe that further improvements could be obtained by increasing the number of experts instead.

Table 6.4: *Perplexities on the YouTube data of models based on **8192**-unit LSTMs.*

LM	Valid	Eval
Background LSTM	105.7	51.0
RADMM	**100.7**	**47.8**

6.1.7 Conclusion

We designed a neural network architecture motivated by data diversity. The proposed model combines domain adaptation with an LSTM based mixture of experts. We developed a training method which makes such a fusion possible. We obtained a single neural language model, RADMM, which can perform well across different domains. The visualization of mixer model's output showed meaningful model decisions, as well as robustness to domain shift within a sentence.

The RADMM is a new model in the family of mixture of experts models proposed in [Hampshire & Waibel 92] and [Jacobs & Jordan$^+$ 91], which was used with recurrent experts in [Tani & Nolfi 99]. The mixture of experts has been revisited recently as a general purpose feed-forward layer in Shazeer et al.'s work [Shazeer & Mirhoseini$^+$ 17]. We focused on building a single domain robust language model in the spirit of the Bayesian interpolation [Allauzen & Riley 11] for the n-gram count models. We achieved this goal by using an adaptive state dependent mixture weights based on the LSTM. This objective differs from previous approaches for using K-component neural language models [Shi & Larson$^+$ 13, Oualil & Klakow 17]. Some similar approaches are those which employ a gating function for combining neural models [Garmash & Monz 16, Zhang & Wu$^+$ 16] and domain-experts [Kim & Stratos$^+$ 17].

We can consider a couple of enhancements to this approach. First, the perplexity of the mixture model was not better than that of experts on some domains. Further investigations on the training strategy of the mixer can thus be interesting. Also, the computational cost of the model is high since we run all experts for each prediction. It would be interesting to investigate a possibility for faster evaluation by making the mixing weights sparser, and first running the mixer before the experts. The same approach can also be extended by using the Transformer as the base model architecture. Finally, the final model we obtain in this approach is very large by construction. In the next section, we investigate an alternative method with an objective of building compact domain robust language models by making use of knowledge distillation.

6.2 Knowledge Distillation From Domain Experts

An obvious limitation of the recurrent adaptive mixture model presented in the previous section is the large model size. Also conceptually, expert components have likely learned redundant information for the basic language modeling. As a follow-up approach, in this section, we propose training methods for building neural language models for such a task, which are not only domain robust, but which have a reasonable model size, and are fast for evaluation.

To achieve this goal, we combine knowledge distillation, which we have introduced for large vocabulary language modeling in the previous chapter 5, using pre-trained domain expert models together with the noise contrastive estimation [Gutmann & Hyvärinen 10, Mnih & Teh 12, Ma & Collins 18, Chen & Liu$^+$ 15] loss. A similar distillation approach from multiple domain expert models into a single model has been successfully used in [You & Su$^+$ 19] for acoustic modeling.

6.2.1 Knowledge Distillation for Domain Robust Language Modeling

The first step of the method we present in this section is similar to the construction of the recurrent adaptive mixture model we investigated in the previous section Sec. 6.1. We first pre-train multiple domain expert models. Then, instead of building a single mixture model out of the expert models, we carry out knowledge distillation by using these experts as teacher models. To be specific, we **interpolate** these experts model to form a single teacher distribution.

We already note here that, we conduct experiments in this section on the AppTek multi-domain dataset (A.7), which contains about 10 B words (from which we selected 1.2 B for neural model training) for language model training, with a vocabulary size of 250 K. Since this setup utilizes a large vocabulary size of 250 K, we make use of the distillation methods for large vocabulary language modeling which we introduced in the previous Chapter 5, Sec. 5.1. We carry out experiments for both sampled softmax and noise contrastive estimation cases[1].

For building the teacher model by interpolating the expert models, we consider two approaches. First, we can simply estimate a single set of interpolation weights based on the perplexity on the whole development data. Alternatively, we can also estimate different sets of interpolation weights, by optimizing them on each domain subset of the development data. Then, depending on the domain of the training sequence, we can use the interpolation weights optimized on the corresponding domain. We will denote this approach by *domain optimized* in tables in the experiments. Such an approach with a dynamic teacher in knowledge distillation has been successfully applied in [You & Su$^+$ 19] for acoustic modeling. This results in a teacher model with a better perplexity, which can potentially improve the student model through distillation.

6.2.2 AppTek English Multi-domain Dataset

As has been aforementioned, we conduct experiments on an English large multi-domain dataset provided by AppTek (A.7), which contains about 10 B words (from which we selected 1.2 B for neural model training) for language model training. In this context, we also note that, typical good improvements with neural language models are not obtained for free. They are results of careful tuning of the model hyper-parameters. In practice, for large scale tasks (with more than a few billion-word training text) containing sub-corpora with multiple domains, it is not straightforward to obtain a good neural language model [Raju & Filimonov$^+$ 19]. First, the model size must be

[1] This is in fact because before working on this project, we only had a stable training set-up for sampled softmax based training. The NCE training only started to be practically reliable, once we started applying the initialization of the bias in the softmax layer by $-\log(V)$ (and scaling it by 1.5 tends to work even better), following [Devlin & Zbib$^+$ 14] which makes the model initially self-normalized. We thank Alexander Gerstenberger who has found and tested this method in our recipe!

increased for a large amount of data. This slows down the training and tuning process which is crucial for obtaining a good neural language model, which makes it a difficult task.

We use the baseline ASR system provided by AppTek (A.7). The language model training data consists of 33 subsets with domains including news, movie subtitles (entertainment), user generated content and sport, which comprises 10.2 B words in total with a vocabulary size of 250 K words. Our domain labels are *movies*, *news*, *social media*, *user generated content (UGC)* and *voice messages (MSG)*. These target domains are defined by the development datasets, which is also consistent with the evaluation set for this dataset (while our models do not make use of the domain labels during the evaluation).

Similar to what we did for the YouTube dataset in the previous section Sec. 6.1.3, the first task is to determine which subsets of the training dataset are relevant to our target domains. For that, we train 4-gram Kneser-Ney language models on each subset of the training data. Then we linearly interpolate the models by using the interpolation weights optimized on every domain specific subset of the dev data. The optimized interpolation weights on each domain indicate the domain relevance of each training subset. Top-8 most relevant subsets are shown in Table 6.5.

Based on this analysis we assign news-04 as news expert `News` and ent-04 as movies expert `Movie`, in total 1.2 B words[2]. We pre-train separate models on each of the two datasets as **experts** for the corresponding domain. Ideally, we should first train a single background model on the whole data and fine-tune that model to the two domain subsets separately to obtain the expert models as in the previous Sec. 6.1. However, in practice, pre-training a single model on the whole data would require the model to be very large, while training "reasonable size" models separately on different subsets is still manageable. In our preliminary experiments, we found the interpolation of models trained on each subset from scratch to outperform the single model trained on the whole dataset for our model size budget. The choice of our interpolated model as baseline is therefore still meaningful.

Table 6.5: *Interpolation weights (scaled by factor 100) for each domain on the AppTek development text for 4-gram models. We removed values smaller than* 10^{-2}. *We show* ***8 most relevant subsets*** *out of 33.*

Train subset	# Running words in M	Development sets					
		All	Movies	News	Social	UGC	MSG
news-01	93	2.0	-	**10.8**	0.2	0.1	-
ent-01	94	5.2	3.7	3.1	**13.7**	**13.5**	6.9
ent-02	174	7.3	**12.7**	1.2	2.1	3.7	**11.4**
news-02	18	2.7	-	6.2	1.9	2.0	0.4
news-03	2,960	3.7	1.0	**6.3**	0.9	4.6	3.0
ent-03	651	**15.9**	**23.3**	3.1	**21.5**	**20.0**	**14.8**
ent-04	469	**22.8**	**48.1**	1.1	**28.2**	**27.0**	**20.7**
news-04	730	**27.6**	4.2	**56.7**	4.4	12.4	9.9

Base language model architectures. Our primary interest for this large AppTek dataset was to first obtain a good LSTM language model[3]. We train both LSTM and Transformer based language models, though: For distillation, a student LSTM model could also benefit from Transformer

[2] We could also merge news-01 and -04 or ent-02, -03, and -04 to train the corresponding experts, if we had more computational resources.

[3] While in principle, we might prefer to use more powerful Transformers, for this AppTek dataset, we first decided to obtain an LSTM language model, for which the code for the first pass decoding was already available. See the corresponding ASR experimental section later Sec. 6.2.6

teachers, as we have seen for the TED-LIUM 2 dataset in Sec. 5.2.1, which would allow us to make use of the good performance of Transformer models, while obtaining more memory efficient LSTM models for evaluation.

For both expert teacher and student models, the base language models have two LSTM layers with a hidden dimension of 2048 and an embedding dimension of 128, which amounts to 600 M parameters given the vocabulary of size 250 K. This model size allows us to have a reasonable training time for our experiments.

For the Transformer models we use 128 embedding dimension, 32 layers, 2048 feed-forward dimension, 768 residual dimension and 16 attention heads, which amounts to 400 M parameters.

We use the frequency sorted log-uniform distribution for the sampled softmax and NCE losses. The sample sizes of 8192 and 1024 are respectively used. We share the noise samples within the same batch[4]. We found this to be crucial for training models with the NCE loss to match performance of models trained with the full softmax. Again, we implemented our models using the TensorFlow [Abadi & Barham+ 16] based open-source toolkit RETURNN [Zeyer & Alkhouli+ 18][5]. All models were trained on a single NVIDIA Tesla V100 GPU (16 GB) at RWTH IT Center.

6.2.3 Results for Sampled Softmax based Distillation

The results for the sampled softmax case are shown in Table 6.6. All models are trained for 6 epochs until convergence. Our distillation loss scale is set to 0.5, for which we achieved the best results. The top part of the table compares the expert LSTM models with the 4-gram models trained only on the corresponding subset, which demonstrates the successful training of the expert LSTM models[6].

The bottom part of Table 6.6 shows the results of knowledge distillation. The teacher model is the interpolation between `News` and `Movie` LSTM models. The results for the two different teacher interpolation methods are presented (which we discussed in Sec. 6.2.1 above): the interpolation weights for building the teacher is either optimized for each domain (*domain optimized: yes*) subset of the development data or only once for the whole development data (*no*).

Table 6.6: *Perplexities for the **sampled softmax** case. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).*

Model Role	Model Type	Domain Optimized	Development Perplexity					
			All	Movie	News	Social	UGC	MSG
Expert `News`	4-gram	-	155.5	186.7	103.0	158.9	174.4	187.2
	LSTM		100.0	123.0	65.7	103.1	96.5	131.5
Expert `Movie`	4-gram		150.4	99.1	246.2	110.5	134.6	154.5
	LSTM		104.4	79.2	134.9	149.7	83.8	118.4
Teacher	LSTM	No	78.7	79.4	69.0	75.3	74.7	95.4
Student			**75.0**	**76.5**	**63.9**	**72.6**	**71.3**	**92.5**
Teacher		Yes	78.7	**75.7**	63.0	74.4	74.2	94.9
Student			**75.2**	77.5	**62.3**	**73.9**	**71.7**	**94.3**

We first note that even if the interpolation weight is not optimized on each domain, the interpolation results in a good teacher model which performs rather well across different domains, while the domain specific optimization gives further improvements.

[4]Again, we thank Alexander Gerstenberger who found this configuration.

[5]The configuration files are available in `https://github.com/rwth-i6/returnn-experiments/tree/master/2020-lm-domain-robust`.

[6]Please see Appendix B for further discussion on why this might not be always straightforward.

For distillation, even when no domain specific interpolation is used, the student model slightly outperforms the interpolated teacher on each domain. We therefore successfully obtain a single domain robust model.

However, we do not obtain further improvements by using a domain adaptive teacher model (*domain optimized* case in the table). The improvements of the teacher's performance obtained by domain optimization do not carry over to the student performance. We note that the word frequency used in the sampling softmax is computed on the whole training set. Use of domain specific sampling distributions might have lead to different conclusions.

6.2.4 Results for NCE based Distillation

Similar experiments were carried out for the NCE case. Table 6.7 shows the results. The perplexities are computed using the full softmax, even if the NCE loss is used for training. While we obtain slightly better perplexities compared with the sampled softmax variants (Sec. 6.2.3), the overall observations are similar: we obtain a good student model which outperforms the interpolated teacher's performance, but the domain adaptive teacher do not give extra gain in performance. The best model is obtained for the case without domain optimized interpolation of the teacher models. Therefore, we will use this model for the final ASR experiment in Sec. 6.2.6.

In addition, we must verify the self-normalization property for the models trained with the NCE loss. The variance of the log normalization term for the model is 0.02 and the mean value is -0.03, which we can consider as acceptably well self-normalized. For comparison, we note that we obtained the variance and mean values of 1.66 and 18.4 for the model trained with the sampled softmax. We then get unnormalized language model scores of 75.0 on the development and 92.1 on the evaluation set with corrections[7], compared with 77.6 and 95.3 respectively without correction.

Table 6.7: *Perplexities of the LSTM models for the **NCE** case. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).*

Model Role	Domain Optimized	Development Perplexity					
		All	Movie	News	Social	UGC	MSG
Expert `News`	-	100.7	126.5	65.3	103.9	96.9	131.6
Expert `Movie`		103.7	77.6	149.0	82.5	80.4	117.5
Teacher	No	77.1	77.5	68.0	73.8	72.1	94.0
Student		**75.0**	**76.2**	**65.0**	**72.5**	**70.4**	**91.6**
Teacher	Yes	77.1	**74.0**	**62.1**	**72.6**	71.6	93.7
Student		**75.1**	76.6	63.3	72.7	**71.4**	93.7

Can we reduce the student model size? Before moving on to the ASR experiments, we briefly investigate the possibility to further reduce the student model size which is a typical benefit we can obtain via knowledge distillation. Table 6.8 shows the results for student models with LSTM size of 1024 and 512 instead of 2048, as well as a model with a bottleneck layer before the softmax [Sainath & Kingsbury$^+$ 13] (cf. LibriSpeech experiments presented in the preliminary, Sec. 3.1.2). We observe that the model with the bottleneck achieves a compression rate of 5.7 with only up to 3.5% degradation, compared with the student model with the original model size.

[7]Following [Goldberger & Melamud 018], we correct the logits by subtracting the mean.

Table 6.8: *Perplexities of small LSTM student models on the AppTek dataset trained with the NCE loss. When an additional linear bottleneck layer is inserted before softmax (Bottleneck), its dimension is set to 512.*

Layer. dimension	Bottleneck	#Param. [M]	Dev Perplexity					
			All	Movie	News	Social	UGC	MSG
2048	No	600	**75.0**	**76.2**	**65.0**	**72.5**	**70.4**	91.6
	Yes	212	76.7	78.1	67.4	73.4	72.6	**91.5**
1024	No	300	81.2	81.3	72.0	76.0	77.6	97.2
512		163	90.5	88.0	86.9	83.1	86.6	102.6

6.2.5 Transformer Experts for an LSTM Student

Finally, Table 6.9 shows the results for distillation from Transformer teachers and an LSTM student model. The Transformer teacher model gives 12% relative improvements in perplexity over the LSTM teacher. However, only marginal improvements were obtained by this approach compared with the distillation using the LSTM teachers (Table 6.6). In contrast to the TED-LIUM experiments we presented in Sec. 5.2.1, the distillation from Transformers to an LSTM student model seems to be more difficult on this large multi-domain dataset.

Table 6.9: *Perplexities for the **sampled softmax** case using Transformer teachers. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).*

Model Role	Type	Domain Optimized	Development Perplexity					
			All	Movie	News	Social	UGC	MSG
Expert `News`	Transformer	-	91.1	118.8	55.3	92.1	86.0	124.7
Expert `Movie`			95.2	74.2	131.6	74.5	73.2	106.0
Teacher	Transformer	No	**69.4**	**73.1**	**56.8**	**65.8**	**63.8**	**88.2**
Student	LSTM		73.6	75.5	62.9	69.0	67.0	90.7
Teacher	Transformer	Yes	**69.4**	**70.1**	**52.0**	**65.1**	**63.5**	**87.8**
Student	LSTM		73.7	76.3	60.2	70.4	70.1	93.6

6.2.6 ASR Experiments

We carry out first pass decoding[8] experiments using our student models trained with the NCE loss. Table 6.10 summarizes the results. We evaluate both decoding using normalized and unnormalized language model scores. In both cases, we obtain improvements of up to 7-8% relative in WER over our 4-gram baseline model trained on all text data. This confirms that we do not need the full softmax computation at evaluation time for our NCE models. The system which uses the unnormalized scores runs 30% faster. Only looking at the time for the language model score computation shows a speedup of 40%.

Table 6.10: *WERs (%) on the AppTek data for first pass recognition experiments using LSTM student models trained with the NCE loss. The perplexity (PPL) column in the case where the explicit normalization is not carried out, indicates the pseudo-perplexity.*

Language Model	Train data	Explicit Normalization	Dev		Eval	
			PPL	WER	PPL	WER
4-gram	10.2 B	Yes	108.7	19.0	119.7	21.8
LSTM	1.2 B	Yes	**75.0**	**17.5**	**91.8**	**20.5**
		No	77.6	17.6	95.3	**20.5**

6.2.7 Conclusion

In this section, we investigated methods to successfully combine the performance of multiple domain expert LSTM language models into a compact single model. A simple knowledge distillation method using a static interpolation of domain expert models worked well in our experiments on the AppTek dataset using two domains. On the other hand, we did not manage to obtain further improvements by using a dynamically interpolated teacher model, despite its better teacher model perplexity.

The combination of the NCE with knowledge distillation was also successful. We demonstrated that the model we obtained can be used in ASR systems without explicit normalization. We achieved up to 8% improvements in WER over a strong 4-gram baseline model trained on a much larger amount of data, on the difficult large scale multi-domain task.

6.3 Summary

In this chapter, we introduced domain robust neural language modeling. We considered language modeling tasks with multiple evaluation datasets with different domains, which are particularly relevant for industry setups as has been illustrated by Google YouTube ASR and AppTek multi-domain ASR datasets used in this chapter. In such tasks, simple domain adaptation is not sufficient, since it would result in a model which is only good on one domain. Our objective was to build a single model which performs well across different evaluation domains, without making use of explicit domain information at test time. For that, we proposed two solutions. In the first approach, we presented a new adaptive mixture model together with its multi-stage training method. In the second approach, we proposed to make use of knowledge distillation from multiple domain expert models, to obtain a compact final model. We developed effective configurations for both approaches, and achieved domain robust language models in both cases. However, an open question of how to continuously extend such models to new domains and data was not addressed in this work (discussed as potential future work in Chapter 10).

[8] We thank Eugen Beck for having conducted these first pass decoding experiments with our LSTM language models [Beck & Zhou+ 19].

7. Cross-Sentence Long-Span Language Modeling

In this chapter, we are interested in exploiting contexts beyond sentence boundaries with neural language modeling for cross-utterance speech recognition. The main evaluation method for language modeling in automatic speech recognition has been the *sentence level* perplexity. This convention was carried over to neural language modeling [Mikolov & Karafiát$^+$ 10, Chen & Wang$^+$ 14]. In fact, this is a natural choice which derives from the typical decoding procedure in speech recognition which is carried out for each utterance independently. The *utterance* which we typically assume to be a *sentence* is a natural unit for parallel decoding in an off-line speech recognition scenario.

Also from the modeling perspective, the cross-sentence evaluation of language models has received limited interest since the conventional n-gram count based approach does not allow long-span modeling. In this context, the perplexity of the language model computed on the sentence level is then the effective perplexity used during the recognition. In consequence, it is also natural to carry out the training of neural language models on the sentence level [Chen & Wang$^+$ 14] to be consistent with the conventional sentence-level evaluation.

On the other hand, recurrent neural networks including the long short-term memory, or Transformers allow language models to handle long and variable length contexts. These long-span neural language models leave open the possibility for *cross-sentence language modeling*, which leverages contexts across the sentence boundaries. There is however no guarantee that the representation of such a model generalizes to any sequence length at test time. The model must learn to make use of the long context for better performance: e.g., if a model is trained on the sentence level, it is natural to assume that such a model is sub-optimal for the cross-sentence evaluation.

Despite the maturity of recurrent neural language modeling in ASR, cross-sentence long-span language modeling still remains underexplored. Surprisingly, none of the previous work [Mikolov 12, Sundermeyer & Ney$^+$ 15, Tüske & Schlüter$^+$ 18] discusses the modifications of the training mode specifically for cross-sentence evaluation, except to a limited extent in a recent work by [Xiong & Wu$^+$ 18]. Therefore, we propose to evaluate different methods for constructing training sequences, in the objective of better cross-sentence language modeling. In Sec. 7.1, we first investigate such an approach for *cross-utterance* speech recognition. In particular, the modification in training is discussed. In the same context, we analyze the robustness of both LSTM and Transformer language models with respect to the discrepancy between training and evaluation sequence lengths. Analyzing such a property is fundamental for working with neural networks which can handle variable length contexts.

In Sec. 7.2, we extend the reach of language modeling by investigating machine translation task as cross-sentence long-span language modeling. This is of interest, not only for the machine translation task itself, but also for the fact that language modeling research is in the end a quest for better, ideally generic, sequence models.

7.1 Cross-Sentence Language Modeling for ASR

We study the impact of the training sequence definition on different evaluation conditions. Our objective is two-fold in this section. First, we aim at improving language models for cross-utterance speech recognition. Second, we investigate the robustness of LSTM and Transformer language models with respect to different evaluation sequence lengths. The experiments in this section have been carried out using RETURNN [Zeyer & Alkhouli$^+$ 18][1].

A number of previous works have investigated *cross-sentence* or *document-level* language modeling to achieve better language models by leveraging the information from previous sentences. For example in [Xiong & Wu$^+$ 18], conversation-session level language modeling has been investigated for speech recognition, where language models have access to session level contexts during both training and evaluation. Some other works [Wang & Cho 16, Ji & Cohn$^+$ 16] point out difficulties of training even LSTM language models by back-propagation though time for long sequences. Therefore, instead of training a single LSTM to handle long contexts, an external model is introduced to extract a feature vector from the previous sentences which is then provided as an extra input to the language model. This approach is closely related to the original work by Mikolov et al. [Mikolov & Zweig 12]. Along the same line, hierarchical recurrent neural network models [Lin & Liu$^+$ 15, Masumura & Tanaka$^+$ 18] can be constructed to make use of the sentence level features from predecessor sentences.

In some cases, the RNN language models have been directly evaluated on the full-corpus without discussing the evaluation and training inconsistency. We note that in practice, in many works (e.g., [Sundermeyer 16]), the RNN based language models are in fact not trained on the sentence level but on a *segment* or a *concatenation of consecutive sentences*. This is motivated by the batch-mode training [Schwenk 07, Schwenk & Rousseau$^+$ 12], where construction of *training sequences* of roughly same length is crucial to reduce zero-padding in the batch for an efficient usage of the device [Chen & Wang$^+$ 14]. Interestingly, the potential impact of moving from the sentence-level to segment or concatenation level training for evaluation is rarely discussed. In some cases, the training and evaluation modes are not even fully described, while it could have direct impact on the performance comparison. When considering cross-sentence evaluation, the concatenation of sentences might be a better training strategy than the sentence-wise training. For the sentence-wise evaluation, this could be the opposite. In neural machine translation, the effect of training batch construction strategies has been studied in [Morishita & Oda$^+$ 17] while the evaluation is limited to the sentence level. In this section, we carry out a thorough analysis of this aspect.

The interest in cross-sentence modeling is not limited to RNN based language models. We have shown in Chapter 4 that Transformer language models also give excellent performance on tasks with long sequences. As opposed to RNNs, the original Transformers' memory requirements increase linearly with the number of tokens, which makes it impractical for arbitrary sequence length. The standard absolute positional encoding [Vaswani & Shazeer$^+$ 17] is also not designed to be used for unlimited sequence length. To overcome these issues, Dai et al. [Dai & Yang$^+$ 19] introduced segment level recurrence and the relative positional encoding. While this allows to successfully train a Transformer language model which can be evaluated on an arbitrary sentence, the discussion on the robustness of the standard Transformer language model without the proposed changes is not presented. Also, we have shown that Transformer language models perform well even without external positional encoding. It is however unclear whether such models are robust in terms of test sequence lengths. In this section, we also evaluate the robustness of these Transformer language models with different methods of training sequence construction.

[1] Some example configuration files are available in `https://github.com/rwth-i6/returnn-experiments/tree/master/2019-lm-cross-sentence`.

7.1.1 Problem Setup

In speech recognition, multiple consecutive utterances can be part of the same discourse, which form a *paragraph*. In this context, it is then natural to consider cross-sentence language modeling. Furthermore, these paragraphs themselves can potentially be related to each other (e.g. recordings of the same TV or radio program), which would motivate language models to be feasible on the full document or full corpus level while being able to benefit from the long context. We hypothesize that the standard sentence-level training is sub-optimal for this purpose because of the inconsistency between the training and evaluation conditions.

We aim at reducing the gap between the cross-sentence evaluation and the training conditions of language models. Since backpropagation through the whole corpus (considering the whole corpus as one sequence) is prohibitive, we study a number of training sequence construction variants with different degrees of consistency with the cross-sentence evaluation; for both LSTM-RNN and Transformer language models.

7.1.2 Training Sequence Construction Methods

The cross-sentence *evaluation* consists in *carrying over* model states across sentence boundaries. In other words, we initialize states at the beginning of a new sentence with the final states from the previous sentence. We now recapitulate the definition of states in LSTM and Transformer and specify the training sequence construction strategies for each case.

RNN based models. The recurrent neural network is an elegant solution in language modeling to handle arbitrarily long contexts. Each LSTM-RNN layer step-by-step summarizes the context of any length into two constant-size memory vectors $h_t^{(l)}$ and $c_t^{(l)}$ based on the new input x_t at time step t and the context states $h_{t-1}^{(l)}$ and $c_{t-1}^{(l)}$ from the previous time step $t-1$. As we have already seen in Eq. (3.6):

$$(h_t^{(l)}, c_t^{(l)}) \quad = \quad \text{LSTM}(h_{t-1}^{(l)}, c_{t-1}^{(l)}, x_t^{(l)})$$

In sentence-wise evaluation, these context states of the LSTM-RNN are initialized with zero vectors (or sometimes with ones) at the beginning of each sentence. In principle, transferring the contextual information from one sentence to its successor can therefore be done simply by setting the initial states in the new sentence by the last context states from the previous sentence. We refer to this process as **context carry-over** (CCO) in the following. This process does not introduce any extra computational cost for RNNs.

In cross-sentence evaluation, the zero states are only used once at the beginning of the document. The model must learn to make use of the non-zero states at the beginning of the sentence and it must learn to form states which are useful beyond the sentence boundaries. Based on these changes, we propose to study the following training sequence construction variants[2].

The most straightforward change from the baseline **sentence-wise** training is to carry over the context when moving from one batch to the next one, as illustrated in Figure 7.1 (b.). We denote this approach as **sentence-wise CCO**. This should allow the model to learn to utilize non-zero initial states at the beginning of sentences.

However, if the backpropagation is limited within the sentence, in principle, the model does not learn to generate states for cross-sentence modeling. Therefore, another training sequence we consider is the concatenation of multiple consecutive sentences (which we denote as **concatenated**), which makes the span of backpropagation beyond the sentence boundaries. As the

[2]The **bold** font in the text indicates the terminology we use later in the experimental sections to refer to these variants.

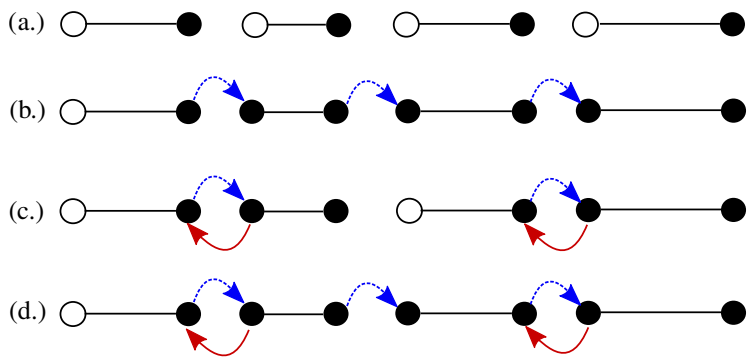

Figure 7.1: *Training sequence variants for LSTM-RNN models. Lines represent sentences. Circles represent RNN states at sentence boundaries (empty circles for zero states and filled circles for non-zero states). Dashed arrows (blue) represent state copying (context carry-over). Solid arrows (red) represent back-propagation.* (a.) ***Sentence-wise*** *training* (b.) ***Sentence-wise CCO*** (c.) ***Concatenated*** *sentences* (d.) ***Concatenated CCO****. One this same figure, we can also visualize the two evaluation modes: (a.) sentence-wise evaluation and (b.) full context evaluation.*

concatenation of the whole document into one sequence is prohibitive, we set a fixed parameter L for concatenation, and we concatenate as many consecutive sentences as the number of words in the concatenated sequence does not exceed L (typical number for L is 200 in our experiments). We therefore obtain multiple long sequences with contexts beyond sentence boundaries, which define the span of the backpropagation (Figure 7.1 (c.)). This approach can be combined with the context carry over, which gives **concatenated CCO** training (Figure 7.1 (d.)). By comparing these variants, we can evaluate both the use of non-zero initial states and the benefit of longer back-propagation through time.

One drawback of the training variants with context carry-over is that it prevents the possibility to shuffle the training sequences as they must be processed in order. Therefore, in addition, we evaluate an approach where we first pre-train the model with **concatenated** sequences and then **fine-tune** the model under **concatenated CCO**.

Transformer based models. The Transformer architecture differs from the RNN on many aspects when considering the cross-sentence training and evaluation. As we have already seen in Eqs. (4.1-4.5), the self-attention module in the l-th layer transforms the input $x_t^{(l)}$ at position t as follows:

$$\begin{aligned} q_t^{(l)}, k_t^{(l)}, v_t^{(l)} &= Qx_t^{(l)}, Kx_t^{(l)}, Vx_t^{(l)} \\ h_t^{(l)} &= \left(h_{t-1}^{(l)}, (k_t^{(l)}, v_t^{(l)})\right) \\ y_t^{(l)} &= x_t^{(l)} + W_0 \,\text{Attention}(h_t^{(l)}, q_t^{(l)}) \end{aligned}$$

where Q, K, V, respectively denote query, key, value projection matrices, and W_0 denotes the projection matrix for the residual connection [He & Zhang$^+$ 16a]. Attention denotes the scaled multi-head dot product self-attention [Vaswani & Shazeer$^+$ 17]; for each new input, the key $k_t^{(l)}$ and value $v_t^{(l)}$ vectors are concatenated in the feature dimension $(k_t^{(l)}, v_t^{(l)})$, and further concatenation with the previous state $h_{t-1}^{(l)}$ is performed in the time dimension to form the new Transformer state $h_t^{(l)}$. In contrast to the RNN which can handle arbitrary context length with a constant memory size, the standard Transformer [Vaswani & Shazeer$^+$ 17] requires to store key $k_t^{(l)}$ and

values $v_t^{(l)}$ vectors in each layer for all predecessor positions. The size of the Transformer state vector $h_t^{(l)}$ above grows linearly with the context size.

While this makes it impractical to apply the Transformer directly on extremely long sequences such as a full corpus, we can still consider some options to evaluate the Transformer on a cross-sentence scenario. First, the simplest approach is to evaluate Transformer models on the concatenation of multiple sentences as already described above for RNNs as **concatenated**, as much as the memory resource allows it. Alternatively, we can use a segment-wise recurrence as in the Transformer-XL [Dai & Yang$^+$ 19] with a fixed context window size: the evaluation is carried out for every n-token segment, and for evaluation of the current segment the model has access only to the states from the previous and current segment. We denote this as **segment-wise CCO**. In principle, the same approach can be done using the sentence as the unit of recurrence instead of the fixed size segment; we denote this approach as **sentence-wise CCO**.

Another important aspect of the Transformer is the positional encoding. If the Transformer uses absolute positional encoding (such as the one based on the sinusoidal functions as described in the original work [Vaswani & Shazeer$^+$ 17]), the model is unlikely to generalize to cross-sentence evaluation where the model has to deal with unseen absolute positions. Again, Transformer-XL [Dai & Yang$^+$ 19] solves this issue by using relative positional encoding (an improved variant from [Shaw & Uszkoreit$^+$ 18]), which would however only work in the case of segment-wise context carry-over with a fixed segment length, in order to avoid unseen relative distances.

In this context, we are particularly interested in studying the Transformer language models without positional encoding which we developed in Chapter 4, Sec. 4.2. A previous work has shown that Transformer language models do not require positional encoding [Irie & Zeyer$^+$ 19a]. It is unclear whether such models only work properly for sentence-wise training and evaluation or they can also be used in the context of cross-sentence evaluation such as **concatenated** sentences or **sentence-wise CCO**.

7.1.3 Experimental Setups

We carry out experiments on two datasets: Switchboard 300 h (Appendix A.4) and Quaero English broadcast news datasets (Appendix A.3). In this section, consideration of the sequence lengths in the training data has a central role. We therefore provide the statistics for both datasets in Table 7.1 (instead of a simple link to the appendix). The standard Switchboard dataset has relatively short training sentences: only 8 sentences are longer than 100 words.

It is also common that language model training data are not fully pre-processed in such a way that each sequence corresponds to a sentence. Such a scenario is represented by the Quaero dataset which contains about 0.2% of the training sequences which are longer than 100 words.

The vocabulary sizes are 30K and 150K respectively for Switchboard and Quaero tasks. The test sets of both datasets have a paragraph structure. Such information can be extracted from the recording IDs without having access to the actual contents of the test sets. The Switchboard Hub5_00 and Hub5e_01 datasets respectively contain 40 and 60 paragraphs of roughly 1000 words. The Quaero development and evaluation sets contain 10 and 8 paragraphs which roughly contain 4000 words each.

7.1.4 Cross-Utterance ASR via Lattice Rescoring

We apply our long-span neural language models to cross-utterance speech recognition by a second pass lattice rescoring. We use the baseline ASR systems for Switchboard (A.4) and Quaero (A.3). For both recurrent neural networks and Transformers, we use the push-forward algorithm (Sec. 1.2.2). The definition of the states to be stored at each node is extended from the original work for the RNN to the Transformers' states as defined above in Sec. 7.1.2. The first pass

Table 7.1: ***Sentence lengths statistics*** *on Switchboard and Quaero training and evaluation datasets.*

		Run. words	Number sentences	Avg. length	Max	Longer than 100
Switchboard Train		27M	2.5M	11	146	**8**
Hub5_00	Total	45K	4.3K	10	72	0
	CH	23K	2.5K	9	51	
	SWB	22K	1.8K	12	72	
Hub5e_01		65K	5.7K	11	88	
Quaero	Train	50M	3.1M	17	2475	**7519**
	Dev	40K	1.4K	29	95	0
	Eval	36K	1.1K	31	91	

recognition is carried out using the standard 4-gram count based language model for each utterance independently to generate lattices. The cross-sentence context is then introduced by the neural language model during lattice rescoring. The lattices are rescored one after another in the order of the utterances. After rescoring one lattice, the state from the best hypothesis is extracted and used as the state of the initial node in the next lattice.

Prefix state caching for rescoring with Transformers. The memory requirement for lattice rescoring with Transformers in cross-sentence recognition can be dramatically large with a naive implementation, because the size of each Transformer state to be stored with each hypothesis at every node in the lattices gets very large with the long cross-sentence context. However, the extraction of the 1-best state after rescoring each lattice allows to minimize this requirement. In fact, since we extract the best state after rescoring a lattice and pass it as the state for the initial node of the next lattice, this *prefix state* is shared across all Transformer states in the next lattice. We store this prefix state only at one location and only store *intra-lattice states* for each hypothesis at each node. The full Transformer state can be retrieved on demand at evaluation time by concatenating the two parts of the state. This allows cross-utterance lattice rescoring with a Transformer language model with almost the same disk space requirement as the utterance independent lattice rescoring.

7.1.5 Text based Experiments: LSTM-RNNs

We train LSTM-RNN language models using different training sequence construction methods we described in Sec. 7.1.2. When we use **concatenated** sentences for training, we concatenate consecutive sentences until the sequence length gets longer than 200 words. This set-up is the same for both datasets. For the concatenation at test time, the concatenation length in terms of number of words is specified in the corresponding tables.

While the concatenation makes the backpropagation span longer, this does not necessary speed down the training. On the contrary, it makes the training faster in our set-ups (about twice faster compared with our baseline sentence-wise training where we completely randomize the sentences) because the concatenation makes the training sequence lengths homogeneous, which allows avoiding zero-padding in the batch to a large extent.

We use a generic model architecture for both tasks. The model has an input embedding layer of dimension 128, two LSTM recurrent layer of dimension 2048. We apply dropout with a rate of 40% between each feed-forward layers except after the input embedding layer for which we use the rate of 20%. The models are trained using the plain stochastic gradient descent with gradient norm clipping (as in our standard recipe, Sec. 3.1.2).

Table 7.2 shows the perplexity results for Switchboard. We first evaluate the baseline sentence-wise training. We first observe that the LSTM-RNN language model trained only on the sentence-level can achieve better perplexity by a short concatenation (about 60 words) than sentence-level evaluation. However, some perplexity degradation is observed when the concatenation length gets longer (about 200 words) and when the model is evaluated on the full corpus without state reset.

Training with context carry-over (**sentence-wise CCO**) results in a model which gives better perplexity when it is evaluated on the full corpus without state reset. However, it should be noted that such a model gives a significantly bad perplexity for the sentence-wise evaluation. By extending this CCO approach with longer backpropagation using concatenated sentences (**concatenated CCO**) effectively gives further improvements.

However, slightly better perplexity can be obtained by training on concatenated sentences without context carry-over (**concatenated**) when such a model is also evaluated on the concatenated sentences. We speculate that this might be due to the fact that CCO prevents data shuffling. Therefore, we introduced CCO as a fine-tuning of the model pre-trained with **concatenated** sequences (we randomize the sequences after concatenation). This **CCO Fine-tuning** finally results in the best performance for the evaluation on the full corpus without state reset. A similar trend has been observed for Quaero as shown in Table 7.3.

Table 7.2: *Perplexities of **LSTM-RNN** on **Switchboard**.* CCO denotes context carry-over. We report average sequence length information for Hub5_00 which is similar to Hub5e_01.

Train	Eval		Perplexity	
	State Reset	Avg Len.	Hub5_00	Hub5e_01
Sentence-wise	Sentence	10	50.7	43.2
	Concatenated	60	**48.3**	**42.3**
		200	50.5	44.4
	Full Corpus	45K	53.7	47.5
Sentence-wise CCO	Sentence	10	63.4	53.5
	Concatenated	200	50.4	43.8
	Full Corpus	45K	**47.0**	**40.9**
Concatenated CCO	Sentence	10	98.6	77.0
	Concatenated	60	52.3	44.5
		200	46.7	40.3
	Full Corpus	45K	**42.9**	**38.1**
Concatenated	Sentence	10	52.4	44.5
	Concatenated	200	**42.3**	**37.3**
	Full Corpus	45K	42.7	38.5
+ CCO Fine-tuning	Sentence	10	60.7	51.0
	Concatenated	60	45.6	39.7
		200	42.5	37.3
	Full Corpus	45K	**40.5**	**36.1**

Table 7.3: *Perplexities of **LSTM-RNN** on **Quaero**. CCO denotes context carry-over. We report average sequence length information for the development set which is similar to the evaluation data.*

Train	Eval		Perplexity	
	State Reset	Avg Len.	Dev	Eval
Sentence-wise	Sentence	29	84.5	86.3
	Concatenated	200	**81.3**	**84.1**
	Full Corpus	40 K	85.2	87.2
Concatenated	Sentence	29	86.8	88.6
	Concatenated	200	77.8	81.0
	Full Corpus	40 K	**77.0**	**80.1**
+ CCO Fine-tuning	Sentence	29	94.2	96.5
	Concatenated	200	80.1	83.5
	Full Corpus	40 K	**74.6**	**77.7**

7.1.6 ASR Experiments: LSTM-RNNs

The improvements in terms of perplexity obtained from cross-sentence evaluation is well known to be typically hard to carry over to a reduction in WER [Kuhn & De Mori 90, Jelinek & Merialdo^{+} 91] compared with the improvements on the sentence-level (Sec. 3.3 in the preliminary chapter). The Switchboard results are presented in Table 7.4. The LSTM language model scores are interpolated with the 4-gram count model used for the first pass to generate the lattices. The interpolation weights are optimized on the cross-validation set (A.4). We use two separate interpolation weights: one for the sentence-wise evaluation and another one for both paragraph level and full corpus level evaluation. We report perplexities based on the true transcriptions in all cases. In case of the cross-sentence evaluation, this may deviate from the effective perplexity used in recognition which is based on the contexts given by the recognition outputs.

While we confirm that the well known correlation for the utterance-wise recognition seems to be more difficult to be directly applied in this cross-utterance context, the best WERs are obtained in the cross-sentence lattice rescoring with the model trained with concatenated sentences and fine-tuning with context carry-over. An improvement from the WER of 10.1% in the standard utterance-wise recognition to the WER of 9.8% in the cross-sentence recognition is achieved. A similar trend is observed in Table 7.5 for Quaero.

Table 7.4: ***ASR results** of **LSTM-RNN** on Switchboard 300 h **Hub5_00** set. Perplexities (PPL) after interpolation with the 4-gram Kneser-Ney language model. CCO denotes context carry-over.*

Training	Recognition	SWB	CH	All	
		WER		PPL	WER
4-gram baseline		8.1	15.4	74.6	11.8
Sentence-wise	Sentence	6.9	13.4	50.1	10.1
	Paragraph	6.8	13.1	49.3	9.9
	Full Corpus	6.8	13.4	50.3	10.1
Concat + CCO Fine-tune	Sentence	7.0	13.8	53.3	10.4
	Paragraph	**6.7**	13.1	**40.4**	9.9
	Full Corpus	**6.7**	**12.9**	**40.4**	**9.8**

Table 7.5: ***ASR results** of **LSTM-RNN** on **Quaero**. Perplexities (PPL) are after interpolation with the 4-gram Kneser-Ney language model. CCO denotes context carry-over.*

Mode		Dev		Eval	
Training	State Reset	PPL	WER	PPL	WER
4-gram baseline		132.7	11.6	131.2	9.8
Sentence-wise	Sentence-wise	81.4	9.0	82.6	7.7
	Full Corpus	77.1	8.9	79.5	7.6
Concat + CCO Fine-tune	Sentence-wise	85.7	8.9	87.2	7.7
	Full Corpus	71.9	**8.7**	74.4	**7.5**

7.1.7 Text based Experiments: Transformers

We train 30-layer Transformer language models with the inner feed-forward dimension of 2048 and the self-attention total dimension of 512 for all conditions for both tasks. We use 8 heads for self-attention and apply 20% dropout for both feed-forward layer and the self-attention. We primarily focus on models without positional encoding.

The perplexity results for Transformer based models are presented in Table 7.6 for Switchboard and Table 7.7 for Quaero. First of all, we observe that the model trained on the sentence-level does not perform well when it is evaluated on longer sequences (**concatenated** 60 and 200). The degradation is much larger than in the case of LSTM-RNNs (Table 7.2). The LSTM-RNNs therefore seem to be *much more robust* with respect to the sequence length compared with Transformers which are based on attention.

When we train Transformers on **concatenated** sentences and evaluate on **concatenated** sequences of similar length, the resulting perplexity is much better compared with the sentence-wise trained model evaluated on the sentence level. However, we again observe that the model does not perform well when it is evaluated on much longer sequences (concatenation up to 500 words). This indicates that the Transformer models without positional encoding fail in generalizing to the sequence lengths which are inconsistent with those seen during the training. Training and evaluation on longer concatenated sequences (up to 1000 words) give further improvements in perplexity while confirming the negative statement on its generalization ability in terms of test sequence lengths.

We also evaluate models trained on **concatenated** sentences in a **sentence wise CCO** manner, which result in a failure mode giving a very high perplexity. Finally, we train Transformers without positional encoding in a **segment-wise CCO** fashion like in the Transformer-XL [Dai & Yang$^+$ 19]. We note that the only difference between this model and the Transformer-XL is the relative positional encoding in the Transformer-XL. We observe that the model gives a reasonable perplexity, however the performance is behind the Transformers trained and evaluated on concatenation of sentences and the Transformer-XL which uses the relative positional encoding. Transformer-XL is precisely designed for this type of evaluation with a fixed size attention window. This indicates that the Transformers without positional encoding can not fully transfer and leverage contextual information from one segment to another.

Table 7.6: *Perplexities of **Transformers** on **Switchboard** (no positional encoding is used except for the model in the last row). CCO denotes context carry-over.*

Train	Eval		Perplexity	
	Mode	Avg. Len	Hub5_00	Hub5e_01
Sentence-wise	Sentence-wise	10	**48.1**	**40.7**
	Concatenated	60	48.8	42.4
		200	70.5	64.6
Concatenated (200)	Sentence-wise	10	51.6	43.8
	Concatenated	200	**39.8**	**35.2**
		500	215.6	189.5
Concatenated (1000)	Sentence-wise	10	53.1	44.9
	Concatenated	500	38.2	34.0
		1000	**37.7**	**33.4**
	Sentence-wise CCO	20	187.0	174.2
Segment-wise CCO	No position enc.	200	45.5	39.9
	Relative pos. enc.		**36.3**	**32.4**

Table 7.7: *Perplexities of **Transformers** on **Quaero**. CCO denotes context carry-over.*

Train	Eval		Perplexity	
	Mode	Avg. Len.	Dev	Eval
Sentence-wise	Sentence-wise	29	**74.1**	**76.1**
	Concatenated	200	83.9	86.4
Concatenated	Sentence-wise	29	74.5	76.5
	Concatenated	100	66.7	69.8
		200	**63.7**	**66.8**
		500	75.7	81.5
	Sentence-wise CCO	60	325.1	324.6
Segment-wise CCO		200	89.0	91.1

7.1.8 ASR Experiments: Transformers

Finally, we carry out cross-utterance lattice rescoring to study the benefit of perplexity improvements offered by cross-sentence contexts in Transformers (Tables 7.6 and 7.7). Table 7.8 shows the results for Switchboard. The baseline approach is the utterance-wise rescoring using a Transformer model trained on the sentence-level. For the cross-sentence rescoring we use the Transformer model trained on **concatenated** sentences up to 1000 words from Table 7.6.

We carry out two state resetting schemes in rescoring: reset every 10 utterances and reset at paragraph boundaries. We observe that we also obtain improvements in terms of WER from cross-sentence contexts using Transformer models. For Quaero (Table 7.9), we found it difficult to carry out cross-utterance lattice rescoring even on the paragraph level because of the high memory requirement. Instead, we only concatenated every 2 utterances to evaluate the effect of cross-sentence context, which already gives some improvements.

Table 7.8: ***ASR results*** *of* ***Transformers*** *on Switchboard-300 h* ***Hub5_00*** *set. Perplexities (PPL) are after interpolation with the 4-gram Kneser-Ney language model.*

Training	State Reset in Recognition	SWB	CH	All	
		WER		PPL	WER
Sentence-wise	Sentence	6.9	13.3	47.2	10.1
Concatenated	Every 10 Sent.	**6.7**	13.0	40.6	**9.8**
	Paragraph	**6.7**	**12.9**	**36.5**	**9.8**

Table 7.9: ***ASR results*** *of* ***Transformers*** *on* ***Quaero****. Perplexities (PPL) are after interpolation with the 4-gram model.*

Mode		Dev		Eval	
Training	State Reset	PPL	WER	PPL	WER
Sentence-wise	Sentence-wise	70.8	8.6	73.3	7.4
Concatenated	Every 2 sentences	66.8	**8.5**	69.3	**7.3**

7.1.9 Conclusion

In this section, we thoroughly studied the impact of training sequence construction methods in different evaluation conditions, for both LSTM and Transformer based language models. In the context of cross-utterance speech recognition, we demonstrated that the use of long training sequences via concatenation and the context carry-over effectively close the gaps between training and evaluation conditions; we obtained improvements in terms of both perplexity and word error rate. Contemporary to our work, [Narayanan & Prabhavalkar$^+$ 19] has conducted a similar investigation for end-to-end speech recognition systems.

These results also shed light on the importance of reporting the exact training and evaluation conditions under which the perplexity is computed, in order to discuss accurate improvements in language modeling.

Finally, we also compared LSTM-RNNs and Transformers' fundamental property of being robust, when the training and evaluation conditions are different in terms of context lengths; globally, we found LSTM models to be more robust than Transformers.

7.2 Translation as Long-Span Language Modeling

[Mikolov & Zweig 12] had noted that language models which are conditioned on a context vector from a sentence in another language can be a useful model for machine translation. Such an approach has been rebranded as a sequence-to-sequence learning [Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15] and has seen lots of success in machine translation and beyond.

More recently, the power of Transformer language models has motivated works in directly considering translation as a single sequential task in which the sentence in the target language follows the one in the source language [Radford & Wu$^+$ 19, Raffel & Shazeer$^+$ 19, He & Tan$^+$ 18]. Such an approach is interesting when studying language modeling, because if the sequence model is powerful enough, we might not need a prior knowledge about the source and target separation, which is implemented in typical sequence to sequence learning approaches, via an explicit separation of encoder and decoder components. Potentially, it also can be more consistent when considering pre-training of the model, for example by using the monolingual source and target text data.

In this short section, we briefly investigate such an approach as a part of cross-sentence long-span language modeling problem, using tools and heuristics we obtain earlier in Chapter 4, in the objective of better understanding the long-span modeling ability of Transformer language models.

7.2.1 Task Definition

The machine translation task consists in generating sentences in target language from the corresponding sentence in the source language. Today's mainstream neural machine translation approach treats the problem as a sequence-to-sequence learning using an encoder component which processes the source sentence `<f>` a b c `</f>` and the decoder which learns to generate the target sentence `<e>` x y z `</e>`.

This problem can be simply formulated as a cross-sentence language modeling task, once we concatenate these two sequences into one: `<f>` a b c `</f>` `<e>` x y z `</e>`. Each pair of parallel training sentences of machine translation becomes one long sentence to train a language model. For decoding, we provide the language model with the source text plus the target start token, i.e. `<f>` a b c `</f>` `<e>`, as the context. The standard beam search can be then carried out as is the case for the baseline translation model.

We are interested in studying Transformer language models for this problem, in terms of performance but also how the model allocates its capacity (different layers), in comparison to our analysis of Sec. 4.2 in Chapter 4.

7.2.2 Experimental Results

We present experiments on the standard WMT 2016 Romanian to English task (Appendix A.8). We compare the Transformer language model with the standard encoder-decoder attention neural machine translation model (NMT). The baseline model was provided by the authors of [Nix & Kim$^+$ 19], which is configured as in the base translation Transformer settings [Vaswani & Shazeer$^+$ 17]: 6 layers in each of encoder and decoder, feed-forward layer size of 2048, the model dimension of 512, and it uses 8 heads. For a fair comparison with this baseline, we use the same configurations in our language model and we set the number of layers in the language model to be the sum of encoder and decoder Transformer layers in the baseline translation model, which is 12.

Table 7.10 presents the results. As has been also reported in [Raffel & Shazeer$^+$ 19], we find that the baseline encoder decoder Transformer outperforms the translation language model. We tried both the reset of positional encoding at the sentence boundary between the source and target

sentence, as well as the use of an additional language ID embeddings as input, as in [Conneau & Lample 19], but only a marginal improvement in TER (from 54.7 to 54.3%) was obtained.

In [Raffel & Shazeer$^+$ 19], it has been reported that a language model augmented with an encoder style bi-directional self-attention on the source part of the context improves results, which is also used in [He & Tan$^+$ 18]. However, such an approach is then not treating the problem as the standard language modeling anymore, which is then out of scope of our study.

These results seem to indicate that there are still room for improvements in language modeling by purely improving the base sequence model, which could better exploit the structure given by the problem.

Table 7.10: *BLEU and TER results for WMT 2016 Romanian-English task. The baseline NMT performance was provided by Arne Nix, which is reported in [Nix & Kim$^+$ 19].*

System	newsdev2016	
	BLEU	TER
Baseline NMT	**34.7**	**52.3**
Translation LM	32.8	54.7

7.2.3 Visualizing Functionality of Each Layer

While the performance of the model compared with the baseline model is rather disappointing, it is still of interest, in comparison with our findings in Sec. 4.2 (Chapter 4), to analyze how a Transformer language model allocates the functionality of its layers.

The attention weights for the 12-layer model are shown in the Figures 7.2 to 7.6, from bottom to top layers. The model makes use of the standard positional encoding, but it does not reset the position at the source and target sentence boundary. Language embeddings are also not provided. The model is therefore trained without any explicit signal about the translation task, except from the data itself.

We note that it is useful to split each figure into 4 rectangular components (left, right, top, bottom): the left-bottom region corresponds to source-source attention, and the right-top region corresponds to target-target attention. These attention types are therefore *intra-sentence*. The left-top region is the cross-sentence attention. The right-bottom region is to be ignored as the language model does not have access to the future context.

We observe that the first layer (Figure 7.2) mainly focuses on the new input to the network, and in the following layers attention is rather intra-sentence and blur (Figure 7.3), then rather focused on local n-gram (Figure 7.4; somehow skipping the latest position). These observations are similar to what we have already presented for the standard language models for speech recognition. In these bottom layers, the attention is therefore mostly intra-sentence.

Interestingly, from the middle of the model, in the 6-th layer (Figure 7.5), the cross source-target sentence attention starts to emerge, and finally all top layers have strong cross-sentence attention trends, as illustrated in Figure 7.6. This is in fact again consistent with what we found for the ASR Transformer language models. In Sec. 4.2, we had identified that the top layers of the Transformer language models were *structured* layers which detect specific patterns. In case of translation language models, it makes sense that these patterns are some alignments between the source and target sentences.

The model therefore identifies the source-to-target mapping structure from the simple language modeling style training.

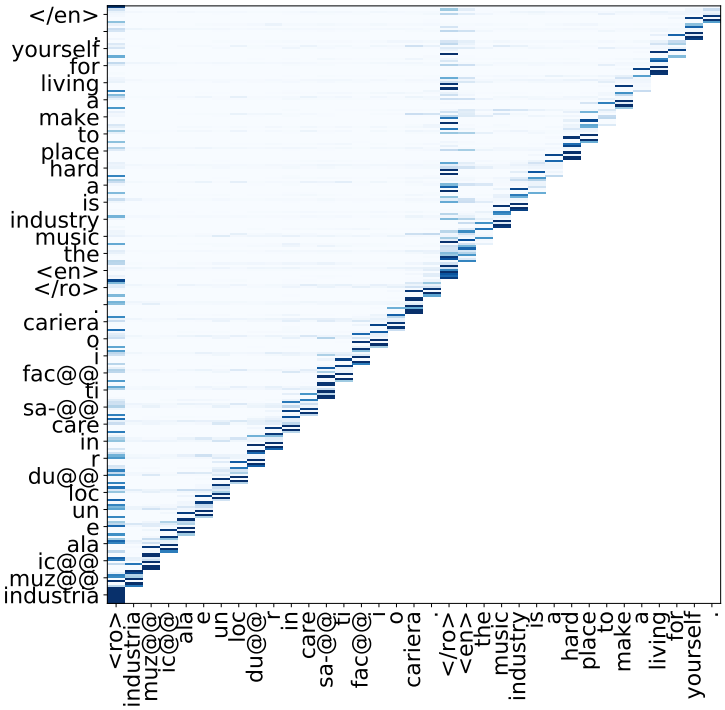

Figure 7.2: *Attention weights in the **first layer**. The x-axis corresponds to the input tokens. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

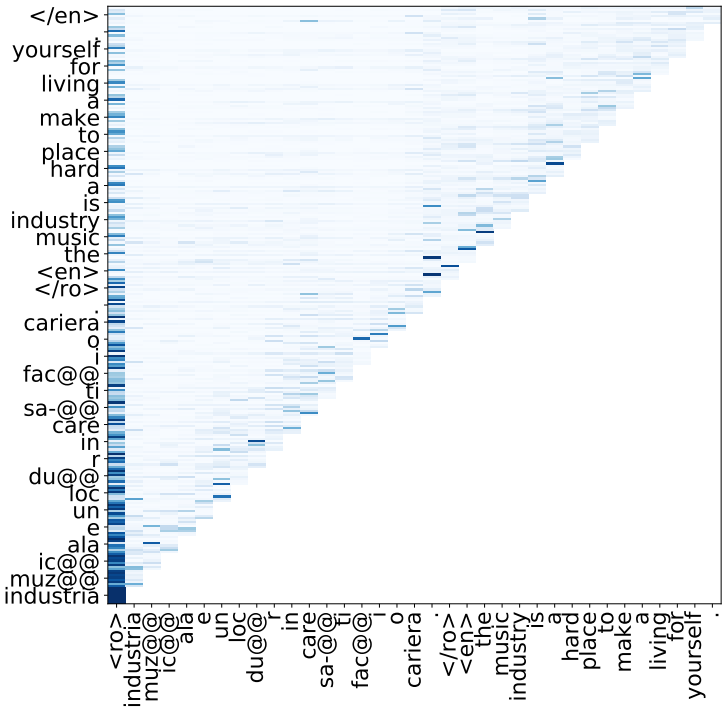

Figure 7.3: *Attention weights in the **3rd layer**. **Attention is rather intra-sentence and blur.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

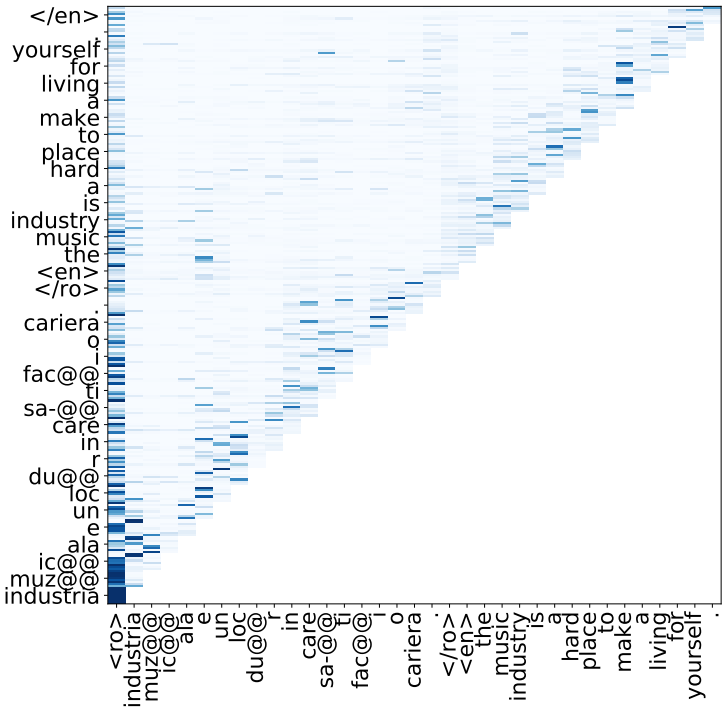

Figure 7.4: *Attention weights in the **5th layer**. **Attention is rather intra-sentence in layers in this region, with some focus on the n-gram (somehow skipping the latest token).** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

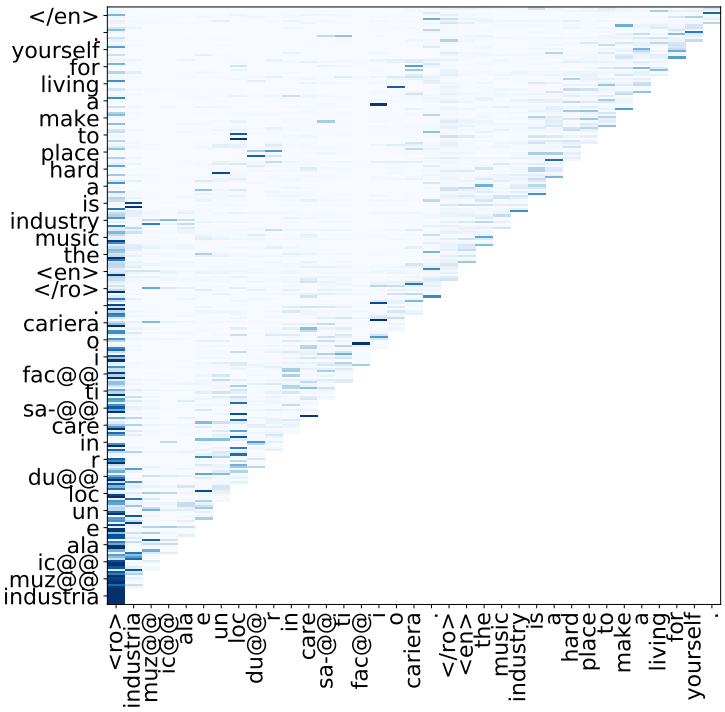

Figure 7.5: *Attention weights in the **6th layer**. **Some cross sentence attention structure starts to emerge.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

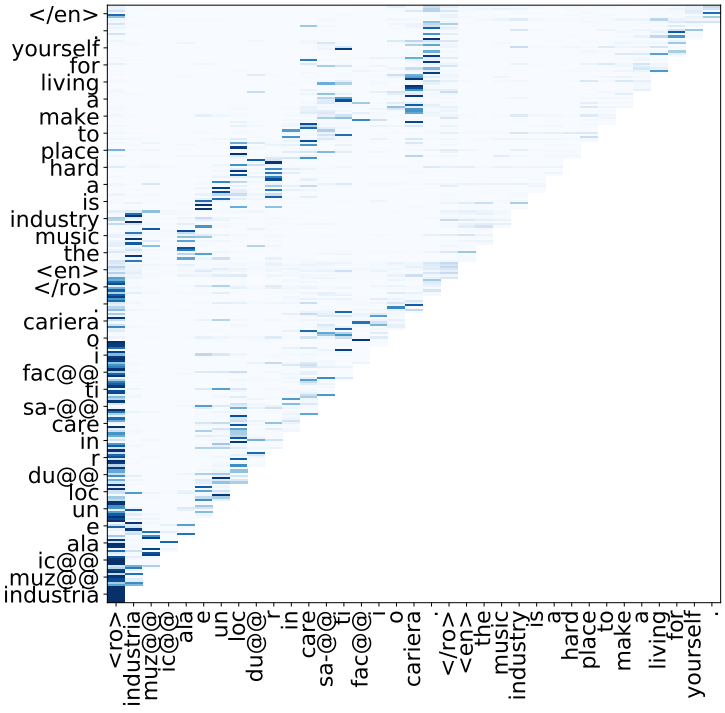

Figure 7.6: *Attention weights in the **9th layer**. **Many cross sentence attentions can be observed.** In this layer, for this example, we can clearly see the model focuses on the position of word "loc" for the prediction at the position of the word "place" in English on the target side. **This is representative of all top layers i.e. from 7th to 12th.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.*

7.3 Summary

RNN and Transformer language models allows handling variable length contexts effortlessly. It gives the possibility to evaluate and train these models on different sequence length spans, constructed by concatenating multiple sentences. In this chapter, we systematically evaluated the impact of different training spans on different evaluation spans. We observed very large variations in terms of perplexity, and demonstrated the importance of such information when reporting perplexities. This analysis had two by-products. First, as we carried out this analysis for both LSTM and Transformer models, we could thus compare the robustness property of both models when training and evaluation differ a lot in terms of sequence lengths. Generally, we found LSTM models to be more robust. Finally, we obtained improved training methods for cross-sentence speech recognition.

We also investigated translation language models; by training language models on the concatenation of source and target sentences of a translation dataset. While its performance was behind that of the standard encoder decoder translation models, we obtained an interesting visualization results, which confirmed the layer types in Transformer language models we had found in Chapter 4, Sec. 4.2. In particular, we presented a more explicit demonstration that attention on specific patterns is only carried out by the top layers.

8. Scientific Achievements

The goal of this thesis was to analyze and advance state-of-the-art neural language modeling in automatic speech recognition, with a special focus on studying and exploiting unique opportunities and challenges which are offered by modeling based on neural networks. While pursuing such a goal, following contributions have been obtained:

Validation of Transformer language models in automatic speech recognition. We validated the applicability, and established the state-of-the-art performance of Transformer language models in automatic speech recognition. On several datasets, we obtained more than 10% relative improvements in perplexity over well tuned baseline LSTM language models, which resulted in up to 10% relative improvements in word error rate.

In particular, we have shown very deep models to be trainable without any extra loss or pretraining, and the trained model to perform well, which demonstrated the power of composing rather weak operations (weighted averaging) multiple times. This was a contrast with our earlier attempts and failures in working with only one layer of attention, which we presented in the preliminary. Previous works had claimed that training such deep Transformer models would require some extra loss in the intermediate layers.

We also proposed modifications in the design of Transformer layers which gave competitive performance to the original model, while drastically reducing the memory requirement, which was a crucial aspect when applying Transformer language models in speech recognition.

Finally, we revealed the internal organization of hidden layers in deep Transformer language models. We identified 4 main layer types (input, blurring, windowing, and structured layers) and linked them to the corresponding operations in natural language processing. No previous work had demonstrated these properties of Transformer language models, using the visualization in the form we provided.

Practical applications of knowledge distillation in ASR language modeling. Despite the popularity of knowledge distillation in general, there has been no application to language modeling prior to our work. This was potentially the case because of the large vocabulary used in language modeling for ASR which makes the straightforward application computationally expensive. We proposed practically useful distillation losses for the three most common output types in large vocabulary neural language modeling: sampled softmax, noise contrastive estimation, and class based factorized output, and we experimentally demonstrated their effectiveness.

We made use of distillation to investigate potential improvements in neural language models which have some convenient structure for decoding, by transferring performance from more powerful models. We investigated distillation from Transformers to LSTM-RNNs, and from LSTM-RNNs to n-gram feed-forward models. In both cases, we found that the structural difference in the model is a hard barrier to completely overcome, while distillation helped reducing this gap.

Also, this step of investigation on knowledge distillation became a crucial component for implementing one of our solutions for domain robust language modeling.

Introduction and solutions for domain robust language modeling with neural networks. We introduced domain robust neural language modeling and proposed two solutions. Domain robustness problem in neural language modeling has not been addressed by any previous work. Our first approach was based on a large adaptive mixture of expert model, following the trend of building "outrageously" large neural networks at the time of this work. The recurrent adaptive mixture model we introduced in this context, was later extended and applied in acoustic modeling in speech recognition by Microsoft [Das & Li$^+$ 19].

In the second approach, we addressed the practical drawback of the first approach which was the model size, by introducing knowledge distillation from pre-trained domain experts. We successfully transferred performance of separate domain experts into a single compact model.

These experiments have been conducted using the real data at large industry scale, provided by Google and AppTek.

Improved training for cross sentence language modeling. We proposed a training strategy for neural language model to perform well for cross-sentence evaluation, which improved both cross-sentence perplexity and word error rate, compared with the naive sentence-wise training.

In addition, we conducted a systematic study of the impact of different training sequence construction methods under different evaluation conditions, which demonstrated the importance of reporting these pieces of information for comparing perplexities. No previous work had reported such a comprehensive comparison. In many publications, such information is ignored.

We conducted the corresponding robustness analysis for both LSTM and Transformer based models. We demonstrated that LSTM-RNN models are much more robust than Transformer language models which are based on attention. No previous work had provided experimental evidence of such a property.

Improved state-of-the-art performance in multiple standard tasks by a large margin. We achieved state-of-the-art performance, on LibriSpeech, TED-LIUM 2, AMI, and Swichboard 300h, at multiple stages of the work presented in this thesis. While these are results of strong team efforts, the language models developed in this thesis played a crucial role.

Not only the new Transformer models, but also our LSTM baselines were shown to be well tuned. In particular on LibriSpeech, we drastically improved performance of both hybrid NN-HMM and end-to-end speech recognition approaches.

We believe in the impact of these works on advancing the field by repeatedly reducing the baseline error rate. For example, our work triggered many follow-up publications on the LibriSpeech dataset with good word error rates, which made use of our baseline LSTM language modeling recipe [Han & Prieto$^+$ 19, Karita & Chen$^+$ 19].

We also contributed to the research community by making the corresponding language modeling recipes, as well as a number of pre-trained models, publicly available online.

Model contributions to other PhD students in the group. Finally, a side contribution of this thesis is that the language models trained for this thesis were used in many other publications by other PhD students in the team.

9. Individual Contributions

Many of the results presented in this thesis are based on the previous publications. According to §5.6 of the doctoral guidelines of the RWTH Aachen University, Faculty of Mathematics, Computer Science and Natural Sciences, September 7, 2018, in this chapter, we provide a list of referenced publications of the author of this thesis, describing his contributions. We note that, for all papers in which Kazuki Irie is the first author, he was the main author who wrote the paper, typically with feedback from his co-authors, without indicating so for each paper. We split the list of publications in two groups as follows:

First set of publications. Experimental results from the following papers have been reported in this thesis:

- [Irie & Schlüter$^+$ 15]: Kazuki Irie designed the weighted bag-of-words models and the experimental setups. He implemented the corresponding model and conducted experiments. The results from this paper were reported in Chapter 3.

- [Irie & Tüske$^+$ 16] : Kazuki Irie proposed the new language models which make use of highway connections and attention. He designed the experimental setups and conducted experiments. The results from this paper were reported in Chapter 3.

- [Irie & Lei$^+$ 18b]: Kazuki Irie proposed to improve n-gram feed-forward language models using LSTM teacher models in the motivation of teaching n-gram models to recover truncated contexts. He designed the experiments, and supervised Master student Zhihong Lei to implement and carry out model training. Kazuki Irie implemented convolutional neural network based models and conducted corresponding experiments. He conducted all ASR rescoring experiments. The results from this paper were reported in Chapter 5. This work received two awards at IEEE ICASSP 2018: the best student paper award and IEEE spoken language processing student travel grant.

- [Irie & Kumar$^+$ 18]: Kazuki Irie proposed the recurrent adaptive mixture model. He implemented the model, designed and carried out experiments during his internship at Google, NY, USA. He conducted the analysis of the new model by visualization. The results from this paper were reported in Chapter 6.

- [Irie & Zeyer$^+$ 19a]: Kazuki Irie proposed to make Transformer based models much deeper than any published results at the time of the work. He designed the experimental setups, conducted experiments, and carried out analysis of the model and interpreted the internal structure of the Transformer model. He implemented lattice rescoring for the new Transformer language models. The results from this paper were reported in Chapter 4. This work received the ISCA best student paper award at Interspeech 2019.

- [Lüscher & Beck+ 19]: In this system paper for LibriSpeech, Kazuki Irie conducted rescoring experiments with neural language models. His work contributed to establish a new state-of-the-art results on the LibriSpeech dataset. He wrote the corresponding part of the paper. The results from this paper were reported in Chapter 4.

- [Irie & Zeyer+ 19b]: Kazuki Irie proposed studying the sequence length robustness of LSTM and Transformer language models for improving cross-utterance speech recognition. He designed the experimental setups and conducted experiments. The results from this paper were reported in Chapter 7.

- [Gerstenberger & Irie+ 20]: Kazuki Irie proposed the use of knowledge distillation for domain robust language modeling, as a follow-up to his own previous work above [Irie & Kumar+ 18]. He designed the experimental setups, and supervised Bachelor student Alexander Gerstenberger to implement and carry out the experiments. He significantly contributed in writing the paper. The results from this paper were reported in Chapter 6, Sec. 6.2 as well as in Alexander Gerstenberger's Bachelor thesis [Gerstenberger 20].

- [Irie & Gerstenberger+ 20]: Kazuki Irie analyzed and proposed the reorganization of the Transformer layer for memory efficiency. He implemented the corresponding modifications, designed and conducted the experiments. The results from this paper were reported in Chapters 4 and 5.

- [Zhou & Michel+ 20]: In this system paper for TED-LIUM 2, Kazuki Irie conducted rescoring experiments with neural language models. His work contributed to establish a new state-of-the-art results on the TED-LIUM 2 dataset. He wrote the corresponding part of the paper. The results from this paper were reported in Chapter 4.

Second set of publications. We referred to the following papers in this thesis, without reporting experimental results:

- [Botros & Irie+ 15]: Kazuki Irie proposed the experimental designs, and supervised Master student Rami Botros. He trained the models, designed and conducted rescoring experiments for keyword search. He significantly contributed in writing the paper.

- [Tüske & Irie+ 16]: Kazuki Irie discussed the idea, shared experimental setups, and carried out rescoring experiments.

- [Menne & Heymann+ 16]: In this system paper for RWTH Aachen's CHiME-4 evaluation, Kazuki Irie trained an LSTM highway language model, provided the rescoring scripts, and wrote the corresponding part of the paper.

- [Schlüter & Doetsch+ 16]: In this review paper, Kazuki Irie contributed in writing the part on neural network based language modeling.

- [Irie & Golik+ 17]: Kazuki Irie proposed to apply the previously published character-aware language modeling approach to byte-level language modeling in low resource speech recognition for keyword search. He designed and conducted experiments.

- [Golik & Tüske+ 17]: In this system paper for RWTH Aachen's IARPA BABEL evaluation, Kazuki Irie set up and trained LSTM language models, and conducted the rescoring experiments.

- [Zeyer & Irie$^+$ 18]: Kazuki Irie set up and trained BPE level LSTM language models. He implemented shallow fusion and conducted the corresponding experiments. He wrote the corresponding part of the paper.
- [Irie & Lei$^+$ 18a]: Kazuki Irie proposed the approach for making use of completion model (a type of models commonly called masked language models or bi-directional language model) in automatic speech recognition. He designed the experimental setups, and supervised Master students Zhihong Lei and Liuhui Deng to implement and conduct experiments.
- [Irie & Prabhavalkar$^+$ 19b]: Kazuki Irie proposed and designed model details and experimental set-ups. He implemented and conducted experiments during his internship at Google, CA, USA. He built the state-of-the-art baseline system for end-to-end speech recognition (on LibriSpeech) as of the time of the work.
- [Zeyer & Bahar$^+$ 19]: Kazuki Irie trained Transformer and LSTM language models. He conducted decoding experiments with the language models. He wrote the corresponding part of the paper.

10. Outlook

Continuous learning language models. One aspect which is clearly missing in the current state of language modeling, and in particular in Chapter 6 of this work on domain robust language modeling, is a mechanism to integrate new data to an already trained model. In Chapter 6, we proposed methods to train one domain robust model from a set of multiple training texts from different domains. But what if new data in a new domain become available afterwards? How can we update the model, while still preserving the robustness of the model? More generally, we can imagine obtaining new data every day, for example from news, and we would want to keep updating the language model. In the current state-of-the-art language modeling setups, we would need to train the whole model from scratch every time we obtain extra data. Similarly, with the domain robust language modeling techniques which we investigated in this thesis, some model component would need to be re-trained on the whole dataset again. Investigating effective methods for such scenarios is necessary.

Another related property which is currently missing to neural language models, is the ability to extend its vocabulary. Some character or sub-word level "open vocabulary" models can appear as a solution for that problem, but in reality, their vocabulary is limited to those which can be produced from the combination of their pre-determined characters or sub-words. We could imagine a flexible language model which can extend its character level vocabulary, for example by learning first one language in one set of characters, and then learn a second language with another set of characters, while ideally getting benefits of having learned the first language to better learn the second one. Investigating an algorithmic solution for handling vocabulary extension and learning can be interesting and useful, for word level as well as for character level models.

Also, these discussions assume that the model architecture is fixed (for example an LSTM). On a higher level, we also might want to design continuous learning approaches which are robust to the change in the base model structure. For example, we experienced a model architecture shift from the LSTM to the Transformer in the course of this thesis.

With the increasing interest in continuous learning, we believe language modeling to be a good application for testing new algorithms.

Systematic progress in ASR language modeling via better hardware? The general philosophy of *training bigger models with some good regularization* illustrated in the preliminary chapter 3 should potentially give improvements for many tasks without having to develop new methods. As we noted, our models for the LibriSpeech dataset were not much overfitting yet. We should get improvements by simply increasing the model size further. More compute might soon make LibriSpeech (850 M running words in training) tomorrow's Penn Treebank (1 M running words in training), if that is not already the case for those who can afford it.

Masked language modeling in ASR? Non-autoregressive language models, such as BERT [Devlin & Chang+ 19] have seen lots of success in many applications for natural language processing. While we explicitly excluded this type of *language models* in the introduction from the scope of this thesis, it is also interesting to consider its potential application to ASR in the future. The direct application of such a model as an ASR language model is not straightforward. But we could try to integrate such a model into the computation of the sentence probability via decomposition such as the one we investigated in [Irie & Lei+ 18a] (where unfortunately this exact decomposition was not investigated, in favor of another one with strong simplifications), for any k $\in \{2, ..., N-2\}$

$$
\begin{aligned}
p(w_0^N) &= p(w_0^{k-1}) \cdot p(w_k, w_{k+1}^N | w_0^{k-1}) \\
&= \underbrace{p(w_0^{k-1})}_{\text{Forward LM}} \cdot \underbrace{p(w_k | w_0^{k-1}, w_N^{k+1})}_{\text{Completion Model}} \cdot \underbrace{p(w_N^{k+1} | w_0^{k-1})}_{\text{Prefix conditioned backward LM}}
\end{aligned}
$$

where the *completion model* in the middle, can be parametrized by a masked language model such as BERT.

(No) Need for language models in ASR in the future? With the advances in the end-to-end speech recognition, it might be the case that the *external* language model would not have anymore impact on the systems' performance, similar to what is experienced in neural machine translation. While we can still think of some adaptation method which makes use of some language models to adapt the main end-to-end speech recognition models, such an approach rather seems to be a workaround. Ideally, such an ability to adapt to the domain should be part of the ASR system's internal language model. Even if the amounts of text-only data continue to largely exceed the amount of transcribed audio, if both quantities continue increasing, we speculate that the relevance of the transcribed data would outweigh the benefit of a large amount of text-only data.

In contrast, it is interesting to observe increasing interests in language modeling in general. While still limited, large scale Transformer language models such as OpenAI's GPT-2 have shown a potential for new applications of language models.

Better not to work on language modeling to improve language modeling? As we saw in Chapter 4, the largest progress in language modeling was obtained *simply* by applying the Transformer architecture. It is clear that future progress in general sequence modeling would improve language modeling for speech recognition, and that investigating further progress in generic sequence modeling is interesting, but it is unclear whether aiming at improving language modeling is the best way to improve language modeling. Transformers gave up the elegant property of RNNs to compress the context into a fixed size vector, which was maybe more acceptable or natural from the view point of machine translation than from that of language modeling.

A. Overview of the Corpora and Systems

This appendix summarizes the corpora and systems which were used in the experiments presented in this thesis. That includes the data description of language modeling, as well as some descriptions of the baseline automatic speech recognition systems.

We note that for each of Quaero and Switchboard task, we need to introduce two ASR systems which are used in experiments in this thesis: the *preliminary system* and *baseline system*. The corresponding preliminary system is only used for some preliminary or analysis related experiments. For the core experiments demonstrating state-of-the-art language modeling performance, we made use of the baseline system in all cases. For all other datasets, a single baseline NN-HMM hybrid ASR system is used all along the thesis.

A.1 LibriSpeech

Language modeling setups. The LibriSpeech datasets [Panayotov & Chen$^+$ 15] for language modeling consists of 800M-word text only data and 960 hours of audio transcriptions which corresponds to 10M-word text data. Based on analysis of count model perplexities, we observe that the audio transcription part does not contain special domain signal which matches the development set. Therefore, we simply merge the two datasets to form a single dataset for language model training. The average sentence length in the resulting training data is 21 words with the maximum length of 600 words. The development and test sets respectively have two parts (according to [Panayotov & Chen$^+$ 15]): dev-clean, dev-other, test-clean, and test-other. This separation is based on the audio-level characteristics, therefore it has no special meaning for language modeling. In the experimental section, we denote by "**Dev**" and "**Test**" the concatenation of *clean* and *other* parts of the respective data. Both datasets consist of about 110K running words with average of 20 words per sentence. The word-level vocabulary contains 200K words. The out-of-vocabulary rates for the dev-clean, dev-other, test-clean, and test-other subsets are 0.3%, 0.5%, 0.4%, and 0.5% respectively. We report all perplexities without making use of contexts beyond the sentence boundary. We use the official baseline 4-gram Kneser-Ney language model provided with the dataset.

The neural language models used for the performance overview presented in Tables 4.16 and 4.17 are as follows. For the word-level experiments, the 96-layer Transformer model from Table 4.4 and the 2-layer 4096-dimension LSTM model from Table 3.3 (also reported in Table 4.4) were used. The numbers for the BPE-level experiment were directly taken from Table 4.9.

Baseline NN-HMM ASR system. The acoustic training consists of 960 hours of read book transcriptions. The system is based on the hybrid NN-HMM. The acoustic model is based on multi-layer bi-directional LSTM [Zeyer & Doetsch$^+$ 17]. Discriminative training with the minimum phone error criterion [Povey & Woodland 02] is used. The system is speaker independent.

Further descriptions of our baseline acoustic model, we refer the readers to the dedicated system paper [Lüscher & Beck+ 19].

Baseline encoder-decoder attention ASR system. For LibriSpeech, we also carried out experiments with end-to-end speech recognition with the encoder decoder attention based model. Our system is based on the standard Listen, Attend, and Spell system [Chan & Jaitly+ 16, Zeyer & Irie+ 18]. We refer to the corresponding system paper [Lüscher & Beck+ 19] for further details.

A.2 TED-LIUM Release 2

Language modeling setups. The language model training data provided by TED-LIUM release 2 consists of **7 subsets** including the TED-LIUM 2 audio transcriptions [Rousseau & Deléglise+ 14]. The total amount of resulting training data for language modeling consists of 270 M running words, of which 2 M are from the audio transcriptions of the acoustic training data. We use the word level vocabulary size of 152 K words. The out-of-vocabulary rates for the development and evaluation texts are 0.0% in both cases. We first train n-gram Kneser-Ney language models on each subset of the training data with the discount parameters optimized on the dev set [Sundermeyer & Schlüter+ 11]. We linearly interpolate these sub-language models using the interpolation weights optimized for the dev perplexity; we include a background n-gram model as the 8th component in interpolation, which is trained on all training texts (which gave 5% rel. improvements on the 4-gram before pruning).

The upper block of Table A.1 shows perplexities for the count models. First of all, we observe large improvements in development perplexity by increasing the order from 4 to 6 (in contrast to what we typically observe e.g., on LibriSpeech). This was in fact due to some **overlap** between the *common crawl* training subset (16 M words) and the development text in the original dataset.

The overall effect of this problem seems to be marginal after pruning. We apply pruning to obtain a reasonably sized model for the first pass decoding. Once the pruning is applied, the improvements from these higher-order n-grams disappear, as shown by perplexities for different orders n in Table A.1. Almost no improvement is obtained by going beyond 4-gram (as is typically the case for a clean dataset. As a side note, we note that to avoid the well known negative effect of entropy pruning on Kneser-Ney language models [Chelba & Brants+ 10], we trained separate Katz $(n-1)$-gram language models [Chen & Goodman 99] to help the pruning process. However, at this pruning ratio (at most factor of 6), the benefit of such extra care was marginal.

Table A.1: ***Perplexity*** *of the word-level (152K vocab)* ***baseline*** *models on* ***TED-LIUM 2****.*

Model	Params. in M	Perplexity	
		Dev	Test
4-gram	343	105.4	124.7
+ pruning	161	113.2	127.9
5-gram	663	92.3	123.2
+ pruning	169	112.4	127.8
6-gram	1021	86.2	121.3
+ pruning	183	116.2	125.9

The neural language models used for the performance overview presented in Tables 4.16 and 4.17 are as follows: The 32-layer Transformer model (from Table 4.11) was used. For the LSTM, the 4-layer 2048-dimension model from the same Table 4.11 was used.

Baseline NN-HMM hybrid ASR system. In TED-LIUM release 2, 207 hours of audio transcriptions are available for acoustic model training. The baseline hybrid NN-HMM system is similar to the one for the LibriSpeech dataset (Sec. A.1). In addition, the data augmentation [Park & Chan$^+$ 19] is applied. For further details, we refer the interested readers to the dedicated system paper [Zhou & Michel$^+$ 20].

A.3 Quaero English

This section describes the English broadcast news and conversation speech recognition task from the Quaero project [Nußbaum-Thom & Wiesler$^+$ 10][1]. While the project had already ended when this thesis has started, this dataset has been used as a reference dataset in our team.

Language modeling setups. The baseline n-gram count models are the same as in [Sundermeyer & Ney$^+$ 15]: 4-gram Kneser-Ney model was trained on the total of 3.1 B of running words, with a vocabulary size of 150 K. The 3.1B data was composed of 11 sub-corpora. Language models were trained on each of the sub-corpus and combined into a single model. The interpolation weights were optimized on the development text using the SRILM toolkit [Stolcke 02]. The development and evaluation texts contain 40 K and 36 K running words, respectively. For further details, we refer to [Tüske & Irie$^+$ 16].

All neural language models are trained on 50 M running words. The 50 M data are the in-domain subsets of the full 3.1 B data. The resulting lexicon size for neural models is 128 K. A renormalization is therefore done for interpolation with the count model [Sundermeyer & Oparin$^+$ 13]. Again, this setup is the same as in [Sundermeyer & Ney$^+$ 15]. The out-of-vocabulary rates for the development and evaluation sets are 0.4% and 0.5% respectively, for both 150 K and 128 K vocabularies.

The neural language models used for the performance overview presented in Table 4.16 are as follows. The sentence-level LSTM with 2 layers of dimension 2048 from Table 7.3 and the sentence-level 32-layer Transformer model from Table 7.7 were used.

Preliminary ASR system. The preliminary system is an outdated system only used in the preliminary Chapter 3. This is the same system used in [Sundermeyer & Ney$^+$ 15]. The acoustic training data consisted of 250 hours of transcribed data in English. The acoustic model is a Gaussian mixture model in the tandem approach [Hermansky & Ellis$^+$ 00]. It uses multilingual bottleneck multi-layer perceptron features [Tüske & Schlüter$^+$ 13] trained on 840 hours of 4 languages (English, French, German, and Polish) data. More details of the system can be found in [Sundermeyer & Ney$^+$ 15].

Baseline NN-HMM ASR system. The acoustic training data consisted of 250 hours of transcribed data in English. The baseline hybrid NN-HMM system is similar to the one for LibriSpeech (Sec. A.1).

A.4 Switchboard 300 h

Language modeling setups. We carry out experiments on different subsets of the Switchboard speech recognition dataset: the statistics are shown in Table A.2. The cross validation (CV) set was prepared by randomly choosing sentences from the original Switchboard (3 M) and Fisher (24 M) transcriptions, resulting in 133 K words (counting sentence end tokens). The rest of the

[1] Quaero Program: http://www.quaero.org

transcriptions, which amounts to 26.7 M running words, are used as training data for all language models: both the 4-gram Kneser-Ney count model and neural language models. This selection is the same as in [Tüske & Michel$^+$ 17]. A vocabulary size of 30 K is used. The cross validation set was used for Newbob tuning of the learning rate during neural language model training and for selecting the interpolation weight for combining the count models trained on the Switchboard and Fisher parts of the data. The Hub5_00 set is used to tune the LM scale for the recognition experiments. Following the common practice, we report numbers based on *Switchboard* (SWB) and *Call-Home* (CH) partitioning of Hub5_00.

For the performance overview presented in Tables 4.16 and 4.17, The 32-layer Transformer model (from Table 7.6) and the 2-layer 2048-dimension LSTM model from Table 7.2 were used.

Table A.2: *Number of running words, OOV rates and average sentence lengths in terms of number of words (Avg. length) of all data sets and subsets used. The vocabulary size is 30 K.*

		# Words	OOV[%]	Avg. length
Train		26.7M	1.6	11.2
Cross Validation		133K	0	12.8
Hub5_00	Total	45K	1.1	10.4
	CH	23K	1.6	9.1
	SWB	22K	0.7	12.3
Hub5e_01		65K	1.0	11.4

Preliminary ASR system. The acoustic modeling makes use of 300 hours of transcribed data for training. This preliminary hybrid NN-HMM system was only used in Chapter 5, Sec. 5.2.2. The system is similar to the one for LibriSpeech (A.1) and this is the same one used in [Tüske & Michel$^+$ 17]. Among the system presented in [Tüske & Michel$^+$ 17], we made use of the acoustic model based on a 5-layer bidirectional LSTM-RNN.

Baseline ASR system. The baseline hybrid NN-HMM system is similar to the one for LibriSpeech (A.1), except that i-vector and affine transformation speaker adaptations were applied [Kitza & Golik$^+$ 19]. The main system is similar to the preliminary system above. Further details of the system is given in [Kitza & Golik$^+$ 19].

A.5 AMI

Language modeling setups. The AMI transcriptions results in about 850 K running words for training language models. In addition, we include the whole 27 M Switchboard and Fisher dataset for training the language models. The word level vocabulary size is 48 K. The out-of-vocabulary rates for the development and evaluation sets are 0.4% and 0.8% respectively.

The baseline Kneser-Ney language model is obtained by interpolating the models separately trained on the 3 corpora (AMI, Switchboard, Fisher) and the background (one text including all three), similar to as is done for TED-LIUM (A.2). The effect of domain adaptation is illustrated in Chapter 3, Sec. 3.1.4.

In the performance overview presented in Tables 4.16, we directly reported the performance of the LSTM and Transformer models after interpolation with the 3-gram count models. The standalone perplexities of these models are presented in Table A.3. The LSTM model has 2 layers with dimension 2048 with a bottleneck layer before the softmax layer of dimension 512. 40%

dropout is applied. The Transformer model has 16 layers, a feed-forward dimension of 2048, a residual dimension of 512, and 8 heads. 20% dropout is applied. No positional encoding is used.

Table A.3: *Standalone perplexities of the 48K vocabulary word-level* ***baseline*** *models on* ***AMI****. Perplexities after fine-tuning on the AMI transcriptions.*

Model	#Param. [M]	Perplexity	
		Dev	Test
3-gram	10	88.9	93.5
4-gram	30	87.3	91.7
LSTM	83	57.3	60.2
Transformer	82	**56.0**	**59.0**

Baseline NN-HMM hybrid ASR system. The AMI meeting corpus [McCowan & Carletta$^+$ 05] contains transcriptions of about 100 hours of meeting recordings. The corresponding data is split into three subsets consisting of one set of 78 hours for training acoustic models and two sets of 9 hours each for development and evaluation sets. In a typical setup, there are two types of segmentation depending on whether the utterances are segmented according to punctuation marks or not. We found the case without punctuation based split to give better word error rates [Vieting 19]. All AMI models (including language models as we mentioned in Sec. 3.1.3) used in this thesis are therefore trained on data without punctuation based split. The baseline hybrid NN-HMM system is similar to the one for LibriSpeech (Sec. A.1). We thank Peter Vieting for having shared his system prior to publication [Vieting 19].

A.6 Google YouTube Dataset

Language modeling training data. This dataset is only used in Chapter 6, Sec. 6.1 in this thesis. As the discussion of the data itself is of interest in the corresponding chapter studying domain robust language modeling, the description of the data is directly provided in the corresponding section.

Baseline NN-HMM ASR System. Kazuki Irie used the lattices generated by the system developed by Google presented in [Soltau & Liao$^+$ 17] (the phone-level variant) during his internship at Google, NY, USA. The baseline system is rather similar to the one for LibriSpeech (Sec. A.1), except the use of the CTC loss [Graves & Fernández$^+$ 06] for training. The model was trained on 125,000 hours of semi-supervised acoustic training data. We refer to [Soltau & Liao$^+$ 17] for further details.

A.7 AppTek Multi-Domain Dataset

Language modeling training data. Similar to above, this dataset is only used in Chapter 6, Sec. 6.2 in this thesis. As the discussion of the data itself is of interest in the corresponding chapter studying domain robust language modeling, the description of the data is directly provided in the corresponding section.

Baseline NN-HMM ASR System. The baseline system was made available by Pavel Golik at AppTek. The system is a hybrid NN-HMM similar to the one for LibriSpeech (A.1). The acoustic

model was trained on a very large collection of various recordings from the broadcast news and media as well as entertainment domains.

A.8 WMT 2016 Romanian to English

The training data consists of 600K parallel sentences. Both development (`newsdev2016`) and evaluation (`newstest2016`) data sets contain 1999 sentences.

We tokenize the sentences based on the joint BPE subword units obtained on the mix of source and target texts. The translation language model vocabulary is build based on the resulting BPE units on the mix of source and target training texts. The resulting BPE level vocabulary size is 20K. The number of BPE tokens in the training, development, and evaluation texts are respectively 38 M, 129K and 137K. We use the baseline translation system used in [Nix & Kim$^+$ 19]. We thank Arne Nix for sharing his baseline as well as the prepared translation datasets.

B. More On the Role of Count Language Models

Today's recipe for neural language modeling (either LSTM or Transformer) is well established enough to always allow us to obtain large improvements (at least 20% in terms of perplexity as a rule of thumb) over the 4-gram count based language model trained on the same amount of data, independent of the amount of data.

Conversely, if such is not the case, we can suspect either some sub-optimality in the tuning of the neural language model (which becomes rather rare once we become used to tuning) or some **overlap between the training and evaluation data**. This is because count language models tend to strongly overfit to the training data. Therefore, when there is an overlap between the training and evaluation data, an n-gram count model (with n higher than 4) suddenly becomes a model which is very hard to beat with neural networks.

The overlapping problem can be in fact detected at a much earlier stage of language model preparation, independent of neural language models. If we obtain a large improvement in terms of perplexity by increasing the order n of n-gram count models to values higher than 4, we should start suspecting the problem. It is a sign of a good match between the training and evaluation data. During the development of this thesis, we had this case twice[1]: with an initial version of the AppTek dataset (the actual dataset used in Sec. 6.2 was obtained after detecting and cleaning up the original dataset) and with the TED-LIUM official dataset as we pointed out in Sec. 4.3.

From this perspective, it is useful to carry out a quick check on perplexities of count language models (which can be obtained normally much faster than those of neural models).

Also, since there is no universal reference number for a good value for the perplexity for a given task (while very bad numbers are immediately clear), count based language models can provide a **reference perplexity** number when tuning neural language models, in order to avoid accidents in hyper-parameter tuning.

These are practical roles of count based language models in the process of building current state-of-the-art neural language models in automatic speech recognition, in addition to its crucial role in **identifying domain signals** as we have seen all along Chapter 6.

[1] We thank Alexander Gerstenberger and Pavel Golik for help with finding out and resolving these problems.

List of Figures

1.1 *Scheme for a conventional HMM based statistical speech recognition system based on Bayes decision rule [Bahl & Jelinek+ 83]. The dashed arrows indicate the second pass lattice rescoring with a second language model (1.2.2), as is done with neural language models in this thesis.* 7

1.2 Pseudo-code adapted from [Sundermeyer 16] for Sundermeyer's push-forward lattice rescoring algorithm. 10

1.3 *Listen Attend and Spell.* Figure taken from [Irie & Prabhavalkar+ 19b]. 12

3.1 *Model of type: Attention after the recurrent layer. No trigger is obtained, the model chooses the most recent context from the GRU. Quaero development perplexity of 109.1, which is similar to 110.6 of the model without the attention layer.* 32

3.2 *Model of type: Attention before the recurrent layer. Some triggers can be observed, but the perplexity is bad: 157.6 which is close to the perplexity of 4-gram model, 163.0.* 32

3.3 *Two example of attention weights from the model in Figure 3.2. For each sentence, the word inside a box is the target word. The numbers in exponent of the context words are the scores in percentage given by the model to predict the target word. Words with the highest weights (triggers) are highlighted with* ***bold*** *font. $ denotes sentence begin token.* 33

3.4 *Correlation between perplexity and word error rate using the preliminary ASR system for* ***Quaero*** *(A.3). Both axes are on the natural log scale. The regression has the equation:* $log(WER) = 0.62 + 0.39 * log(PPL)$. 35

3.5 *Correlation between perplexity and word error rate using the hybrid NN-HMM ASR system for* ***LibriSpeech*** *on the dev-clean subset (A.1). Both axes are on the natural log scale. The regression has the equation:* $log(WER) = -0.79 + 0.40 * log(PPL)$. . 35

3.6 Correlation between perplexity and word error rate; using the preliminary ASR system for Quaero (A.3). Both axes are on the natural log scale. The regression has the equation: $\log(\text{WER}) = 1.34 + 0.20 * \log(\text{PPL})$. 37

3.7 Correlation between perplexity and word error rate for TED-LIUM 2 (A.2) using the 4-gram count language model. Both axes are on the natural log scale. The regression has the equation: $\log(\text{WER}) = 1.76 + 0.25 * \log(\text{PPL})$. 37

4.1 Illustration for Transformer language model components. 40

4.2 *Attention weights in the* ***first layer*** *for the model* ***with*** *positional encoding. The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 51

4.3 *Attention weights in the **first layer** for the model **without** positional encoding. The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 51

4.4 *Attention weights in the second layer representing the "blur" **bottom layers (2-3)** for the model without positional encoding. **These layers seem to carry out averaging over all positions, thus collecting global information. Some heads focus on difficult words, here "verandah".** The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 52

4.5 *Attention weights in the 5th layer representing the "window" **mid layers (4-9)** for the model without positional encoding. **These layers focus on the local n-gram.** The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 53

4.6 *Attention weights in 24th layer representing the "structured" **top layers (10-24)** for the model without positional encoding. **It seems to be some feature detector attending to some specific patterns.** The x-axis corresponds to the input words. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 53

4.7 Illustration for the standard Transformer layer. 55

4.8 Illustration for the modified Transformer layer. 55

5.1 *Effect of the teacher weight λ in Eq.* (5.4) *on the Switchboard cross validation set.* 68

6.1 *Recurrent adaptive mixture model (RADMM) based neural language model.* 75

6.2 *Example 1: Category **News & Politics**. The x-axis corresponds to the input words. The y-axis shows the expert domains.* 80

6.3 *Example 2: Category **News & Politics**. The x-axis corresponds to the input words. The y-axis shows the expert domains.* 80

6.4 *Example 3: Category **Howto & Style**. The x-axis corresponds to the input words. The y-axis shows the expert domains.* 81

6.5 *Example 4: Category **Gadgets & Games**. The x-axis corresponds to the input words. The y-axis shows the expert domains.* 81

7.1 *Training sequence variants for LSTM-RNN models. Lines represent sentences. Circles represent RNN states at sentence boundaries (empty circles for zero states and filled circles for non-zero states). Dashed arrows (blue) represent state copying (context carry-over). Solid arrows (red) represent back-propagation.* (a.) ***Sentence-wise** training* (b.) ***Sentence-wise CCO*** (c.) ***Concatenated** sentences* (d.) ***Concatenated CCO**. One this same figure, we can also visualize the two evaluation modes: (a.) sentence-wise evaluation and (b.) full context evaluation.* 92

7.2 *Attention weights in the **first layer**. The x-axis corresponds to the input tokens. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 103

7.3 *Attention weights in the **3rd layer**. **Attention is rather intra-sentence and blur.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 104

7.4 *Attention weights in the **5th layer**. **Attention is rather intra-sentence in layers in this region, with some focus on the n-gram (somehow skipping the latest token).** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 104
7.5 *Attention weights in the **6th layer**. **Some cross sentence attention structure starts to emerge.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 105
7.6 *Attention weights in the **9th layer**. **Many cross sentence attentions can be observed.** In this layer, for this example, we can clearly see the model focuses on the position of word "loc" for the prediction at the position of the word "place" in English on the target side. **This is representative of all top layers i.e. from 7th to 12th.** The x-axis corresponds to the input token. The y-axis shows the target words, where for each target word position, 8 attention heads are shown vertically.* 105

List of Tables

3.1 *Perplexities on Quaero English development data for standalone LSTM and GRU. The perplexities for 1- and 2-layer LSTMs are taken from [Sundermeyer & Ney$^+$ 15]. Exceptionally here, in order to be consistent with [Sundermeyer & Ney$^+$ 15] the perplexities are evaluated by concatenating evaluation sentences into sequences in the original order such that each sequence contains at most 100 words.* 19

3.2 *Large and regularized models work well. Perplexities of 2-layer LSTM language model on Quaero English. The baseline 600-unit model architecture corresponds to the best model at the time of [Sundermeyer & Ney$^+$ 15] (re-trained on the sentence level for a fairer comparison, instead of directly using the model from [Sundermeyer & Ney$^+$ 15] trained on the concatenated sentences, as we report perplexities on the sentence level here).* 20

3.3 *Perplexities of LSTM language models on LibriSpeech. Illustrating model tuning on a large dataset.* 21

3.4 *Perplexities of LSTM language models on AMI. Effect of training consistent with evaluation segmentation (split after punctuation). The development and evaluation sets are* ***not segmented****.* 22

3.5 *Effect of fine-tuning on the target domain data. Perplexities of an LSTM language model on AMI.* 23

3.6 *Character level perplexities of word-level and BPE-level LSTM language models on LibriSpeech.* 24

3.7 *Comparison of different feed-forward layer types. Perplexities are reported with* ***2-layer*** *models on Quaero development set.* 25

3.8 *Effect of the depth. Perplexities on Quaero development set.* 25

3.9 *Perplexities on Quaero development set. The number of hidden units are set to 300 in each layer.* 27

3.10 *Perplexities on Quaero set. The number of hidden units are set to 2048 in each layer. Dropout of 20% is used.* 27

3.11 *Perplexity results on Quaero for neural language models with an additional bag-of-words input feature. All models including the 4-gram Kneser-Ney model are trained on 50 M words for comparison. A hidden layer size of 500 is used.* 30

3.12 *Perplexity and WER (in %) results on Quaero for neural language models with an additional bag-of-words input feature. Perplexities are those of models interpolated with the 4-gram Kneser-Ney model trained on 3.1 B.* 30

4.1 *Perplexity on word level LibriSpeech* ***after 2.5 epoch*** *(25 sub-epochs in our setup; 6.5 M updates). The number of heads H* is 8 for all models below. 43

4.2 *Effect of number of heads. Perplexity on word level LibriSpeech* ***after 2.5 epoch*** *for* $(L = 12, d_{ff} = 2048, d_{res} = 512, H)$. 44
4.3 *Effect of activation functions. Perplexity on word level LibriSpeech* ***after 1 epoch*** *(10 sub-epochs in our setup) for* $(L = 24, d_{ff} = 2048, d_{res} = 512, H = 8)$. 44
4.4 *Final perplexities on LibriSpeech* ***after full convergence****. The baseline 4-gram and LSTM numbers are taken from Table 3.3.* d_{res} *is 512 for all Transformer models.* 44
4.5 *Effect of gate bias initialization. Perplexity on the LibriSpeech Dev set after 1 sub-epoch for* $(L = 24, d_{ff} = 2048, d_{ff} = 512, H = 8)$ *with highway connections.* 45
4.6 *Residual connection vs. Highway connection in Transformer models* $(L = 24, d_{ff} = 2048, d_{res} = 512, H = 8)$*. Perplexity after convergence.* 45
4.7 *Perplexity on LibriSpeech* ***after 2.5 epoch*** *for* $(L, d_{ff} = 8192, d_{res} = 1024, H = 16)$ *models with shared parameters across all layers.* 46
4.8 *WERs (%) for* ***hybrid NN-HMM*** *systems on LibriSpeech. The 4-gram model is used in the first pass to generate lattices for rescoring. The row "Lattice" shows oracle WERs of the lattices.* 47
4.9 *WERs (%) for* ***attention-based models*** *on LibriSpeech. Perplexities are on the 10 K BPE level.* 47
4.10 *Effect of sinusoidal positional encoding. Perplexity* ***after 5 epochs*** *(13 M updates; full convergence) for* $(L, d_{ff} = 2048, d_{res} = 512, H = 8)$ *models.* 49
4.11 ***Perplexity*** *of the word-level (152K vocab)* ***baseline*** *models on* ***TED-LIUM 2****.* . 57
4.12 *Perplexity of the word-level (152K vocab) models on* ***TED-LIUM 2****.* $d_{kv} = 768$ *and* $H = 12$ *for all models. The models with* $F = 1$ *are standard Transformers.* . . 57
4.13 ***Perplexity*** *of the word-level (200K vocab) model on* ***LibriSpeech****.* d_{kv} *is* ***512*** *for all models. The numbers for the standard models are taken from Table 4.4.* 58
4.14 *Effect of sharing KV for both standard and small state Transformers.* ***Perplexity*** *on* ***TED-LIUM 2*** *(152K vocab).* 58
4.15 *WERs on* ***TED-LIUM 2****. Perplexities are after* ***interpolation*** *with the 4-gram LM. Lattices are generated by either 4-gram or 4-gram + LSTM LMs in the* ***first pass****.* 59
4.16 *Perplexities and word error rates overview comparing LSTM and Transformer (Trafo) language models across different ASR datasets. A 4-gram Kneser-Ney language model is used to generate the lattices in all tasks except AMI for which 3-gram is used, and lattice rescoring is carried out using either the LSTM or Transformer language model, except for the LibriSpeech BPE level experiment which uses the attention based end-to-end system and shallow fusion. Except for the LibriSpeech experiments, the reported perplexities obtained by interpolating the rescoring neural language model with the n-gram language model. For LibriSpeech, Dev and Eval correspond to* ***dev-other*** *and* ***eval-other****. For Switchboard, numbers for* ***Switchboard*** *and* ***CallHome*** *parts of Hub5_00 set are presented as Dev and Eval for this table. "Train" indicates the number of tokens in the training data, and "Voc" indicates the vocabulary size.* 60
4.17 *Perplexities and word error rates for model combination between LSTM and Transformer language models across* ***standard ASR datasets****. For Switchboard, numbers for* ***Switchboard*** *and* ***CallHome*** *parts of Hub5_00 set are presented as Dev and Eval for this table. "Train" indicates the number of words in the training data, and "Voc" indicates the vocabulary size.* 61

5.1 *Results of* ***knowledge distillation****.* ***Perplexities*** *for the word-level* ***TED-LIUM 2****.* 66
5.2 *Perplexity results of* ***knowledge distillation*** *based on the class based output.* . . . 69

5.3 *Perplexity results on Switchboard of knowledge distillation based on class based output, using* ***contexts across sentence boundaries (up to 100 words)***. 69
5.4 *Perplexity results for MSE based distillation using the gated linear unit (GLU) or the gated tangent unit (GTU) in the final hidden layer. The baseline perplexities are copied from Table 5.2 for easy comparison.* 70
5.5 *MLP vs. CNN with class output based distillation. The best perplexities for the MLP are copied from Table 5.2 for easy comparison. All modes are* ***5-grams***. . . . 70
5.6 *WER results on Switchboard. All results are reported after* ***interpolation*** *with the baseline count model.* 70

6.1 *YouTube training data split by categories. "Self weight" indicates the optimal interpolation weights for 5-gram count models trained on each domain when minimizing the perplexity on the subset of the validation set with the same domain (not all domains are in the validation set). 9 categories with the highest self weight are in* ***bold***. 76
6.2 *Perplexity overview for the YouTube dataset. The validation perplexities are split by categories. Background and RADMM are* ***single*** *models while Experts are* ***one model per category***. 78
6.3 *WER results on the YouTube eval set. Perplexities computed on the second pass 133 K vocabulary.* 79
6.4 *Perplexities on the YouTube data of models based on* ***8192****-unit LSTMs.* 82
6.5 *Interpolation weights (scaled by factor 100) for each domain on the AppTek development text for 4-gram models. We removed values smaller than* 10^{-2}. *We show* ***8 most relevant subsets*** *out of 33.* 84
6.6 *Perplexities for the* ***sampled softmax*** *case. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).* . . 85
6.7 *Perplexities of the LSTM models for the* ***NCE*** *case. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).* 86
6.8 *Perplexities of small LSTM student models on the AppTek dataset trained with the NCE loss. When an additional linear bottleneck layer is inserted before softmax (Bottleneck), its dimension is set to 512.* 87
6.9 *Perplexities for the* ***sampled softmax*** *case using Transformer teachers. Expert models are interpolated to form the teacher model. Interpolation weights are either optimized for each domain (Domain optimized) or only optimized once on the whole development set (All).* 87
6.10 *WERs (%) on the AppTek data for first pass recognition experiments using LSTM student models trained with the NCE loss. The perplexity (PPL) column in the case where the explicit normalization is not carried out, indicates the pseudo-perplexity.* 88

7.1 ***Sentence lengths statistics*** *on Switchboard and Quaero training and evaluation datasets.* 94
7.2 *Perplexities of* ***LSTM-RNN*** *on* ***Switchboard***. CCO denotes context carry-over. We report average sequence length information for Hub5_00 which is similar to Hub5e_01. 96
7.3 *Perplexities of* ***LSTM-RNN*** *on* ***Quaero***. *CCO denotes context carry-over. We report average sequence length information for the development set which is similar to the evaluation data.* 96

7.4 ***ASR results** of **LSTM-RNN** on Switchboard 300 h **Hub5_00** set. Perplexities (PPL) after interpolation with the 4-gram Kneser-Ney language model. CCO denotes context carry-over.* 97
7.5 ***ASR results** of **LSTM-RNN** on **Quaero**. Perplexities (PPL) are after interpolation with the 4-gram Kneser-Ney language model. CCO denotes context carry-over.* 97
7.6 *Perplexities of **Transformers** on **Switchboard** (no positional encoding is used except for the model in the last row). CCO denotes context carry-over.* 99
7.7 *Perplexities of **Transformers** on **Quaero**. CCO denotes context carry-over.* . . . 99
7.8 ***ASR results** of **Transformers** on Switchboard-300 h **Hub5_00** set. Perplexities (PPL) are after interpolation with the 4-gram Kneser-Ney language model.* 100
7.9 ***ASR results** of **Transformers** on **Quaero**. Perplexities (PPL) are after interpolation with the 4-gram model.* 100
7.10 *BLEU and TER results for WMT 2016 Romanian-English task. The baseline NMT performance was provided by Arne Nix, which is reported in [Nix & Kim+ 19].* . . 102

A.1 ***Perplexity** of the word-level (152K vocab) **baseline** models on **TED-LIUM 2**.* . 116
A.2 *Number of running words, OOV rates and average sentence lengths in terms of number of words (Avg. length) of all data sets and subsets used. The vocabulary size is 30 K.* 118
A.3 *Standalone perplexities of the 48 K vocabulary word-level **baseline** models on **AMI**. Perplexities after fine-tuning on the AMI transcriptions.* 119

Bibliography

[Abadi & Barham+ 16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al.: TensorFlow: A system for large-scale machine learning. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, Savannah, GA, USA, Nov. 2016.

[Al-Rfou & Choe+ 19] R. Al-Rfou, D. Choe, N. Constant, M. Guo, L. Jones: Character-level language modeling with deeper self-attention. In *Proc. Conference on Artificial Intelligence (AAAI)*, pp. 3159–3166, Honolulu, HI, USA, Jan. 2019.

[Allauzen & Riley 11] C. Allauzen, M. Riley: Bayesian Language Model Interpolation for Mobile Speech Input. In *Proc. Interspeech*, pp. 1429–1432, Florence, Italy, Aug. 2011.

[Auli & Galley+ 13] M. Auli, M. Galley, C. Quirk, G. Zweig: Joint Language and Translation Modeling with Recurrent Neural Networks. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1044–1054, Seatle, WA, USA, Oct. 2013.

[Ba & Caruana 14] J. Ba, R. Caruana: Do Deep Nets Really Need to be Deep? In *Proc. Advances in Neural Information Processing Systems (NIPS)*, Vol. 27, pp. 2654–2662, Quebec, Canada, Dec. 2014.

[Ba & Kiros+ 16] J.L. Ba, J.R. Kiros, G.E. Hinton: Layer Normalization. Preprint arXiv:1607.06450, 2016.

[Baevski & Auli 19] A. Baevski, M. Auli: Adaptive input representations for neural language modeling. In *Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[Bahdanau & Cho+ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural machine translation by jointly learning to align and translate. In *Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Bahdanau & Chorowski+ 16] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio: End-to-end attention-based large vocabulary speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949, Shanghai, China, March 2016.

[Bahl & Jelinek+ 83] L.R. Bahl, F. Jelinek, R.L. Mercer: A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, pp. 179–190, March 1983.

[Baker 75] J.K. Baker: Stochastic Modeling for Automatic Speech Understanding. In *Speech Recognition*. Academic Press, New York, NY, USA, 1975.

[Bayes 63] T. Bayes: An essay towards solving a problem in the doctrine of chances. *Philosophical transactions*, Vol. 53, pp. 370–418, 1763.

[Beck & Zhou+ 19] E. Beck, W. Zhou, R. Schlüter, H. Ney: LSTM Language Models for LVCSR in First-Pass Decoding and Lattice-Rescoring. Preprint arXiv:1907.01030, July 2019.

[Bellman 57] R.E. Bellman: Dynamic Programming. Princeton University Press, 1957.

[Bengio 12] Y. Bengio: Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pp. 437–478. Springer, 2012.

[Bengio & De Mori+ 91] Y. Bengio, R. De Mori, G. Flammia, R. Kompe: Global optimization of a neural network-hidden Markov model hybrid. In *IEEE International Joint Conference on Neural Networks*, pp. 789–794, Seatle, WA, USA, Nov. 1991.

[Bengio & Ducharme+ 00] Y. Bengio, R. Ducharme, P. Vincent: A Neural Probabilistic Language Model. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, Vol. 13, pp. 932–938, Denver, CO, USA, 2000.

[Bengio & Ducharme+ 03] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin: A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, Vol. 3, pp. 1137–1155, 2003.

[Botros & Irie+ 15] R. Botros, K. Irie, M. Sundermeyer, H. Ney: On Efficient Training of Word Classes and Their Application to Recurrent Neural Network Language Models. In *Proc. Interspeech*, pp. 1443–1447, Dresden, Germany, Sept. 2015.

[Bourlard & Morgan 89] H. Bourlard, N. Morgan: A Continuous Speech Recognition System Embedding MLP into HMM. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 186–193. Denver, CO, USA, Nov. 1989.

[Bourlard & Morgan 94] H. Bourlard, N. Morgan: *Connectionist speech recognition: a hybrid approach*, Vol. 247. Springer, 1994.

[Brown & Desouza+ 92] P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, J.C. Lai: Class-based n-gram models of natural language. *Computational linguistics*, Vol. 18, No. 4, pp. 467–479, 1992.

[Buciluǎ & Caruana+ 06] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil: Model compression. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Disc. and Data Mining*, pp. 535–541, Philadelphia, PA, USA, Aug. 2006.

[Chan & Jaitly+ 16] W. Chan, N. Jaitly, Q. Le, O. Vinyals: Listen, Attend and Spell: a Neural Network for Large Vocabulary Conversational Speech Recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964, Shanghai, China, March 2016.

[Chan & Ke+ 15] W. Chan, N.R. Ke, I. Lane: Transferring knowledge from a RNN to a DNN. In *Proc. Interspeech*, pp. 3264–3268, Dresden, Germany, Sept. 2015.

[Chelba & Brants+ 10] C. Chelba, T. Brants, W. Neveitt, P. Xu: Study on interaction between entropy pruning and kneser-ney smoothing. In *Proc. Interspeech*, pp. 2422–2425, Makuhari, Japan, Sept. 2010.

[Chen & Beeferman$^+$ 98] S.F. Chen, D. Beeferman, R. Rosenfield: Evaluation metrics for language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp. 275–280, Lansdowne, VA, USA, Feb. 1998.

[Chen & Firat$^+$ 18] M.X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, M. Hughes: The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Proc. Association for Computational Linguistics (ACL)*, pp. 76–86, Melbourne, Australia, July 2018.

[Chen & Goodman 99] S.F. Chen, J. Goodman: An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, Vol. 13, No. 4, pp. 359–393, 1999.

[Chen & Liu$^+$ 15] X. Chen, X. Liu, M.J.F. Gales, P.C. Woodland: Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5411–5415, Brisbane, Australia, April 2015.

[Chen & Ragni$^+$ 17] X. Chen, A. Ragni, X. Liu, M.J. Gales: Investigating Bidirectional Recurrent Neural Network Language Models for Speech Recognition. In *Proc. Interspeech*, pp. 269–273, Stockholm, Sweden, Aug. 2017.

[Chen & Wang$^+$ 14] X. Chen, Y. Wang, X. Liu, M.J.F. Gales, P.C. Woodland: Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch. In *Proc. Interspeech*, pp. 641–645, Singapore, Sept. 2014.

[Cheng & Dong$^+$ 16] J. Cheng, L. Dong, M. Lapata: Long Short-Term Memory-Networks for Machine Reading. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 551–561, Austin, TX, USA, Nov. 2016.

[Chiu & Sainath$^+$ 18] C. Chiu, T.N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R.J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, M. Bacchiani: State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778, Calgary, Canada, April 2018.

[Cho & Gülçehre$^+$ 14] K. Cho, Ç. Gülçehre, B. van Merriënboer, D. Bahdanau, F.B.H. Schwenk, Y. Bengio: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, Oct. 2014.

[Chorowski & Bahdanau$^+$ 15] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio: Attention-Based Models for Speech Recognition. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 577–585. Montréal, Canada, Dec. 2015.

[Chung & Gülçehre$^+$ 14] J. Chung, Ç. Gülçehre, K. Cho, Y. Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Deep Learning workshop at Conf. on Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, Dec. 2014.

[Clarkson & Robinson 97] P.R. Clarkson, A.J. Robinson: Language model adaptation using mixtures and an exponentially decaying cache. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 2, pp. 799–802, Munich, Germany, April 1997.

[Clarkson & Robinson 98] P. Clarkson, T. Robinson: The applicability of adaptive language modelling for the broadcast news task. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, pp. 233–236, Sydney, Australia, 1998.

[Clevert & Unterthiner+ 16] D.A. Clevert, T. Unterthiner, S. Hochreiter: Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *Int. Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[Conneau & Lample 19] A. Conneau, G. Lample: Cross-lingual Language Model Pretraining. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 7057–7067, Vancouver, Canada, Dec. 2019.

[Cui & Kingsbury+ 17] J. Cui, B. Kingsbury, B. Ramabhadran, G. Saon, T. Sercu, K. Audhkhasi, A. Sethy, M. Nussbaum-Thom, A. Rosenberg: Knowledge Distillation Across Ensembles of Multilingual Models for Low-resource Languages. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4825–4829, New Orleans, LA, USA, March 2017.

[Dahl & Yu+ 12] G.E. Dahl, D. Yu, L. Deng, A. Acero: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 1, pp. 30–42, 2012.

[Dai & Yang+ 19] Z. Dai, Z. Yang, Y. Yang, W.W. Cohen, J. Carbonell, Q.V. Le, R. Salakhutdinov: Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. Association for Computational Linguistics (ACL)*, pp. 2978–2988, Florence, Italy, July 2019.

[Das & Li+ 19] A. Das, J. Li, C. Liu, Y. Gong: Universal Acoustic Modeling Using Neural Mixture Models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5681–5685, Brighton, UK, May 2019.

[Dauphin & Fan+ 17] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier: Language Modeling with Gated Convolutional Networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, pp. 933–941, Sydney, Australia, Aug. 2017.

[Dauphin & Schoenholz 19] Y.N. Dauphin, S. Schoenholz: MetaInit: Initializing learning by learning to initialize. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 12624–12636. Vancouver, Canada, Dec. 2019.

[David & Mermelstein 80] S. David, P. Mermelstein: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 87, No. 4, pp. 1738–1752, Aug. 1980.

[Dehghani & Gouws+ 19] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, Ł. Kaiser: Universal Transformers. In *Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[Devlin 15] J. Devlin: A Practical Guide to Real-Time Machine Translation. In *Proc. EMNLP 2015 - Tenth Workshop on Statistical Machine Translation (WMT), Invited talk*, Lisbon, Portugal, Sept. 2015.

[Devlin & Chang+ 19] J. Devlin, M. Chang, K. Lee, K. Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. North American Chapter of the Association for Computational Linguistics on Human Language Technologies (NAACL-HLT)*, pp. 4171–4186, Minneapolis, MN, USA, June 2019.

[Devlin & Quirk+ 15] J. Devlin, C. Quirk, A. Menezes: Pre-Computable Multi-Neural Network Language Models. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 256–260, Lisbon, Portugal, Sept. 2015.

[Devlin & Zbib+ 14] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul: Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proc. Assoc. for Computational Linguistics (ACL)*, pp. 1370–1380, Baltimore, Maryland, June 2014.

[Duchi & Hazan+ 11] J. Duchi, E. Hazan, Y. Singer: Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159, 2011.

[Duda & Hart 73] R.O. Duda, P.E. Hart: Pattern recognition and scene analysis, 1973.

[Elman 90] J.L. Elman: Finding structure in time. *Cognitive science*, Vol. 14, No. 2, pp. 179–211, 1990.

[Franzini & Lee+ 90] M. Franzini, K.F. Lee, A. Waibel: Connectionist Viterbi training: a new hybrid method for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 425–428, Albuquerque, NM, USA, April 1990.

[Frinken & Zamora-Martinez+ 12] V. Frinken, F. Zamora-Martinez, S. Espana-Boquera, M.J. Castro-Bleda, A. Fischer, H. Bunke: Long-short term memory neural networks language modeling for handwriting recognition. In *Proc. International Conference on Pattern Recognition (ICPR)*, pp. 701–704, Tsukuba, Japan, Nov. 2012.

[Gal & Ghahramani 16] Y. Gal, Z. Ghahramani: A theoretically grounded application of dropout in recurrent neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 1019–1027, Barcelona, Spain, Dec. 2016.

[Gangireddy & Swietojanski+ 16] S.R. Gangireddy, P. Swietojanski, P. Bell, S. Renals: Unsupervised Adaptation of Recurrent Neural Network Language Models. In *Proc. Interspeech*, pp. 2333–2337, San Francisco, CA, USA, Sept. 2016.

[Garmash & Monz 16] E. Garmash, C. Monz: Ensemble Learning for Multi-Source Neural Machine Translation. In *Proc. Int. Conf. on Comp. Linguistics (COLING)*, pp. 1409–1418, Osaka, Japan, Dec. 2016.

[Gehring & Auli+ 17] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin: Convolutional Sequence to Sequence Learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, pp. 1243–1252, Sydney, Australia, Aug. 2017.

[Geras & Mohamed+ 16] K.J. Geras, A.R. Mohamed, R. Caruana, G. Urban, S. Wang, O. Aslan, M. Philipose, M. Richardson, C. Sutton: Blending LSTMs into CNNs. In *Workshop Track of International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[Gers & Schmidhuber+ 00] F.A. Gers, J. Schmidhuber, F. Cummins: Learning to forget: Continual prediction with LSTM. *Neural computation*, Vol. 12, No. 10, pp. 2451–2471, 2000.

[Gers & Schraudolph+ 03] F.A. Gers, N.N. Schraudolph, J. Schmidhuber: Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, Vol. 3, pp. 115–143, 2003.

[Gerstenberger 20] A. Gerstenberger: Domain Robust, Fast, and Compact Neural Language Models for ASR. Bachelor thesis, RWTH Aachen University, April 2020.

[Gerstenberger & Irie+ 20] A. Gerstenberger, K. Irie, P. Golik, H. Ney: Domain Robust, Fast, and Compact Neural Language Models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, to appear, Barcelona, Spain, May 2020.

[Goldberger & Melamud 018] J. Goldberger, O. Melamud: Self-Normalization Properties of Language Modeling. In *Proc. Assoc. for Computational Linguistics (ACL)*, pp. 764–773, Santa Fe, USA, Aug. 2018.

[Golik & Tüske+ 17] P. Golik, Z. Tüske, K. Irie, E. Beck, R. Schlüter, H. Ney: The 2016 RWTH Keyword Search System for Low-Resource Languages. In I.M. Alexey Karpov, Rodmonga Potapova, editor, *International Conference Speech and Computer*, Vol. 10458 of *Lecture Notes in Computer Science, Subseries Lecture Notes in Artificial Intelligence*, pp. 719–730, Hatfield, UK, Sept. 2017. Springer Cham, Switzerland.

[Goodfellow & Warde-Farley+ 13] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio: Maxout networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, Vol. 28, pp. 1319–1327, Atlanta, GA, USA, June 2013.

[Goodman 01] J. Goodman: Classes for fast maximum entropy training. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 561–564, Salt Lake City, UT, USA, May 2001.

[Graves 12] A. Graves: Sequence transduction with recurrent neural networks. In *Representation Learning Workshop, Int. Conf. on Machine Learning (ICML)*, Edinburgh, Scotland, June 2012.

[Graves 13] A. Graves: Generating sequences with recurrent neural networks. Preprint arXiv:1308.0850, 2013.

[Graves & Fernández+ 06] A. Graves, S. Fernández, F.J. Gomez, J. Schmidhuber: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, pp. 369–376, Pittsburgh, PA, USA, June 2006.

[Greff & Srivastava+ 17] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber: LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 10, pp. 2222–2232, 2017.

[Gülçehre & Firat+ 17] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.C. Lin, F. Bougares, H. Schwenk, Y. Bengio: On Using Monolingual Corpora in Neural Machine Translation. *Computer Speech & Language*, Vol. 45, pp. 137–148, Sept. 2017.

[Gutmann & Hyvärinen 10] M. Gutmann, A. Hyvärinen: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. Int. Conf. on AI and Statistics*, pp. 297–304, 2010.

[Halpern & Hall+ 16] Y. Halpern, K. Hall, V. Schogol, M. Riley, B. Roark, G. Skobeltsyn, M. Buml: Contextual Prediction Models for Speech Recognition. In *Proc. Interspeech*, pp. 2338–2342, San Francisco, CA, USA, Sept. 2016.

[Hampshire & Waibel 92] J.B. Hampshire, A. Waibel: The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 7, pp. 751–769, 1992.

[Han & Chandrashekaran+ 17] K.J. Han, A. Chandrashekaran, J. Kim, I. Lane: The CAPIO 2017 conversational speech recognition system. Preprint arXiv:1801.00059, 2017.

[Han & Prieto+ 19] K.J. Han, R. Prieto, K. Wu, T. Ma: State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, pp. 54–61, Sentosa, Singapore, 2019.

[Hannun & Lee+ 19] A. Hannun, A. Lee, Q. Xu, R. Collobert: Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions. Preprint arXiv:1904.02619, 2019.

[He & Tan+ 18] T. He, X. Tan, Y. Xia, D. He, T. Qin, Z. Chen, T.Y. Liu: Layer-wise coordination between encoder and decoder for neural machine translation. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7944–7954, Montréal, Canada, Dec. 2018.

[He & Zhang+ 16a] K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[He & Zhang+ 16b] K. He, X. Zhang, S. Ren, J. Sun: Identity Mappings in Deep Residual Networks. In *Proc. European Conf. on Computer Vision (ECCV)*, pp. 630–645, Amsterdam, Netherlands, Oct. 2016.

[Hendrycks & Gimpel 18] D. Hendrycks, K. Gimpel: Gaussian Error Linear Units (GELUs). Preprint arXiv:1606.08415, 2018.

[Hermansky & Ellis+ 00] H. Hermansky, D.P.W. Ellis, S. Sharma: Tandem connectionist feature extraction for conventional HMM systems. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1635–1638, Istanbul, Turkey, June 2000.

[Hinton & Srivastava+ 12] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov: Improving neural networks by preventing co-adaptation of feature detectors. Preprint arXiv:1207.0580, 2012.

[Hinton & Vinyals+ 14] G. Hinton, O. Vinyals, J. Dean: Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, Montreal, Canada, Dec. 2014.

[Hochreiter & Schmidhuber 96] S. Hochreiter, J. Schmidhuber: LSTM Can Solve Hard Long Time Lag Problems. In *Proc. Conference on Neural Information Processing Systems (NIPS)*, pp. 473–479, Cambridge, MA, USA, 1996.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[Hoffmeister 11] B. Hoffmeister: *Bayes Risk Decoding and its Application to System Combination.* Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, July 2011.

[Hori & Kubo+ 14] T. Hori, Y. Kubo, A. Nakamura: Real-time one-pass decoding with recurrent neural network language model for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6364–6368, Florence, Italy, May 2014.

[Huang & Acero+ 01] X. Huang, A. Acero, H.W. Hon, R. Foreword By-Reddy: *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice hall PTR, 2001.

[Huang & Sethy+ 17] Y. Huang, A. Sethy, B. Ramabhadran: Fast Neural Network Language Model Lookups at N-Gram Speeds. In *Proc. Interspeech*, pp. 274–278, Stokholm, Sweden, Aug. 2017.

[Huang & Zweig+ 14] Z. Huang, G. Zweig, B. Dumoulin: Cache based recurrent neural network language model inference for first pass speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6354–6358, Florence, Italy, May 2014.

[Hwang & Sung 17] K. Hwang, W. Sung: Character-level language modeling with hierarchical recurrent neural networks. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5720–5724, New Orleans, LA, USA, March 2017.

[ICSI 00] ICSI: Berkeley, "Quicknet". http://www1.icsi.berkeley.edu/Speech/qn.html, 2000.

[Irie & Gerstenberger+ 20] K. Irie, A. Gerstenberger, R. Schlüter, H. Ney: How much self-attention do we need? Trading attention for feed-forward layers. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, to appear, Barcelona, Spain, May 2020.

[Irie & Golik+ 17] K. Irie, P. Golik, R. Schlüter, H. Ney: Investigations on byte-level convolutional neural networks for language modeling in low resource speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5740–5744, New Orleans, LA, USA, March 2017.

[Irie & Kumar+ 18] K. Irie, S. Kumar, M. Nirschl, H. Liao: RADMM: Recurrent Adaptive Mixture Model with Applications to Domain Robust Language Modeling. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6079–6083, Calgary, Canada, April 2018.

[Irie & Lei+ 18a] K. Irie, Z. Lei, L. Deng, R. Schlüter, H. Ney: Investigation on Estimation of Sentence Probability By Combining Forward, Backward and Bi-directional LSTM-RNNs. In *Proc. Interspeech*, pp. 392–395, Hyderabad, India, Sept. 2018.

[Irie & Lei+ 18b] K. Irie, Z. Lei, R. Schlüter, H. Ney: Prediction of LSTM-RNN Full Context States as a Subtask for N-gram Feedforward Language Models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6104–6108, Calgary, Canada, April 2018.

[Irie & Prabhavalkar+ 19a] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, P. Nguyen: Model Unit Exploration for Sequence-to-Sequence Speech Recognition. Preprint arXiv:1902.01955, 2019.

[Irie & Prabhavalkar+ 19b] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, P. Nguyen: On the Choice of Modeling Unit for Sequence-to-Sequence Speech Recognition. In *Proc. Interspeech*, pp. 3800–3804, Graz, Austria, Sept. 2019.

[Irie & Schlüter+ 15] K. Irie, R. Schlüter, H. Ney: Bag-of-Words Input for Long History Representation in Neural Network-based Language Models for Speech Recognition. In *Proc. Interspeech*, pp. 2371–2375, Dresden, Germany, Sept. 2015.

[Irie & Tüske+ 16] K. Irie, Z. Tüske, T. Alkhouli, R. Schlüter, H. Ney: LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition. In *Proc. Interspeech*, pp. 3519–3523, San Francisco, CA, USA, Sept. 2016.

[Irie & Zeyer+ 19a] K. Irie, A. Zeyer, R. Schlüter, H. Ney: Language Modeling with Deep Transformers. In *Proc. Interspeech*, pp. 3905–3909, Graz, Austria, Sept. 2019.

[Irie & Zeyer+ 19b] K. Irie, A. Zeyer, R. Schlüter, H. Ney: Training Language Models for Long-Span Cross-Sentence Evaluation. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, pp. 419–426, Sentosa, Singapore, Dec. 2019.

[Iyer & Ostendorf 99] R. Iyer, M. Ostendorf: Modeling Long Distance Dependence in Language: Topic Mixtures versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing*, Vol. 7, No. 1, pp. 30–39, 1999.

[Jacobs & Jordan+ 91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton: Adaptive mixtures of local experts. *Neural computation*, Vol. 3, No. 1, pp. 79–87, 1991.

[Jean & Cho+ 15] S. Jean, K. Cho, R. Memisevic, Y. Bengio: On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proc. Association for Computational Linguistics (ACL)*, pp. 1–10, Beijing, China, July 2015.

[Jelinek 76] F. Jelinek: Continuous Speech Recognition by Statistical Methods. *Proceedings of the IEEE*, Vol. 64, No. 10, pp. 532–556, April 1976.

[Jelinek & Bahl+ 75] F. Jelinek, L. Bahl, R. Mercer: Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, Vol. 21, No. 3, pp. 250–256, 1975.

[Jelinek & Mercer+ 77] F. Jelinek, R.L. Mercer, L.R. Bahl, J.K. Baker: Perplexitya measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, Vol. 62, No. S1, pp. S63–S63, 1977.

[Jelinek & Merialdo+ 91] F. Jelinek, B. Merialdo, S. Roukos, M. Strauss: A Dynamic Language Model for Speech Recognition. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp. 293–295, Feb. 1991.

[Ji & Cohn+ 16] Y. Ji, T. Cohn, L. Kong, C. Dyer, J. Eisenstein: Document context language models. In *Int. Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[Jozefowicz & Vinyals+ 16] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu: Exploring the limits of language modeling. Preprint arXiv:1602.02410, 2016.

[Jozefowicz & Zaremba+ 15] R. Jozefowicz, W. Zaremba, I. Sutskever: An empirical exploration of recurrent network architectures. In *Proc. Int. Conf. on Machine Learning (ICML)*, pp. 2342–2350, Lille, France, July 2015.

[Karita & Chen+ 19] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N.E.Y. Soplin, R. Yamamoto, X. Wang et al.: A comparative study on transformer vs rnn in speech applications. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Sentosa, Singapore, 2019.

[Kim & Rush 16a] Y. Kim, A.M. Rush: Sequence-Level Knowledge Distillation. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1317–1327, Austin, TX, USA, Nov. 2016.

[Kim & Rush 16b] Y. Kim, Y.J.D.S.A. Rush: Character-Aware Neural Language Models. In *Proc. AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, Feb. 2016.

[Kim & Stratos+ 17] Y.B. Kim, K. Stratos, D. Kim: Domain Attention with an Ensemble of Experts. In *Proc. Association for Computational Linguistics (ACL)*, pp. 643–653, Vancouver, Canada, July 2017.

[Kingma & Ba 15] D.P. Kingma, J. Ba: Adam: A Method for Stochastic Optimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Kitaev & Kaiser+ 20] N. Kitaev, L. Kaiser, A. Levskaya: Reformer: The Efficient Transformer. In *Int. Conf. on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[Kitza & Golik+ 19] M. Kitza, P. Golik, R. Schlter, H. Ney: Cumulative Adaptation for BLSTM Acoustic Models. In *Proc. Interspeech*, pp. 754–758, Graz, Austria, Sept. 2019.

[Klakow & Peters 02] D. Klakow, J. Peters: Testing the correlation of word error rate and perplexity. *Speech Communication*, Vol. 38, No. 1, pp. 19–28, 2002.

[Kneser & Ney 91] R. Kneser, H. Ney: Forming word classes by statistical clustering for statistical language modelling. In *Proc. First Int. Conf. on Quantitative Linguistics (QUALICO)*, pp. 221–226, Trier, Germany, 1991.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved backing-off for m-gram language modeling. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 181–184, Detroit, MI, USA, May 1995.

[Kneser & Steinbiss 93] R. Kneser, V. Steinbiss: On the dynamic adaptation of stochastic language models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 586–589, Minneapolis, MN, USA, April 1993.

[Kozielski & Nuhn+ 14] M. Kozielski, M. Nuhn, P. Doetsch, H. Ney: Towards Unsupervised Learning for Handwriting Recognition. In *Proc. International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 549–554, Crete, Greece, Sept. 2014.

[Kuhn & De Mori 90] R. Kuhn, R. De Mori: A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 6, pp. 570–583, 1990.

[Kumar & Nirschl+ 17] S. Kumar, M. Nirschl, D. Holtmann-Rice, H. Liao, A. Theertha Suresh, F. Yu: Lattice rescoring strategies for long short-term memory language models in speech recognition. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Okinawa, Japan, Dec. 2017.

[Kuncoro & Ballesteros+ 16] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, N.A. Smith: Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1744–1753, Austin, TX, USA, Nov. 2016.

[Lample & Sablayrolles+ 19] G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, H. Jégou: Large memory layers with product keys. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, Dec. 2019.

[Lan & Chen+ 19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. Preprint arXiv:1909.11942, Sept. 2019.

[Lee & Park+ 15] K. Lee, C. Park, I. Kim, N. Kim, J. Lee: Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR. In *Proc. Interspeech*, pp. 2102–2106, Dresden, Germany, Sept. 2015.

[Levenshtein 66] V.I. Levenshtein: Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklay*, Vol. 10, No. 10, pp. 707–710, 1966.

[Li & Zhao+ 14] J. Li, R. Zhao, J.T. Huang, Y. Gong: Learning small-size DNN with output-distribution-based criteria. In *Proc. Interspeech*, pp. 1910–1914, Singapore, Sept. 2014.

[Liao & McDermott+ 13] H. Liao, E. McDermott, A. Senior: Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, pp. 368–373, Olomouc, Czech Republic, Dec. 2013.

[Lin & Feng+ 17] Z. Lin, M. Feng, C.N.d. Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio: A structured self-attentive sentence embedding. In *Int. Conf. on Learning Representations (ICLR)*, Toulon, France, April 2017.

[Lin & Liu+ 15] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, S. Li: Hierarchical Recurrent Neural Network for Document Modeling. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 899–907, Lisbon, Portugal, Sept. 2015.

[Lippmann 88] R.P. Lippmann: An introduction to computing with neural nets. In *Artificial neural networks: theoretical concepts*, pp. 36–54, 1988.

[Liu & Chen+ 16] X. Liu, X. Chen, Y. Wang, M.J. Gales, P.C. Woodland: Two efficient lattice rescoring methods using recurrent neural network language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 24, No. 8, pp. 1438–1449, 2016.

[Liu & Saleh+ 18] P.J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, Ł. Kaiser, N. Shazeer: Generating wikipedia by summarizing long sequences. In *Int. Conf. on Learning Representations (ICLR)*, Vancouver, Canada, April 2018.

[Liu & Wang+ 14] X. Liu, Y. Wang, X. Chen, M.J.F. Gales, P.C. Woodland: Efficient lattice rescoring using recurrent neural network language models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4908–4912, Florence, Italy, May 2014.

[Lu & Guo+ 17] L. Lu, M. Guo, S. Renals: Knowledge distillation for small-footprint highway networks. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4820–4824, New Orleans, LA, USA, March 2017.

[Lu & Zhang+ 15] L. Lu, X. Zhang, K. Cho, S. Renals: A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. In *Proc. Interspeech*, pp. 3249–3253, Dresden, Germany, Sept. 2015.

[Luong & Pham+ 15] M.T. Luong, H. Pham, C.D. Manning: Effective Approaches to Attention-based Neural Machine Translation. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, Lisbon, Portugal, Sept. 2015.

[Lüscher & Beck+ 19] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, H. Ney: RWTH ASR Systems for LibriSpeech: Hybrid vs Attention. In *Proc. Interspeech*, pp. 231–235, Graz, Austria, Sept. 2019.

[Ma & Collins 18] Z. Ma, M. Collins: Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3698–3707, Brussels, Belgium, Oct.-Nov. 2018.

[Ma & Nirschl+ 17] M. Ma, M. Nirschl, F. Biadsy, S. Kumar: Approaches for Neural-Network Language Model Adaptation. In *Proc. Interspeech*, pp. 259–263, Stockholm, Sweden, Aug. 2017.

[Makhoul & Schwartz 95] J. Makhoul, R. Schwartz: State of the art in continuous speech recognition. *Proceedings of the National Academy of Sciences*, Vol. 92, No. 22, pp. 9956–9963, Oct. 1995.

[Makino & Kawabata+ 83] S. Makino, T. Kawabata, K. Kido: Recognition of consonant based on the perceptron model. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 738–741, Boston, MA, USA, April 1983.

[Masumura & Tanaka+ 18] R. Masumura, T. Tanaka, A. Ando, H. Masataki, Y. Aono: Role Play Dialogue Aware Language Models Based on Conditional Hierarchical Recurrent Encoder-Decoder. In *Proc. Interspeech*, pp. 1259–1263, 2018.

[McCowan & Carletta+ 05] I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos et al.: The AMI meeting corpus. In *Proc. Int Conference on Methods and Techniques in Behavioral Research (Measuring Behavior)*, pp. 137–140, Wageningen, Netherlands, Aug. 2005.

[Melis & Dyer+ 18] G. Melis, C. Dyer, P. Blunsom: On the State of the Art of Evaluation in Neural Language Models. In *Int. Conf. on Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[Menne & Heymann+ 16] T. Menne, J. Heymann, A. Alexandridis, K. Irie, A. Zeyer, M. Kitza, P. Golik, I. Kulikov, L. Drude, R. Schlüter, H. Ney, R. Haeb-Umbach, A. Mouchtaris: The RWTH/UPB/FORTH System Combination for the 4th CHiME Challenge Evaluation. In *The 4th International Workshop on Speech Processing in Everyday Environments*, pp. 39–44, San Francisco, CA, USA, Sept. 2016.

[Merity & Keskar+ 18] S. Merity, N.S. Keskar, R. Socher: Regularizing and Optimizing LSTM Language Models. In *Int. Conf. on Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[Mikolov 12] T. Mikolov: *Statistical Language Models based on Neural Networks.* Ph.D. thesis, PhD thesis, Brno University of Technology, 2012.

[Mikolov & Joulin+ 15] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, M. Ranzato: Learning Longer Memory in Recurrent Neural Networks. In *Proc. Workshop Track, Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[Mikolov & Karafiát+ 10] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, S. Khudanpur: Recurrent neural network based language model. In *Interspeech*, pp. 1045–1048, Makuhari, Japan, Sept. 2010.

[Mikolov & Kombrink+ 11] T. Mikolov, S. Kombrink, L. Burget, J.H. Cernocky, S. Khudanpur: Extensions of recurrent neural network language model. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, Prague, Czech Republic, May 2011.

[Mikolov & Sutskever+ 13] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean: Distributed representations of words and phrases and their compositionality. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 3111–3119, Lake Tahoe, NV, USA, Dec. 2013.

[Mikolov & Zweig 12] T. Mikolov, G. Zweig: Context dependent recurrent neural network language model. In *Proc. Spoken Language Technologies (SLT)*, pp. 234–239, Miami, FL, USA, 2012.

[Miller & Giles 93] C.B. Miller, C.L. Giles: Experimental comparison of the effect of order in recurrent neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 04, pp. 849–872, 1993.

[Mnih & Teh 12] A. Mnih, Y.W. Teh: A Fast and Simple Algorithm for Training Neural Probabilistic Language Models. In *Proc. Int. Conf. on Machine Learning (ICML)*, ICML'12, pp. 419–426, Edinburgh, Scotland, 2012.

[Morishita & Oda+ 17] M. Morishita, Y. Oda, G. Neubig, K. Yoshino, K. Sudoh, S. Nakamura: An Empirical Study of Mini-Batch Creation Strategies for Neural Machine Translation. In *Proc. First Workshop on Neural Machine Translation, NMT@ACL*, pp. 61–68, Vancouver, Canada, Aug. 2017.

[Nair & Hinton 10] V. Nair, G.E. Hinton: Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. Int. Conf. on Machine Learning (ICML)*, pp. 807–814, Haifa, Israel, June 2010.

[Nakamura & Maruyama+ 90] M. Nakamura, K. Maruyama, T. Kawabata, K. Shikano: Neural network approach to word category prediction for English texts. In *Proc. Conference on Computational linguistics*, pp. 213–218, 1990.

[Nakamura & Shikano 89] M. Nakamura, K. Shikano: A study of English word category prediction based on neural networks. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 731–734, Glasglow, UK, May 1989.

[Narayanan & Prabhavalkar+ 19] A. Narayanan, R. Prabhavalkar, C.C. Chiu, D. Rybach, T.N. Sainath, T. Strohman: Recognizing long-form speech using streaming end-to-end models. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Sentosa, Singapore, Dec. 2019.

[Ney 84] H. Ney: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. *IEEE Transactions on Speech and Audio Processing*, Vol. 32, No. 2, pp. 263–271, April 1984.

[Ney & Essen 91] H. Ney, U. Essen: On smoothing techniques for bigram-based natural language modelling. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 825–828, Toronto, Canada, May 1991.

[Ney & Essen+ 94] H. Ney, U. Essen, R. Kneser: On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, Vol. 8, No. 1, pp. 1–38, 1994.

[Nix & Kim+ 19] A. Nix, Y. Kim, J. Rosendahl, S. Khadivi, H. Ney: Masked Translation Model. [Online]. : https://openreview.net/forum?id=HygaSxHYvH, 2019.

[Nolden 17] D. Nolden: *Progress in Decoding for Large Vocabulary Continuous Speech Recognition.* Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, April 2017.

[Nußbaum-Thom & Wiesler+ 10] M. Nußbaum-Thom, S. Wiesler, M. Sundermeyer, C. Plahl, S. Hahn, R. Schlüter, H. Ney: The RWTH 2009 QUAERO ASR evaluation system for English and German. In *Proc. Interspeech*, pp. 1517–1520, Makuhari, Japan, Sept. 2010.

[Oerder & Ney 93] M. Oerder, H. Ney: Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 119–122, Minneapolis, MI, USA, April 1993.

[Ortmanns & Ney+ 97] S. Ortmanns, H. Ney, X. Aubert: A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech & Language*, Vol. 11, No. 1, pp. 43–72, 1997.

[Oualil & Klakow 17] Y. Oualil, D. Klakow: A Neural Network approach for mixing language models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5710–5714, New Orleans, LA, USA, March 2017.

[Panayotov & Chen+ 15] V. Panayotov, G. Chen, D. Povey, S. Khudanpur: LibriSpeech: an ASR corpus based on public domain audio books. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, South Brisbane, Queensland, Australia, April 2015.

[Parikh & Saluja+ 14] A.P. Parikh, A. Saluja, C. Dyer, E.P. Xing: Language Modeling with Power Low Rank Ensembles. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1487–1498, Doha, Qatar, Oct. 2014.

[Parikh & Täckström+ 16] A.P. Parikh, O. Täckström, D. Das, J. Uszkoreit: A Decomposable Attention Model for Natural Language Inference. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2249–2255, Austin, TX, USA, Nov. 2016.

[Park & Chan+ 19] D.S. Park, W. Chan, Y. Zhang, C.C. Chiu, B. Zoph, E.D. Cubuk, Q.V. Le: SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proc. Interspeech*, pp. 2613–2617, Graz, Austria, Sept. 2019.

[Peters & Neumann+ 18] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer: Deep Contextualized Word Representations. In *Proc. North American Chapter of the Association for Computational Linguistics on Human Language Technologies (NAACL-HLT)*, pp. 2227–2237, New Orleans, LA, USA, June 2018.

[Povey & Cheng+ 18] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, S. Khudanpur: Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Proc. Interspeech*, pp. 3743–3747, Hyderabad, India, Sept. 2018.

[Povey & Woodland 02] D. Povey, P.C. Woodland: Minimum Phone Error and I-smoothing for improved discriminative training. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 105–108, Orlando, FL, USA, May 2002.

[Prabhavalkar & Rao+ 17] R. Prabhavalkar, K. Rao, T. Sainath, B. Li, L. Johnson, N. Jaitly: A Comparison of Sequence-to-Sequence Models for Speech Recognition. In *Proc. Interspeech*, pp. 939–943, Stockholm, Sweden, Aug. 2017.

[Rabiner 89] L.R. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.

[Rabiner 93] L. Rabiner: *Fundamentals of speech recognition*. Pearson Education India, 1993.

[Radford & Narasimhan+ 18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever: Improving language understanding by generative pre-training. [Online]. : https://blog.openai.com/language-unsupervised/, 2018.

[Radford & Wu+ 19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever: Language Models are Unsupervised Multitask Learners. [Online]. : https://blog.openai.com/better-language-models/, 2019.

[Rae & Potapenko+ 20] J.W. Rae, A. Potapenko, S.M. Jayakumar, C. Hillier, T.P. Lillicrap: Compressive Transformers for Long-Range Sequence Modelling. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[Raffel & Shazeer+ 19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu: Exploring the limits of transfer learning with a unified text-to-text transformer. Preprint arXiv:1910.10683, 2019.

[Raju & Filimonov+ 19] A. Raju, D. Filimonov, G. Tiwari, G. Lan, A. Rastrow: Scalable Multi Corpora Neural Language Models for ASR. In *Proc. Interspeech*, pp. 3910–3914, 2019.

[Rao & Sak+ 17] K. Rao, H. Sak, R. Prabhavalkar: Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, pp. 193–199, Okinawa, Japan, Dec. 2017.

[Renals & Morgan+ 94] S. Renals, N. Morgan, H. Bourlard, M. Cohen, H. Franco: Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 1, pp. 161–174, Jan 1994.

[Roark & Saraclar+ 04] B. Roark, M. Saraclar, M. Collins, M. Johnson: Discriminative Language Modeling with Conditional Random Fields and the Perceptron Algorithm. In *Proc. Association for Computational Linguistics (ACL)*, pp. 47–54, Barcelona, Spain, July 2004.

[Robinson & Fallside 91] T. Robinson, F. Fallside: A recurrent error propagation network speech recognition system. *Computer Speech & Language*, Vol. 5, No. 3, pp. 259–274, 1991.

[Rosenberg & Zhang+ 19] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. Moreno, Y. Wu, Z. Wu: Speech Recognition with Augmented Synthesized Speech. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Sentosa, Singapore, Dec. 2019.

[Rosenfeld 96] R. Rosenfeld: A Maximum Entropy Approach to Adaptive Statistical Language Modelling. *Computer Speech and Language*, Vol. 10, No. 3, pp. 187–228, 1996.

[Rousseau & Deléglise+ 14] A. Rousseau, P. Deléglise, Y. Estève: Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. In *Proc. Int. Conf. on Language Resources and Evaluation (LREC)*, pp. 3935–3939, Reykjavik, Iceland, 2014.

[Rumelhart & Hinton+ 86] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning representations by back-propagating errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[Sainath & Kingsbury+ 13] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran: Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6655–6659, Vancouver, Canada,, May 2013.

[Sak & Senior+ 14] H. Sak, A.W. Senior, F. Beaufays: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech*, pp. 338–342, Singapore, Sept. 2014.

[Salazar & Kirchhoff+ 19] J. Salazar, K. Kirchhoff, Z. Huang: Self-Attention Networks for Connectionist Temporal Classification in Speech Recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7115–7119, Brighton, UK, May 2019.

[Schlüter & Doetsch+ 16] R. Schlüter, P. Doetsch, P. Golik, M. Kitza, T. Menne, K. Irie, Z. Tüske, A. Zeyer: Automatic Speech Recognition Based on Neural Networks. In *Int. Conf. Speech and Computer*, Vol. 9811 of *Lecture Notes in Computer Science, Subseries Lecture Notes in Artificial Intelligence*, pp. 3–17, Budapest, Hungary, Aug. 2016.

[Schuster & Nakajima 12] M. Schuster, K. Nakajima: Japanese and Korean voice search. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, Kyoto, Japan, March 2012.

[Schwenk 07] H. Schwenk: Continuous space language models. *Computer Speech & Language*, Vol. 21, No. 3, pp. 492–518, 2007.

[Schwenk & Gauvain 02] H. Schwenk, J.L. Gauvain: Connectionist language modeling for large vocabulary continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 762–765, Orlando, FL, USA, 2002.

[Schwenk & Gauvain 04] H. Schwenk, J. Gauvain: Neural network language models for conversational speech recognition. In *Proc. Interspeech*, Jeju Island, Korea, Oct. 2004.

[Schwenk & Gauvain 05] H. Schwenk, J.L. Gauvain: Training Neural Network Language Models on Very Large Corpora. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 201–208, Vancouver, Canada, Oct. 2005.

[Schwenk & Rousseau+ 12] H. Schwenk, A. Rousseau, M. Attik: Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *NAACL-HLT Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 11–19, Montréal, Canada, June 2012.

[Seide & Li+ 11] F. Seide, G. Li, D. Yu: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In *Proc. Interspeech*, pp. 437–440, Florence, Italy, Aug. 2011.

[Sennrich & Haddow+ 16a] R. Sennrich, B. Haddow, A. Birch: Improving Neural Machine Translation Models with Monolingual Data. In *Proc. Association for Computational Linguistics (ACL)*, pp. 86–96, Berlin, Germany, Aug. 2016.

[Sennrich & Haddow+ 16b] R. Sennrich, B. Haddow, A. Birch: Neural Machine Translation of Rare Words with Subword Units. In *Proc. Association for Computational Linguistics (ACL)*, pp. 1715–1725, Berlin, Germany, August 2016.

[Shannon 48] C.E. Shannon: A mathematical theory of communication. *Bell system technical journal*, Vol. 27, No. 3, pp. 379–423, 1948.

[Shannon 51] C.E. Shannon: Prediction and entropy of printed English. *Bell system technical journal*, Vol. 30, No. 1, pp. 50–64, 1951.

[Shannon & Weaver 49] C.E. Shannon, W. Weaver: *The mathematical theory of communication.* University of Illinois press, 1949.

[Shareghi & Gerz+ 19] E. Shareghi, D. Gerz, I. Vulić, A. Korhonen: Show Some Love to Your n-grams: A Bit of Progress and Stronger n-gram Language Modeling Baselines. In *Proc. North American Chapter of the Association for Computational Linguistics on Human Language Technologies (NAACL-HLT)*, pp. 4113–4118, Minneapolis, MI, USA, June 2019.

[Shaw & Uszkoreit+ 18] P. Shaw, J. Uszkoreit, A. Vaswani: Self-Attention with Relative Position Representations. In *Proc. North American Chapter of the Association for Computational Linguistics on Human Language Technologies (NAACL-HLT)*, pp. 464–468, New Orleans, LA, USA, June 2018.

[Shazeer & Mirhoseini+ 17] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q.V. Le, G.E. Hinton, J. Dean: Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *Int. Conf. on Learning Representations (ICLR)*, Toulon, France, April 2017.

[Shi & Larson+ 13] Y. Shi, M. Larson, P. Wiggers, C.M. Jonker: K-Component Adaptive Recurrent Neural Network Language Models. In *Proc. Int. Conf. on Text, Speech, and Dialogue (TSD)*, pp. 311–318, Pilsen, Czech Republic, Sept. 2013.

[Soltau & Liao+ 17] H. Soltau, H. Liao, H. Sak: Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition. In *Proc. Interspeech*, pp. 3707–3711, Stockholm, Sweden, Aug. 2017.

[Sperber & Niehues+ 18] M. Sperber, J. Niehues, G. Neubig, S. Stüker, A. Waibel: Self-Attentional Acoustic Models. In *Proc. Interspeech*, pp. 3723–3727, Hyderabad, India, Sept. 2018.

[Sriram & Jun+ 18] A. Sriram, H. Jun, S. Satheesh, A. Coates: Cold Fusion: Training Seq2Seq Models Together with Language Models. In *Proc. Interspeech 2018*, pp. 387–391, Hyderabad, India, Aug. 2018.

[Srivastava & Greff+ 15a] R.K. Srivastava, K. Greff, J. Schmidhuber: Highway Networks. In *the Deep Learning workshop at Int. Conf. on Machine Learning (ICML)*, Lille, France, July 2015.

[Srivastava & Greff+ 15b] R.K. Srivastava, K. Greff, J. Schmidhuber: Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2368–2376, Montreal, Canada, Dec. 2015.

[Srivastava & Hinton+ 14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[Stahlberg & Cross+ 18] F. Stahlberg, J. Cross, V. Stoyanov: Simple Fusion: Return of the Language Model. In *Proc. Third Conference on Machine Translation (WMT)*, pp. 204–211, Brussels, Belgium, Oct. 2018.

[Stolcke 02] A. Stolcke: SRILM-an extensible language modeling toolkit. In *Proc. Interspeech*, pp. 901–904, Denver, CO, USA, 2002.

[Sundermeyer 16] M. Sundermeyer: *Improvements in Language and Translation Modeling*. Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, June 2016.

[Sundermeyer & Ney+ 15] M. Sundermeyer, H. Ney, R. Schlüter: From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 23, No. 3, pp. 517–529, March 2015.

[Sundermeyer & Oparin+ 13] M. Sundermeyer, I. Oparin, J.L. Gauvain, B. Freiberg, R. Schlüter, H. Ney: Comparison of feedforward and recurrent neural network language models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8430–8434, Vancouver, Canada, May 2013.

[Sundermeyer & Schlüter+ 11] M. Sundermeyer, R. Schlüter, H. Ney: On the Estimation of Discount Parameters for Language Model Smoothing. In *Proc. Interspeech*, pp. 1433–1436, Florence, Italy, Aug. 2011.

[Sundermeyer & Schlüter+ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM Neural Networks for Language Modeling. In *Proc. Interspeech*, pp. 194–197, Portland, OR, USA, Sept. 2012.

[Sundermeyer & Schlüter+ 14] M. Sundermeyer, R. Schlüter, H. Ney: rwthlm The RWTH Aachen University Neural Network Language Modeling Toolkit. In *Proc. Interspeech*, pp. 2093–2097, Singapore, Sept. 2014.

[Sundermeyer & Tüske+ 14] M. Sundermeyer, Z. Tüske, R. Schlüter, H. Ney: Lattice Decoding and Rescoring with Long-Span Neural Network Language Models. In *Proc. Interspeech*, pp. 661–665, Singapore, Sept. 2014.

[Sutskever & Vinyals+ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to Sequence Learning with Neural Networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 3104–3112, Montréal, Canada, Dec. 2014.

[Synnaeve & Xu+ 19] G. Synnaeve, Q. Xu, J. Kahn, E. Grave, T. Likhomanenko, V. Pratap, A. Sriram, V. Liptchinsky, R. Collobert: End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. Preprint arXiv:1911.08460, 2019.

[Tani & Nolfi 99] J. Tani, S. Nolfi: Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, Vol. 12, No. 7, pp. 1131–1141, 1999.

[Ter-Sarkisov & Schwenk+ 15] A. Ter-Sarkisov, H. Schwenk, L. Barrault, F. Bougares: Incremental adaptation strategies for neural network language models. In *Proc. Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pp. 48–56, Beijing, China, July 2015.

[Tillmann & Ney 97] C. Tillmann, H. Ney: Word Triggers and the EM Algorithm. In *Proc. Special Interest Group Workshop on Computational Natural Language Learning (ACL)*, pp. 117–124, Madrid, Spain, July 1997.

[Tjandra & Sakti+ 17] A. Tjandra, S. Sakti, S. Nakamura: Listening while speaking: Speech chain by deep learning. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, pp. 301–308, Okinawa, Japan, Dec. 2017.

[Toshniwal & Kannan+ 18] S. Toshniwal, A. Kannan, C.C. Chiu, Y. Wu, T.N. Sainath, K. Livescu: A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*, Athens, Greece, Dec. 2018.

[Tran & Bisazza+ 16] K. Tran, A. Bisazza, C. Monz: Recurrent Memory Network for Language Modeling. In *Proc. North American Chap. of the Assoc. for Comput. Ling. on Human Lang. Tech. (NAACL-HLT)*, pp. 321–331, San Diego, CA, USA, June 2016.

[Tüske & Irie+ 16] Z. Tüske, K. Irie, R. Schlüter, H. Ney: Investigation on log-linear interpolation of multi-domain neural network language model. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, March 2016.

[Tüske & Michel+ 17] Z. Tüske, W. Michel, R. Schlüter, H. Ney: Parallel Neural Network Features for Improved Tandem Acoustic Modeling. In *Proc. Interspeech*, Stockholm, Sweden, Aug. 2017.

[Tüske & Schlüter+ 13] Z. Tüske, R. Schlüter, H. Ney: Multilingual Hierarchical MRASTA Features for ASR. In *Proc. Interspeech*, pp. 2222–2226, Lyon, France, Aug. 2013.

[Tüske & Schlüter+ 18] Z. Tüske, R. Schlüter, H. Ney: Investigation on LSTM Recurrent N-gram Language Models for Speech Recognition. In *Proc. Interspeech*, pp. 3358–3362, Hyderabad, India, Sept. 2018.

[van Aken & Winter+ 19] B. van Aken, B. Winter, A. Lser, F.A. Gers: How Does BERT Answer Questions? In *Proc. ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1823–1832, Beijing, China, Nov. 2019.

[Vaswani & Shazeer+ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin: Attention is All you Need. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008. Long Beach, CA, USA, Dec. 2017.

[Vieting 19] P. Vieting: AMI System. Unpublished work, 2019.

[Waibel & Hanazawa+ 89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K.J. Lang: Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 3, pp. 328–339, 1989.

[Wang & Cho 16] T. Wang, K. Cho: Larger-Context Language Modelling with Recurrent Neural Network. In *Proc. Association for Computational Linguistics (ACL)*, pp. 1319–1319, Berlin, Germany, Aug. 2016.

[Wang & Mohamed+ 19] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang et al.: Transformer-based acoustic modeling for hybrid speech recognition. Preprint arXiv:1910.09799, 2019.

[Wang & Zhao+ 20] B. Wang, D. Zhao, C. Lioma, Q. Li, P. Zhang, J.G. Simonsen: Encoding word order in complex embeddings. In *Int. Conf. on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[Watanabe & Hori+ 17] S. Watanabe, T. Hori, J. Le Roux, J.R. Hershey: Student-Teacher Network Learning with Enhanced Features. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5275–5278, New Orleans, LA, USA, March 2017.

[Weiss & Chorowski+ 17] R.J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, Z. Chen: Sequence-to-Sequence Models Can Directly Translate Foreign Speech. In *Proc. Interspeech*, pp. 2625–2629, Stockholm, Sweden, Aug. 2017.

[Weng & Stolcke+ 98] F. Weng, A. Stolcke, A. Sankar: Efficient lattice representation and generation. In *Int. Conf. on Spoken Language Processing (ICSLP)*, Sydney, Australia, Nov. 1998.

[Wessel & Schluter+ 01] F. Wessel, R. Schluter, H. Ney: Explicit word error minimization using word hypothesis posterior probabilities. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 33–36, Salt Lake City, UT, USA, May 2001.

[Wong & Gales 16] J.H. Wong, M.J. Gales: Sequence Student-Teacher Training of Deep Neural Networks. In *Proc. Interspeech*, pp. 2761–2765, San Francisco, CA, USA, Sept. 2016.

[Xiong & Droppo+ 17] W. Xiong, J. Droppo, X. Huang, F. Seide, M.L. Seltzer, A. Stolcke, D. Yu, G. Zweig: Toward human parity in conversational speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, Vol. 25, No. 12, pp. 2410–2423, 2017.

[Xiong & Wu+ 18] W. Xiong, L. Wu, J. Zhang, A. Stolcke: Session-level Language Modeling for Conversational Speech. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2764–2768, Brussels, Belgium, Oct.-Nov. 2018.

[Yang & Dai+ 18] Z. Yang, Z. Dai, R. Salakhutdinov, W.W. Cohen: Breaking the softmax bottleneck: A high-rank RNN language model. In *Int. Conf. on Learning Representations (ICLR)*, Vancouver, Canada, April 2018.

[Yao & Cohn+ 15] K. Yao, T. Cohn, K. Vylomova, K. Duh, C. Dyer: Depth-gated LSTM. Presented at Jelinek Summer Workshop, Preprint arXiv:1508.03790, Aug. 2015.

[You & Su+ 19] Z. You, D. Su, D. Yu: Teach an All-rounder with Experts in Different Domains. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6425–6429, Brighton, UK, May 2019.

[Young 92] S.J. Young: The General Use of Tying in Phoneme Based HMM Recognizers. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 1, pp. 569–572, San Francisco, CA, USA, March 1992.

[Yu & Deng+ 10] D. Yu, L. Deng, G.E. Dahl: Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition. In *NIPS 2010 workshop on Deep Learning and Unsupervised Feature Learning*, Vancouver, Canada, Dec. 2010.

[Yu & Deng+ 13] D. Yu, L. Deng, F. Seide: The deep tensor neural network with applications to large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 2, pp. 388–396, 2013.

[Yu & Deng 16] D. Yu, L. Deng: *Automatic Speech Recognition: A Deep Learning Approach.* Springer, 2016.

[Zaremba & Sutskever+ 14] W. Zaremba, I. Sutskever, O. Vinyals: Recurrent neural network regularization. Preprint arXiv:1409.2329, 2014.

[Zeghidour & Xu+ 18] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, R. Collobert: Fully Convolutional Speech Recognition. Preprint arXiv:1812.06864, 2018.

[Zeyer & Alkhouli+ 18] A. Zeyer, T. Alkhouli, H. Ney: RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition. In *Proc. Assoc. for Computational Linguistics (ACL)*, Melbourne, Australia, July 2018.

[Zeyer & Bahar+ 19] A. Zeyer, P. Bahar, K. Irie, R. Schlter, H. Ney: A comparison of Transformer and LSTM encoder decoder models for ASR. In *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Sentosa, Singapore, Dec. 2019.

[Zeyer & Doetsch+ 17] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlter, H. Ney: A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2462–2466, New Orleans, LA, USA, March 2017.

[Zeyer & Irie+ 18] A. Zeyer, K. Irie, R. Schlüter, H. Ney: Improved training of end-to-end attention models for speech recognition. In *Proc. Interspeech*, pp. 7–11, Hyderabad, India, Sept. 2018.

[Zhang & Chen+ 16] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, J. Glass: Highway Long Short-Term Memory RNNs for Distant Speech Recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5755–5759, Shanghai, China, March 2016.

[Zhang & Dauphin+ 19] H. Zhang, Y.N. Dauphin, T. Ma: Residual Learning Without Normalization via Better Initialization. In *Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[Zhang & Jiang+ 15] S. Zhang, H. Jiang, M. Xu, J. Hou, L. Dai: The Fixed-Size Ordinally-Forgetting Encoding Method for Neural Network Language Models. In *Proc. Association for Computational Linguistics (ACL)*, pp. 495–500, Beijing, China, July 2015.

[Zhang & Jiang+ 16] S. Zhang, H. Jiang, S. Xiong, S. Wei, L. Dai: Compact Feedforward Sequential Memory Networks for Large Vocabulary Continuous Speech Recognition. In *Proc. Interspeech*, pp. 3389–3393, San Francisco, CA, USA, Sept. 2016.

[Zhang & Liu+ 17] S. Zhang, C. Liu, H. Jiang, S. Wei, L. Dai, Y. Hu: Nonrecurrent Neural Structure for Long-Term Dependence. *IEEE/ACM Transactions Audio, Speech & Language Processing*, Vol. 25, No. 4, pp. 871–884, 2017.

[Zhang & Wu+ 16] J. Zhang, X. Wu, A. Way, Q. Liu: Fast Gated Neural Domain Adaptation: Language Model as a Case Study. In *Proc. Int. Conf. on Computational Linguistics (COLING)*, pp. 1386–1397, Osaka, Japan, Dec. 2016.

[Zhou & Michel+ 20] W. Zhou, W. Michel, K. Irie, M. Kitza, R. Schlüter, H. Ney: The RWTH ASR System for TED-LIUM Release 2: Improving Hybrid-HMM with SpecAugment. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, to appear, Barcelona, Spain, May 2020.