# Language Modeling and Machine Translation: Improvements in Training and Modeling

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

**M.Sc. Communications Engineering**
**Yingbo Gao**
aus Dalian

Berichter: Universitätsprofessor Dr.-Ing. Hermann Ney
Professor Dr. François Yvon

Tag der mündlichen Prüfung: 28. August 2024

Intelligence is a balance between counting and smoothing.

# Acknowledgments

My deepest and sincerest gratitude goes to Sen. Prof. Dr. Ing. Hermann Ney, who gave me the opportunity to go deeper into the field of statistical learning, and provided me invaluable advice and guidance along the way. His genuine interest in the problems, profound knowledge in the field and great dedication to the science form a reservoir, from which I am privileged to be able to draw strength in the rest of my life.

I would like to thank Prof. François Yvon, who kindly agreed to be the second examiner of this dissertation. I am grateful for the exchanges we had at conferences and seminars, and I am humbled to have someone with such sparkling interests and deep insights to review my work.

In addition, I would like to thank the various sponsors and projects that funded my studies, without which this dissertation could not have come together: ERC SEQCLAS, DFG CoreTec, BMBF NeuroSys, BMG HYKIST, and Apptek GmbH.

The five years I spent at Informatik 6 at RWTH is a period of my life that I will always treasure, and this is due to the great studying and working environment enabled by my lovely colleagues: Christian Herold, Weiyue Wang, Yunsu Kim, David Thulke, Jan Rosendahl, Zijian Yang, Tamer Alkhouli, Parnia Bahar, Andreas Guta, Julian Schamper, Markus Freitag, Jörn Wübker, Kazuki Irie, Eugen Beck, Mohammad Zeineldeen, Wilfried Michel, Albert Zeyer, Markus Kitza, Nick Rossenbach, Peter Vieting, Wei Zhou, Tina Raissi, Simon Berger, Benedikt Hilmes, Tobias Menne, Stefan Koltermann, Thomas Dackweiler, Stephanie Jansen, Christian Dugast, Volker Steinbiß and Ralf Schlüter.

I would also like to express my gratitude towards the bright and hard-working students that I was lucky enough to have the opportunity to work with during this period: Baohao Liao, Alexander Gerstenberger, Khoa Viet Tran, Jingjing Huo, Felix Schmidt, David Südholt, Malcolm Burdorf, Weiran Yang.

Also, hats off to the colleagues at Apptek and eBay, with whom I grew a lot as a researcher and from whom I learned a lot about the business world: Evgeny Matusov, Pavel Golik, Jintao Jiang, Javier Jorge, Patrick Wilken, Sarah Barenek, Zoltán Tüske, Patrick Dötsch, Harald Hanselmann, Steve Cook, Mudar Yaghi; Shahram Khadivi, Daniel Stein, Nicola Ueffing, Christian Plahl, Pavel Petrushkov, Leonard Dahlmann, Michael Kozielski, José G. C. de Souza.

Of course there are more names and people whose names I never got to learn that I fail to list here, but this does not diminish my gratitude to them. Be it a casual discussion during a conference, a kind word of encouragement, a direct criticism, a helping hand to clean the working environment, etc., I am thankful for all who accompanied me on this journey.

Finally, to my closest friends and family, from the bottom of my heart, I have mixed feelings of being sorry and thankful: sorry that I am selfish in pursuing what I want and not spending more time with you, and thankful that despite my selfishness, you were always there for me.

# Abstract

The field of statistical language modeling and machine translation has seen rapid developments in recent years, with artificial neural networks taking center of the stage, dominating the modeling approaches. To this day, despite numerous efforts from a theoretical standpoint, the training strategies associated with artificial neural networks are largely empirical and can be roughly described as "recipes polished through generations of researchers". In this dissertation, we consider and try to solve three problems related to the core training and modeling approaches in neural language modeling and machine translation.

First, we study the computational inefficiencies during both training and testing that are related to having large vocabularies for neural language models. We revisit the traditional training criteria, and highlight that the inefficiencies originate from having to traverse over the large vocabulary. As a solution, sampling-based training criteria were proposed. We describe how the model optimums are attained, and show that with a correction step, the performance of different sampling-based training criteria can be similar. On top of that, we propose a new self-normalized importance sampling training criterion, which performs on par with the state-of-the-art noise contrastive estimation approach.

Second, we turn our attention to smoothing in neural machine translation. As in all statistical models, we want to generalize well to unseen events after training. Because of the exponential growth of the number of possible events with increasing sequence lengths, smoothing is an important topic to study. Specifically, we study label smoothing, input smoothing and multi-agent mutual learning. We compare training settings and propose new algorithms to enable better regularized training. We show through extensive experiments, that the translation performance of a strong baseline model can still be improved if one does a good job in smoothing.

Third, we question the long-standing paradigm of using encoder-decoder models to do machine translation. Historically, the separation of the encoder and the decoder is a natural design choice because the source and the target sentence are two sequences in different languages, and it makes sense to explicitly encode the source information first before decoding from it. However, it is also possible to concatenate the source and target sentences together, and train a language model on the concatenated sequences. This alternative language modeling approach has the benefit of dropping the decoder and maintaining an encoder-only monolithic architecture. Experimental results suggest that this method can perform on par with state-of-the-art encoder-decoder models, and also to some degree explains the translation capabilities of recent large language models.

# Kurzfassung

Der Bereich der statistischen Sprachmodellierung und maschinellen Übersetzung hat in den letzten Jahren rasante Entwicklungen erlebt, wobei künstliche neuronale Netze im Mittelpunkt stehen und die Modellierungsansätze dominieren. Trotz zahlreicher Bemühungen aus theoretischer Sicht sind die mit künstlichen neuronalen Netzen verbundenen Trainingsstrategien bis heute weitgehend empirisch und können gewissermaßen als "Rezepte, die über Generationen von Forschern ausgefeilt wurden" beschrieben werden. In dieser Dissertation untersuchen wir drei Probleme im Zusammenhang mit den zentralen Trainings- und Modellierungsansätzen in der neuronalen Sprachmodellierung und maschinellen Übersetzung.

Zunächst untersuchen wir die rechnerischen Ineffizienzen sowohl beim Training als auch beim Testen, die mit dem großen Vokabular für neuronale Sprachmodelle zusammenhängen. Wir greifen die traditionellen Trainingskriterien erneut auf und heben hervor, dass die Ineffizienzen darauf zurückzuführen sind, dass das große Vokabular durchlaufen werden muss. Zu diesem Zweck wurden stichprobenbasierte Trainingskriterien vorgeschlagen. Wir leiten ab, wo die Modelloptima erreicht werden, und begründen, dass mit einem Korrekturschritt die Leistung verschiedener stichprobenbasierter Trainingskriterien nahezu identisch ist. Darüber hinaus schlagen wir ein neues selbstnormalisiertes Importance-Sampling-Trainingskriterium vor, dessen Leistung mit dem Ansatz der prominenten Noise Contrastive Estimation gleichwertig ist.

Zweitens richten wir unsere Aufmerksamkeit auf die Glättung in der neuronalen maschinellen Übersetzung. Aufgrund des exponentiellen Wachstums der Anzahl möglicher Sequenze mit zunehmender Sequenzlänge ist die Glättung ein wichtiges Untersuchungsthema, da wir nach dem Training gut auf unsichtbare Ereignisse verallgemeinern möchten. Insbesondere untersuchen wir label smoothing, input smoothing und multi-agent mutual learning. Wir variieren Trainingseinstellungen und schlagen neue Algorithmen vor, um ein besser reguliertes Training zu ermöglichen, und zeigen durch umfangreiche Experimente, dass die Übersetzungsleistung eines starken Basismodells noch verbessert werden kann, wenn man bei der Glättung gute Arbeit leistet.

Drittens stellen wir die seit langem bestehende Paradigma in Frage, encoder-decoder Modelle für die maschinelle Übersetzung zu verwenden. Historisch gesehen war die Trennung von Encoder und Decoder eine natürliche Entwurfsentscheidung, da der Quell- und der Zielsatz zwei Sequenzen in unterschiedlichen Sprachen sind und es sinnvoll ist, die Quellinformationen zuerst explizit zu kodieren, bevor man daraus dekodiert. Es ist jedoch auch möglich, die Quell- und Zielsätze miteinander zu verbinden und ein Sprachmodell auf den verketteten Sequenzen zu trainieren. Dieser alternative Sprachmodellierungsansatz hat den Vorteil, dass der Decoder weggelassen und eine monolithische Architektur nur für den Encoder beibehalten werden kann.

Experimentelle Ergebnisse deuten darauf hin, dass diese Methode mit dem modernen Encoder-Decoder-Modell mithalten kann und in gewissem Maße auch die Übersetzungsfähigkeiten neuerer großer Sprachmodelle erklärt.

# Contents

# 1. Introduction

The probabilistic interpretation of language is the foundation of recent successes in speech and natural language processing. The notion of the "probability of a sentence" may be controversial [Norvig 12] in terms of understanding what language truly is, but it is at the heart of modern automatic speech recognition and statistical machine translation applications.

Broadly speaking, the steps to build a probabilistic model for human language applications can be summarized as follows:

1. Define some performance measure for the task at hand, e.g. translation edit rate for machine translation.
2. Gather some test corpora in the desired domain, and strictly keep them away from the optimization procedure.
3. Gather as much training data as possible, preferably sufficiently similar to the test corpora.
4. Define and initialize a probabilistic model of enough modeling capacity, the parameterization of which can be an artificial neural network.
5. Optimize the model to learn from the empirical distribution of the training corpora, but be wary of overfitting and aim for generalization.
6. Use Bayes decision rule to generate hypotheses for test inputs, and evaluate these model outputs according to the performance measure.

Carefully planning and executing the steps above is a challenging task, but brilliant minds have helped to push human language technology to an impressive level with the statistical approach. The goal of this chapter is to first expand on the topic of the statistical interpretation of natural language from a modern perspective. Then, the attention is turned towards the statistical approach to two important applications, language modeling and machine translation, discuss state-of-the-art neural-network-based models, and touch upon issues related to training and modeling, which are the key questions related to this dissertation.

## 1.1 On The Statistical Interpretation of Natural Language

### 1.1.1 The Controversy

Today, the "probability of a sentence", whether it is in the context of a language modeling task, and or a conditioned sentence generation task like machine translation, is a widely accepted

terminology. However, it was not always the case. It fact, if one reflects on the internal thought process when formulating a sentence, one can probably agree that people do not start with some artificial "begin-of-sentence" word, ask themselves "what is the probability of the next word?", and end the sentence with an artificial "end-of-sentence" word. More likely, one thinks about the subject, the object, and the verb, and formulate the sentence to be semantically and syntactically correct. A similar view is shared by many, and one of the most prominent examples is from Noam Chomsky, who is considered to be a founding figure of modern linguistics, and he famously wrote [Chomsky 69]:

> *But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.*

Prior to that, he gave two made-up example sentences in his influential book "Syntactic Structures" [Chomsky 57]:

> *Neither (a) "colorless green ideas sleep furiously" nor (b) "furiously sleep ideas green colorless", nor any of their parts, has ever occurred in the past linguistic experience of an English speaker. But (a) is grammatical, while (b) is not.*

This line of thought pinpoints the one of the difficulties to model natural language, that a pure statistical approach may neglect important linguistic phenomena.

### 1.1.2 The Modern View

Nowadays, the everyday life is greatly changed because of the convenience that natural language technologies bring. The discussions about controversies around the statistical approach are less heated. However, it makes sense to revisit those thoughts from a modern perspective.

In fact, Chomsky's view against a statistical approach is well-motivated from a linguistic point of view, and very tempting indeed. However, one strong counter-evidence is right there in the argument itself. That is, after he wrote those sentences, both (a) and (b) in the previous example have been published and re-published, and now they have occurred in the linguistic experience of many English speakers. While it is still true that no matter how many times the two sentences are published, (a) is still grammatical, and (b) is not, the observation that they do exist in the English literature now highlights an important property of statistical models of natural language:

> *The statistical models of natural language try to model the information, and not directly the linguistic phenomena in the natural language.*

In this context, natural language is indeed the subject we care about. However, unlike linguistics which try to describe language focusing on its grammatical, cultural, historical, psychological, etc. aspects, statistical models try to describe the information in the language. In other words, no matter how unnatural a sentence may be, be it "furiously sleep ideas green colorless" or some bizarre phrase from the Internet, with probabilistic models, we want to express how much information there is in the sentence and make use of that quantity[1].

For instance, a statistical language model trained in the year of 1950, i.e. before the example sentence "colorless green ideas sleep furiously" was ever introduced, would assign a much smaller probability mass to the sentence compared to a language model trained today. That tells us

[1] Some typical use cases are, performing beam search, rescoring hypotheses, filtering corpus, etc.

something - the sentence is as grammatically wrong today as it was decades ago, but the great contributions from Chomsky and the decades-long challenging of our understanding of what language really is does not go unnoticed by the probabilistic language model. In fact, the domain of information in which a probabilistic sequence model captures is no longer confined in the natural language, but greatly expanded to many fields, e.g. modeling DNA [Benegas & Batra$^+$ 23], molecules [Flam-Shepherd & Zhu$^+$ 22], audio signals [van den Oord & Dieleman$^+$ 16], images [Dosovitskiy & Beyer$^+$ 21], etc.

Additionally, if we take a broader view of the topic, one can even say that the probabilistic modeling of sequential data is core to artificial intelligence. This is because that we as humans discover, communicate and preserve information via language. Here, language is a general term, which can be mathematical notations, sheet music, chemical formulas etc. Considering that we can digitize the information, store and share it (think of the bit streams transmitted through the Internet at every second), it is safe to say that human knowledge can be represented in a sequential format. The author takes the position that successful modeling of such sequential data is key to the understanding of human knowledge and the development of artificial intelligence. To this end, two straightforward examples can be listed:

1. In entropy coding, the underlying probabilistic model determines the higher bound of compression ratio. That is, when plugging in a much stronger probabilistic model, which performs the "simple" task of predicting the next word given the previous words, one can achieve much higher compression ratio [Bellard 23]. In fact, it is not hard to show that the highest compression ratio is achieved if the true distribution can be plugged in.

2. As an intuitive example, consider a probabilistic language model which assigns a 99% probability to the word "2" after seeing "1 + 1 =". It is safe to say that this model contains more knowledge about counting than any of our ancestors who did not know anything about counting. In fact, while evaluating large language models is intrinsically hard, this is how recent large language models are typically evaluated, i.e. against pre-curated knowlege benchmarks that are presented to the model in the "1 + 1 =" manner [Gao & Tow$^+$ 21, Hugging Face H4 23].

Coming back to natural language, the discussion here about the statistical interpretation lays the foundation of this thesis. Establishing that the statistical natural language models model the information rather than directly the linguistics phenomena, we can be relieved from the stress to having to address specific linguistic issues, and instead we can more focus on the training and modeling issues that are central to this thesis. In other words, we respect the data as is, and focus on the statistical learning and modeling aspects.

## 1.2 Statistical Language Modeling

### 1.2.1 Basic Concepts

Statistical language modeling is the task of assigning a probability to a sequence of discrete symbols. Here, "discrete symbols" can mean words, subwords, characters, etc., and refer to the atomic level at which the statistical models operate. Similarly, "a sequence of discrete symbols" can mean a sentence, sentences, a document, etc. Below, for simplicity, we will refer to such a discrete symbol as a "word".

Denoting a word with $w$, the sequence length with $M$, and word position indices in $1, 2, ..., M$, a language model can be expressed as:

$$p(w_1^M) \tag{1.1}$$

Before going into details of how this quantity can be modeled, it helps to think about the size of the probbility space. Assume the vocabulary size is $V$, that is, a word $w_m$ at some position $m$ can choose from $V$ different values (e.g. dog, cat, etc.). Because at each positions we have $V$ many choices, and all these choices are independent (possibly leading to unnatural sentences but we do not care for this calculation), the total possible number of token sequences with length $M$ is then the product of the number of choices at each position, i.e. exponential in $M$:

$$V^M \tag{1.2}$$

Because the space is quite large, the word choices take discrete values, and we want the probability space to be normalized (summing up to one for all possible word sequences), to have an intuitive understanding of the task, a suitable metaphor can be the following: think of many small boxes and we need to pour a fixed amount of dirt into these boxes somehow, and then each language model we develop corresponds to a certain way of pouring dirt.

It is common to factor this probability in an autoregressive manner, and operate in the log space for numerical stability:

$$\log p(w_1^M) = \log \left\{ p(w_0) \prod_{m=1}^{M} p(w_m | w_0^{m-1}) \right\} \tag{1.3}$$

$$= \log p(w_0) + \sum_{m=1}^{M} \log p(w_m | w_0^{m-1}) \tag{1.4}$$

In such a decomposition, $w_0$ is defined to be an artificial start symbol, prepended to the actual word sequence, e.g. `<bos>` or `<s>`, which always has a probability of one:

$$p(w_0) := 1 \tag{1.5}$$

In order to ensure normalization over all possible sequences, an artificial end symbol $w_M$, e.g. `<eos>` or `</s>`, is appended to the end of the actual word sequence, which takes away some probability mass to end the sequence at the corresponding length:

$$0 \leq p(w_M | w_0^{M-1}) \leq 1 \tag{1.6}$$

The quantity $p(w_m | w_0^{m-1})$ in Eq.1.3 is often referred to as the conditional probability of target word $w_m$ given context (or history) $w_0^{m-1}$. When the conditional dependence is limited to a certain number of predecessor words, we say the context is limited. Correspondingly, when there is no truncation in the conditional dependence, we say the context is unlimited. This distinction between limited and unlimited context naturally leads to different designs of language models.

In the case of limited context, traditionally, the task of language modeling is approached with $n$-gram count-based language models [Jelinek & Bahl$^+$ 75, Jelinek & Mercer 80, Katz 87, Church & Gale 91, Ney & Essen 91, Kneser & Ney 95, Chen & Goodman 96, Chen & Goodman 99]. In such models, the conditional dependence follows an $(n-1)$-order Markov assumption:

$$p(w_m | w_0^{m-1}) := p(w_m | w_{\max(0,m-n+1)}^{m-1}) \tag{1.7}$$

Because here the event space is again exponential in the context length, the training data is unavoidably sparse, which calls for smoothing methods (to assign non-zero probability masses to unseen events) on the empirical counts. A widely adopted smoothing method is an improvement from absolute discounting [Ney & Essen 91] called Kneser-Ney smoothing [Kneser & Ney 95], which assigns discounted probability masses to lower-order events in proportion to the diversity of the context.

With the continuous development of artificial neural networks [Werbos 74, Schmidhuber 92, Hinton 09, CireAan & Meier$^+$ 12, Martinez & Bengio$^+$ 13, Pascanu & Mikolov$^+$ 13, Krizhevsky & Sutskever$^+$ 17], popularization of word vector concepts [Baker & McCallum 98, Bengio & Ducharme$^+$ 03, Mikolov & Chen$^+$ 13, Mikolov & Sutskever$^+$ 13, Grave & Bojanowski$^+$ 18, Mikolov & Grave$^+$ 18], as well as the introduction of more and more efficient training methods and advances in hardware and software [Hinton & Osindero$^+$ 06, Nickolls & Buck$^+$ 08, LeCun & Bottou$^+$ 12, Al-Rfou & Alain$^+$ 16, Abadi & Agarwal$^+$ 15, Paszke & Gross$^+$ 19, Li & Zhao$^+$ 20], modern language models with limited contexts have shifted away from traditional count-based models to neural-network-based models. Common examples include feedforward neural language models [Xu & Rudnicky 00, Schwenk & Gauvain 02, Bengio & Ducharme$^+$ 03] and more recently, transformer [Vaswani & Shazeer$^+$ 17] neural language models [Liu & Saleh$^+$ 18a, Radford & Narasimhan 18, Al-Rfou & Choe$^+$ 19, Radford & Wu$^+$ 19a, Irie & Zeyer$^+$ 19].

In the case of language models with unlimited context, recurrent-neural-network-based models are adopted, a prominent example of which is the long short-term memory neural language model [Hochreiter & Schmidhuber 97, Kim & Jernite$^+$ 16, Sundermeyer & Schlüter$^+$ 12, Liu & Cao$^+$ 18, Merity & Keskar$^+$ 18, Herold & Gao$^+$ 18]. Such models build upon feedforward language models, and keep essential components such as word vectors and log-likelihood training. The difference is that they further employ the long short-term memory neural network module to replace the feedforward layers, and use the recurrent feedback to model the potentially long-range autoregressiveness in the sequence of words.

The language modeling task discussed in this dissertation finds its roots in early studies in information theory [Shannon 48, Shannon 51], estimating probability masses for unseen events in large spaces [Good 53, Gale & Sampson 95], and early developments of automatic speech recognition [Jelinek & Bahl$^+$ 75, Jelinek 76]. This is to be differentiated from recent works on masked language modeling [Devlin & Chang$^+$ 19], where the task is to learn useful hidden representations for natural language sequences by training on a cloze task, rather than directly discriminating word sequences against one another.

### 1.2.2 Evaluation Metrics

There are two common metrics to evaluate a language model, namely, perplexity [Jelinek & Mercer$^+$ 77] (directly) and word error rate (indirectly).

The perplexity (PPL) of a language model $p(w_0^M)$ is defined to be:

$$\mathrm{PPL}\big\{p(w_0^M)\big\} = \exp\Big\{-\frac{1}{M}\sum_{m=1}^{M}\log p(w_m|w_0^{m-1})\Big\} \tag{1.8}$$

Notice that the log probability of the artificial start symbol is dropped because it is always zero. In practical applications, we often append an artificial end symbol to the word sequence, and treat it as a regular word when calculating the perplexity. Say the vocabulary size of a language model is $V$, and the perplexity of the language model on some test set is $V'$, the perplexity can also be interpreted as the "effective vocabulary size" [Bahl & Jelinek$^+$ 83]. This

is because:

$$\exp\big(-\frac{1}{M}\sum_{m=1}^{M}\log\frac{1}{V'}\big)=V' \tag{1.9}$$

In other words, if we plug in an artificial model that assigns uniform probabilities in a vocabulary of size $V'$ in each position, we will obtain a model with the same perplexity, $V'$. Therefore, intuitively, the perplexity measures on average how confused or perplexed the model is in each position. For example, a naive uniform model, i.e. guessing at random, should give a perplexity of $V$ (also a good way to judge the quality of a random initialization), and we typically prefer a model that gives lower perplexity on the validation data (it explains the data well and generalizes better).

Another important property of PPL is that we use the empirical distribution of some test corpus to calculate it, this can be seen by rewriting Eq. 1.8:

$$\begin{aligned}\mathrm{PPL}\big\{p(w_0^M)\big\}&=\exp\big\{-\frac{1}{M}\sum_{m=1}^{M}\log p(w_m|w_0^{m-1})\big\}\\&=\exp\big\{-\sum_{x,c}\frac{N_{x,c}}{N}\log p(c|x)\big\}\\&=\exp\big\{-\sum_{x}\frac{N_x}{N}\sum_{c}\frac{N_{x,c}}{N_x}\log p(c|x)\big\}\\&=\exp\big\{-\sum_{x}pr(x)\sum_{c}pr(c|x)\log p(c|x)\big\}\end{aligned} \tag{1.10}$$

Because we already use $m$ in $M$ to denote position indices, we use $N$ with a subscript to denote the count of an event specified in the subscript, although $M=N$. In other words, here, $N_x$ and $N_{x,c}$ denote the counts of seeing some "previous words context" $x$ and "context and next word pair" $x, c$. Additionally, with this rewrite, the model distribution of target word $c$ given context $x$ becomes $p(c|x)$. Here, $pr$ denotes the empirical distribution of some event in the test corpus. We do these rewrites because it clearly exhibits the point, that PPL is defined on the empirical distribution of some test corpus. This is important to notice, because if the test corpus is not sufficiently large or representative, its empirical distribution may be far from the true distribution we really care about. For example, in a real-world application, the name of a company (a certain $c$) may be important given some context (a certain $x$), but if a certain test set has an empirical distribution $pr(c|x)$ that assigns zero probability to the event, i.e. $pr(\text{company\_name}|\text{our company is called})=0$, the PPL reported on this test will not reflect how accurate the model performs in such cases. While this point is introduced in the context of language modeling, it is true for machine translation as well, where the difference is that the context is expanded to further include the source sentence. In other words, one needs to be careful about the test corpus when reporting PPL, and make sure that interesting events are sufficiently represented in the test corpus.

The word error rate (WER) of an automatic speech recognition model is defined to be:

$$\mathrm{WER}=\frac{\min\Big(\#(\text{insertions})+\#(\text{deletions})+\#(\text{substitutions})\Big)}{\#(\text{reference words})}\cdot 100\% \tag{1.11}$$

Here, the numerator refers to the minimum number of edit operations between the hypothesis sentence and the reference sentence, i.e. the Levenshtein distance [Levenshtein et al. 66],

and can be calculated efficiently with dynamic programming. The total edit count is then normalized by the reference transcription length. In the field of automatic speech recognition, it is also well known that there is a roughly log-linear empirical correlation between word error rate and perplexity [Bahl & Jelinek$^+$ 83, Chen & Beeferman$^+$ 98, Klakow & Peters 02, Sundermeyer & Ney$^+$ 15a, Irie 20]. That is, regardless of the language model (be it count-based or neural-network-based), plugging in a stronger language model (with a lower perplexity) during search, typically results in lower word error rates.

## 1.3 Statistical Machine Translation

### 1.3.1 Basic Concepts

Similar to statistical language modeling, the task of statistical machine translation also requires a model that assigns a probability to a target sequence of words, given some source sequence of words. Again, "words" here mean the atomic level at which the statistical machine translation system operates, e.g. words, subwords, characters, etc., and "sequences" can be at the sentence or document level. Below, without any loss of generality, we use "source sentences" and "target sentences" to refer to such sequences.

Denoting source words as $f$, source word position indices in $1, 2, ..., J$, target words as $e$, target word position indices in $1, 2, ..., I$, a parallel sentence pair can be expressed as:

$$\text{source sentence}: \quad f_1^J = f_1, f_2, ..., f_J \tag{1.12}$$

$$\text{target sentence}: \quad e_1^I = e_1, e_2, ..., e_I \tag{1.13}$$

As in statistical language modeling, in practical systems, we often prepend an artificial start-of-sentence symbol and append an artificial end-of-sentence symbol to the source and target sentences.

Our goal with machine translation is to find the best translation hypothesis or candidate in the target language given some source sentence. Using Bayes decision rule, this can be expressed as:

$$f_1^J \rightarrow \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I}{\text{argmax}} \big\{ \Pr(e_1^I | f_1^J) \big\} \tag{1.14}$$

Here, $\Pr(e_1^I | f_1^J)$ denotes the true posterior probability, and because we do not know what it is, we plug in our statistical model $p(e_1^I | f_1^J)$ trained on the training data to approximate it.

Depending on how $p(e_1^I | f_1^J)$ is defined, historically, statistical machine translation models can be broadly categorized into three types: noisy-channel modeling, log-linear modeling, and discriminative modeling.

The noisy-channel modeling variant [Shannon 48, Brown & Cocke$^+$ 90] assumes that the target sentence $e_1^I$ is generated by transmitting the source sentence $f_1^J$ through some noisy channel $p(f_1^J | e_1^I)$:

$$\text{argmax} \big\{ \Pr(e_1^I | f_1^J) \big\} = \underset{I, e_1^I}{\text{argmax}} \big\{ \frac{\Pr(f_1^J | e_1^I) \cdot \Pr(e_1^I)}{\Pr(f_1^J)} \big\} \tag{1.15}$$

$$= \underset{I, e_1^I}{\text{argmax}} \big\{ \Pr(f_1^J | e_1^I) \cdot \Pr(e_1^I) \big\} \tag{1.16}$$

$$= \underset{I, e_1^I}{\text{argmax}} \big\{ p(f_1^J | e_1^I) \cdot p(e_1^I) \big\} \tag{1.17}$$

The derivation involves applying the Bayes' rule and dropping the denominator $\Pr(f_1^J)$, which is independent of $I$ and $e_1^I$. As can be seen, a target-side language model $p(e_1^I)$ and an inverse translation model $p(f_1^J|e_1^I)$ are needed, the latter of which is commonly decomposed into a length model, an alignment model and a lexicon model in traditional IBM models [Brown & Cocke$^+$ 88, Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93] and the hidden Markov model [Vogel & Ney$^+$ 96].

The log-linear modeling variant [Papineni & Roukos$^+$ 98, Och & Ney 02] allows various feature functions $h_m(f_1^J, e_1^I)$ to be used. These feature functions depend on the sentence pair $(f_1^J, e_1^I)$, and give different scores regarding different aspects of the translation:

$$\operatorname*{argmax}\big\{\Pr(e_1^I|f_1^J)\big\} = \operatorname*{argmax}_{I,e_1^I}\big\{p(e_1^I|f_1^J)\big\} \tag{1.18}$$

$$= \operatorname*{argmax}_{I,e_1^I}\Big\{\frac{\exp\Big(\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\Big)}{\sum_{I',e'^{I'}_1}\exp\Big(\sum_{m=1}^M \lambda_m h_m(f_1^J, e'^{I'}_1)\Big)}\Big\} \tag{1.19}$$

$$= \operatorname*{argmax}_{I,e_1^I}\Big\{\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\Big\} \tag{1.20}$$

Above, $\lambda_m$ is the weight for each feature function and $M$ is the total number of features. Here, the feature scores come from various models, such as a phrased-based model [Och & Tillmann$^+$ 99, Zens & Och$^+$ 02, Koehn & Och$^+$ 03], a word translation model [Brown & Cocke$^+$ 88, Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93, Vogel & Ney$^+$ 96], a language model [Kneser & Ney 95, Chen & Goodman 96] and a reordering model [Tillmann 04, Galley & Manning 08, Cherry & Moore$^+$ 12, Wübker 17, Guta 20]. They are combined in a log-linear fashion and normalized over all possible target sentences across all lengths. During search, the normalization can be dropped because there is no dependence on $I$ and $e_1^I$ in the denominator. Log-linear modeling is a generalization from noisy-channel modeling, and it is possible to include many more features, as mentioned above [Chiang & Knight$^+$ 09, Hopkins & May 11, Green & Wang$^+$ 13]. The learning of the weights $\lambda_m$ can be done with minimum error rate training [Och 03]. Note that log-linear modeling is a general method of model combination, and is included here as a separate variant because of the historical development of machine translation. There is no strict limitation on what can be used as a feature function, and naturally, neural machine translation models can also be combined in a similar fashion [Stahlberg & Cross$^+$ 18].

The discriminative modeling variant [Kalchbrenner & Blunsom 13, Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17] has been the focus of development for machine translation in the last decade. The central idea is to encode the source sentence information into some hidden space and decode from that space depending on which target position we are in, and directly output the conditional probability given the context of that position. While non-autoregressive models [Gu & Bradbury$^+$ 18] do exist, the most common way to model the posterior distribution is through autoregressive factorization of the sequence

probability:

$$\operatorname{argmax}\left\{\Pr(e_1^I|f_1^J)\right\} = \operatorname*{argmax}_{I,e_1^I}\left\{\prod_{i=1}^{I}\Pr(e_i|e_0^{i-1},f_1^J)\right\} \tag{1.21}$$

$$= \operatorname*{argmax}_{I,e_1^I}\left\{\sum_{i=1}^{I}\log\Pr(e_i|e_0^{i-1},f_1^J)\right\} \tag{1.22}$$

$$= \operatorname*{argmax}_{I,e_1^I}\left\{\sum_{i=1}^{I}\log p(e_i|e_0^{i-1},f_1^J)\right\} \tag{1.23}$$

The parameterization of such approaches is primarily done with artificial neural networks. Because there is no involvement of a reverse translation model or log-linear combination, as in classic statistical machine translation, and the prediction of each target word is directly achieved with the same neural network, such models are conceptually simpler and are sometimes referred to as end-to-end or sequence-to-sequence models. It is worth mentioning that because the search space complexity is exponential in $I$, we trim down the search space to a fixed size for each target position $i$ and perform beam search when carrying out the argmax in practice.

### 1.3.2 Evaluation Metrics

To decide if a machine translation system is doing well is not an easy task: the annual Conference on Machine Translation[2] has a dedicated shared task on evaluation metrics every year [Barrault & Biesialska+ 20, Akhbardeh & Arkhangorodsky+ 21]. This is partly due to the high cost associated with obtaining high-quality reference translations, and also because of the word-reordering nature of translation. Among others, the bilingual evaluation understudy score [Papineni & Roukos+ 02] and translation edit rate [Snover & Dorr+ 06] are two commonly used metrics to judge the quality of machine translation hypotheses against reference translations.

The bilingual evaluation understudy (BLEU) score of a hypothesis $\hat{e}_1^{\hat{I}}$ against a reference translation $e_1^I$ is defined to be:

$$\textsc{Bleu}(\hat{e}_1^{\hat{I}}, e_1^I) = \mathrm{BP}\cdot\exp\left\{\sum_{n=1}^{N} w_n \log\left(\mathrm{prec}_n(\hat{e}_1^{\hat{I}}, e_1^I)\right)\right\} \tag{1.24}$$

$$= \min(1,\exp(1-\frac{I}{\hat{I}}))\cdot\exp\left\{\sum_{n=1}^{N} w_n \log\frac{\sum_{\hat{e}^{(n)}\in\hat{e}_1^{\hat{I}}}\min\left(c(\hat{e}^{(n)};\hat{e}_1^{\hat{I}}), c(\hat{e}^{(n)};e_1^I)\right)}{\sum_{\hat{e}'^{(n)}\in\hat{e}_1^{\hat{I}}} c(\hat{e}'^{(n)};\hat{e}_1^{\hat{I}})}\right\} \tag{1.25}$$

Above, BP is short for "brevity penalty", which is one (no penalty) for hypotheses longer than the reference and less than one (penalized) for shorter hypotheses. The second term at the right-hand side of the equation is the geometric mean of the clipped $n$-gram accuracies $\mathrm{prec}_n(\hat{e}_1^{\hat{I}}, e_1^I)$. Here, $n$ is the order of $n$-grams being counted up to $N$, $w_n$ is the weight for each order, $\hat{e}^{(n)}$ or $\hat{e}'^{(n)}$ refers to a unique $n$-gram in the hypothesis $\hat{e}_1^{\hat{I}}$, and $c(\hat{e}^{(n)};\cdot)$ is the count of that $n$-gram in the corresponding target sentence. BLEU can be unreliable when considered on a single reference sentence [Papineni & Roukos+ 02], therefore in practice:

[2]Previously "Workshop on Machine Translation".

- We gather the count statistics over an entire test corpus before calculating the clipped $n$-gram precision.
- When multiple reference translations exist for one source sentence, we plug in the reference sentence length $I$ into BP using the reference sentence that has the closest length to the hypothesis sentence $\hat{e}^{\hat{I}}$, and plug in the clipped count using the reference sentence with which the most $n$-gram hits are found [Qin & Specia 15].

The translation edit rate (TER) of a machine translation system is defined to be:

$$\text{TER} = \frac{\min\left(\#(\text{insertions}) + \#(\text{deletions}) + \#(\text{substitutions}) + \#(\text{shifts})\right)}{\#(\text{reference words})} \cdot 100\% \quad (1.26)$$

Here, the numerator is the minimum number of edit operations required to change a hypothesis $\hat{e}_1^{\hat{I}}$ into the reference $e_1^I$, and the denominator is the reference sentence length $I$. If multiple references are available for a source sentence, the numerator becomes the minimum edit operations into any one of the references, and the denominator is replaced by the average reference length. Similar to BLEU, when the test set consists of multiple source sentences, i.e. at the corpus level, the count statistics for each sentence are gathered first and averaged later. Compared to the word error rate metric introduced in Section 1.2.2, TER additionally allows shift operations. The shift operation allows for the move of a contiguous word sequence to another position in the hypothesis, and the cost is the same as the other three types of edits regardless of the length of the moved word sequence or the distance of the move. Searching for the optimal shift edits is very costly, and in practice a greedy search is done to determine the shift edits, while dynamic programming is still used to decide the Levenshtein edits. Note that TER is not bounded under 100%, and a simple example to think of is when the reference is only ten words, while the hypothesis is 1000 words long.

## 1.4 State-of-the-Art Models

### 1.4.1 Neural Language Modeling

Nowadays, the state-of-the-art language modeling is primarily done with artificial neural networks. While different neural language models may differ somewhat in their neural network architecture design, the essential elements of these models are similar:

- Batched autoregressive training in parallel: given a word sequence $w_1^M$, the input to the network is $w_0, w_1, ..., w_{M-1}$ and the target output of the network is $w_1, w_2, ..., w_M$. That is, for target position $m$, the network is presented with input $w_1^{m-1}$ and is tasked to predict $w_m$. Processing along the time (i.e. word positions $1, 2, ..., M$) dimension is done in parallel during training. Sentences are batched (several sentences fed to the network together), bucketed (sentences of similar lengths grouped together to reduce the number of paddings), and padded (artificial pad symbols `<pad>` appended to short sentences until sentence lengths are equal).
- Word embedding into continuous-valued hidden space: in order for the network to perform arithmetic operations on the words, a word embedding module is included to project the raw word indices to continuous-valued word vectors [Mikolov & Chen$^+$ 13, Mikolov & Sutskever$^+$ 13]. The word embedding module is a large $V \times D$ matrix, where $V$ is the

vocabulary size and $D$ is the hidden dimension, and the process of embedding a word is essentially a row-wise lookup in the matrix.

- Mixing context to aggregate information: a hidden module is then used to mix the context information from different word positions to obtain the context vector. The choice of the module can be different, e.g. long short-term memory networks [Schmidhuber 92, Sundermeyer & Schlüter$^+$ 12] or transformer encoders [Vaswani & Shazeer$^+$ 17, Irie & Zeyer$^+$ 19], but its purpose is the same - to enrich the word vectors with contextual word information from other positions. This step is also where the modeling capabilities of different neural language models differ the most, e.g. the long short-term memory network is able to make use of very long context, but has an information bottleneck along the time dimension (information from far-away positions gets weaker), while transformer networks attend to a fixed number of previous positions (although commonly up to the beginning of the sentence, effectively using unlimited context), and has a constant lookup cost for different positions (the model decides the weight for far-away positions with attention and there is no information bottleneck along the time dimension). It is common to use the same dimension size $D$ for word vectors and context vectors, while it is also possible to bottleneck the context vector dimension to reduce parameter count [Gerstenberger & Irie$^+$ 20].
- Projection of the context vector into vocabulary size and softmax activation on logits: the $D$-dimensional context vector for each position is then projected up to the vocabulary size $V$ dimensions to obtain the logits. This is done via a linear projection matrix in $D \times V$, which is possibly tied with the embedding matrix [Press & Wolf 17]. The training criterion for neural language models is most often cross entropy, which requires normalized outputs. Therefore, a softmax function is commonly used to activate the logits and produce the conditional probabilities $p(w_m|w_0^{m-1})$.

Large neural language models [Devlin & Chang$^+$ 19, Shoeybi & Patwary$^+$ 19, Brown & Mann$^+$ 20, BigScience Workshop 22, Zhang & Roller$^+$ 22, Schulman & Zoph$^+$ 22], which contain billions of parameters and are trained on hundreds of billions of words, are the most recent additions to state-of-the-art language modeling (not necessarily autoregressive). They often make use of some word segmentation algorithm [Sennrich & Haddow$^+$ 16a, Kudo & Richardson 18] to operate in a subword vocabulary (often applied on the byte-level to avoid out-of-vocabulary characters) to mitigate the problem of a large vocabulary, while word-level language models are still relevant for automatic speech recognition [Wang & Mohamed$^+$ 20, Lüscher & Beck$^+$ 19a]. Language modeling is a fundamental task in statistical learning of human language, or even human knowledge, and as it stands today, neural networks are unarguably the best weapon in our arsenal to tackle the problem.

### 1.4.2 Neural Machine Translation

For neural machine translation, artificial neural networks are also widely applied. The key components of the state-of-the-art machine translation models are:

- Batched autoregressive training in parallel at the target side: similar to that of language modeling, the generation of target sentence $e_1^I$ is done by inputting the entire source sentence $f_1^J$ and the shifted-by-one-position target sentence $e_0^{I-1}$ to the model. Batching, bucketing and padding are done in a way similar to language modeling. Note that while autoregressive training is done on the target side, it is not necessarily the case on the

source side. For instance, it is possible to apply auxiliary losses such as length prediction loss [Yang & Gao+ 20] and source reconstruction loss [Gao & Herold+ 22a] on the source side.

- Word embedding and positional encoding into continuous-valued hidden space: both source tokens and target tokens are mapped to continuous-valued hidden spaces (possibly shared [Press & Wolf 17, Vaswani & Shazeer+ 17]), and positional information of different tokens is also encoded (either implicitly included by recurrence in recurrent neural networks [Sutskever & Vinyals+ 14, Bahdanau & Cho+ 15] or explicitly included using a positional encoding module [Vaswani & Shazeer+ 17, Shaw & Uszkoreit+ 18, Rosendahl & Tran+ 19]). Here, the modeling unit is typically at the subword level, e.g. byte-pair encoding units [Sennrich & Haddow+ 16b] or sentencepiece [Kudo & Richardson 18] units, to avoid the problem of large vocabulary and deliver better generalization on unseen words.

- Encoding of source information via a dedicated source encoder: the information in the entire source sentence is summarized by a so-called encoder into a sequence of source hidden vectors [Kalchbrenner & Blunsom 13, Sutskever & Vinyals+ 14, Cho & van Merriënboer+ 14b, Cho & van Merriënboer+ 14a, Bahdanau & Cho+ 15, Vaswani & Shazeer+ 17]. Because all source words are available at test time, it is possible and reasonable to "look at" all source positions, i.e. past, current and future for each position, when summarizing information (this is also called using "bi-directional" context because one can traverse the sequence from left to right or from right to left). Commonly, the result of this step is a sequence of hidden vectors of the same length as the original source sentence, with the hidden vector at each position having summarized the source contextual information of that position.

- Mixing context by attending to source representations via a dedicated target decoder: for each target position, the target context information is aggregated up to that position, and the attention mechanism [Bahdanau & Cho+ 15, Luong & Pham+ 15, Niu & Zhong+ 21] is used to query source-side hidden vectors with the target context. This step is key in solving the alignment problem in translation, i.e. building a mapping between different source and target positions. Commonly, a weighted summation over the relevant (the "relevance" of each source position is decided by the attention mechanism) source hidden vectors is done, and combined with the target context to generate the context vector for final projection into the target vocabulary.

- Projection and activation of context vector followed by beam search: like in language modeling, the $D$-dimensional context vector is finally projected up to $V$ dimensions and activated via a softmax function to obtain the posterior distribution over the target vocabulary. The $D \times V$ projection matrix is often shared with the target embedding matrix, and a three-way tying is possible when source and target vocabulary is further shared [Press & Wolf 17, Vaswani & Shazeer+ 17, Sennrich & Haddow+ 16b]. The training criterion is commonly cross entropy with label smoothing [Szegedy & Vanhoucke+ 16, Pereyra & Tucker+ 17, Gao & Wang+ 20]. During test time, beam search [Koehn & Knowles 17, Freitag & Al-Onaizan 17, Koehn 20, Liang & Wang+ 22] is applied to trim down the otherwise exponential search space into a fixed-sized window at each target position, and length normalization [Wu & Schuster+ 16, Yang & Huang+ 18, Provilkov

& Malinin 21] is used to combat the tendency of neural models to generate short and sometimes even empty sentences [Koehn & Knowles 17, Shi & Xiao+ 20].

Since the introduction of the transformer model [Vaswani & Shazeer+ 17], the translation quality of neural machine translation has improved significantly, with reported similar-to-human performance on certain test sets when considering automatic metrics [Hassan & Aue+ 18]. That said, there are still active research areas, such as document-level translation [Kim & Tran+ 19, Huo & Herold+ 20], multilingual training [Johnson & Schuster+ 17, Aharoni & Johnson+ 19, Lin & Wu+ 21], and speech translation [Kano & Sakti+ 17, Bérard & Besacier+ 18, Anastasopoulos & Chiang 18, Inaguma & Duh+ 19, Bahar & Zeyer+ 19, Wang & Wu+ 20, Bahar & Bieschke+ 21, Tran & Thulke+ 22]. Another interesting trend is that some very large pre-trained models have exhibited capabilities to do few-shot or even zero-shot translations [Raffel & Shazeer+ 22, Radford & Kim+ 22, Schulman & Zoph+ 22], and can even work as strong automatic evaluators of the performances of machine translation systems [Kocmi & Federmann 23], posing questions about the potential of further making use of monolingual data beyond model combination [Stahlberg & Cross+ 18] and back-translation [Sennrich & Haddow+ 16a, Edunov & Ott+ 18, Graça & Kim+ 19].

## 1.5 Issues Related to Training and Modeling

Having discussed the basic concepts of language modeling and machine translation, as well as state-of-the-art neural-network-based approaches to both tasks, in this section, a brief walk through of issues related to training and modeling relevant for this dissertation is provided.

### 1.5.1 Large Vocabulary

The first issue is the potentially very large vocabulary associated with neural language models (and automatic speech recognition and machine translation models). Zipf's law [Zipf 49, Powers 98, Herold & Gao+ 18] is a well-known empirical law that describes the long-tail distribution of word frequencies found in natural languages. To build a practical word-level language model, it is necessary to include those low-frequency words in the tail. However, this would result in large vocabularies, which could go up to the order of several hundred thousands or even a million for large datasets and commercial systems [Chelba & Mikolov+ 13, Wang & Mohamed+ 19, Lüscher & Beck+ 19a, Gao & Thulke+ 21, Yang & Gao+ 22]. While subword-level [Sennrich & Haddow+ 16b, Kudo & Richardson 18, Radford & Wu+ 19b, Brown & Mann+ 20, Irie & Zeyer+ 19, Zeyer & Bahar+ 19, Zhou & Zeineldeen+ 21, Meyer & Michel+ 22, Deng & Hsiao+ 22] and character-level models [Hwang & Sung 16, Chung & Cho+ 16, Lee & Cho+ 17, Hakimi Parizi & Cook 18, Al-Rfou & Choe+ 19, Banar & Daelemans+ 21, Deng & Hsiao+ 22] do exist, word-level models are still relevant, especially in automatic speech recognition systems based on hybrid deep neural networks and hidden Markov models [Lüscher & Beck+ 19a, Gao & Thulke+ 21, Yang & Gao+ 22].

The large vocabulary in itself does not directly imply any significant changes in the neural architectures, meaning that the softmax-based output projection can still be intact, but it poses a challenge in the parameter count and training/testing efficiencies. More specifically, on one hand, too large of a parameter count can lead to out-of-memory issues, and one has to take measures such as reducing the batch size, applying various parallelism tricks, etc. to counter it. On the other hand, the training and testing speed of a neural language model might be critical in certain applications, e.g. we probably do not want to spend the majority of decoding budget

for an automatic speech recognition system on language model forward passes, and thus is also important to control.

The root causes of these potential issues are:

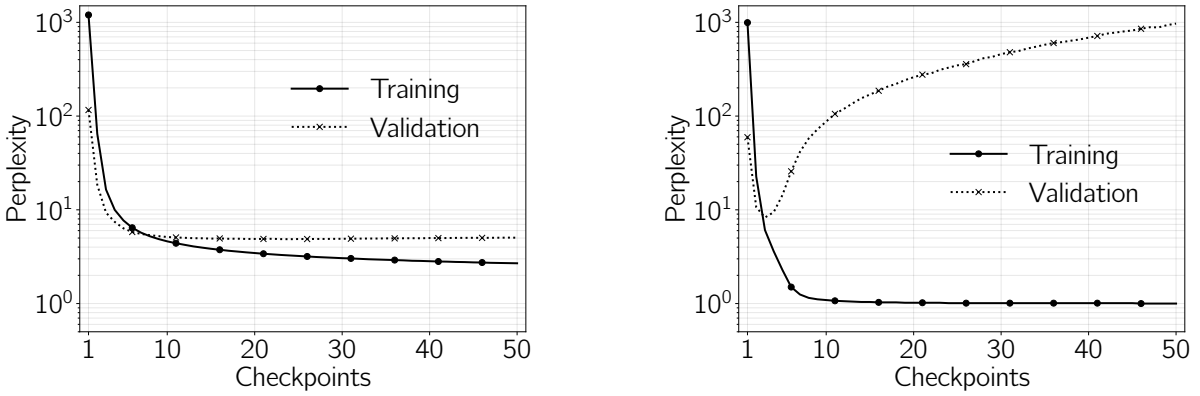
1. The word embedding and output projection matrices often make up the lion's share of the total model parameter count, and scale linearly with the vocabulary size $V$ [Chen & Si$^+$ 18, Gao & Liao$^+$ 20].

2. The softmax function includes a traversal over the entire vocabulary in the summation in the denominator (denoting the logit of model on a certain word $v$ with $q_v$):

$$\text{softmax}(q_v) = \frac{\exp q_v}{\sum_{v'}^{V} \exp q_{v'}} \tag{1.27}$$

In this dissertation, the issue of large vocabulary in language modeling is addressed in Chapter 3 and is approached using sampling-based training criteria, the essential idea of which is to avoid the traversal over the entire vocabulary $V$ and achieve self-normalization during training. We show both theoretically and experimentally, that when sampling-based training criteria are properly applied, significant speedups can be achieved both in training and testing of neural language models with large vocabularies.

### 1.5.2 Overfitting




(a) with regularization

(b) without regularization

Figure 1.1: Training and validation curves of two training runs on the IWSLT2014 German-English Translation Task, with and without regularization. When no regularization is applied, the model overfits on the training data (the training perplexity gets very close to one) and the validation perplexity quickly diverges.

A second issue with the neural-network-based models nowadays is that they are typically over-parameterized. For instance, in the commonly used IWSLT2014 German-English translation dataset, the total number of training subword units is less than 8 million (counting both source and target sides, see Table A.4), while baseline transformer models typically have around 37 million trainable parameters. In other words, it is not uncommon to train models whose number of parameters is even more than the number of tokens in the training corpus. This is an interesting observation because the capacity of the neural model is adequate to even "remember"

the entire training data, but doing that gives no guarantee of generalization performance. As an example, in Figure 1.1, two training runs are performed on this dataset. In one case (Figure 1.1a), all regularization tricks are kept. In the other case (Figure 1.1b), all regularization tricks are disabled. One quickly notices the divergent behavior of the validation perplexity in the latter case, and the training perplexity also drops down to very close to one, getting near the theoretical lower bound given by the training empirical distribution. Granted, the IWSLT2014 German-English dataset is a very small one by today's standards, but when the training corpus becomes larger, the training corpus typically also scales larger [Ott & Edunov$^+$ 18, Devlin & Chang$^+$ 19, Brown & Mann$^+$ 20, Hoffmann & Borgeaud$^+$ 22]. This type of overfitting phenomenon is commonly seen in general neural network training, which greatly hinders the generalization performance of the models. Therefore, it calls for regularization methods during the training process.

There are many methods that can regularize the model to avoid overfitting. To list a few, training smaller models, applying dropout [Srivastava & Hinton$^+$ 14a], introducing layer normalization [Ba & Kiros$^+$ 16], using weight decay [Krogh & Hertz 91], etc. While the actual methods differ in details, there are several axes along which the regularization are typically designed. For instance, it is possible: to constrain the model representational capabilities, to augment and introduce noises to the data, to have multiple models jointly make the prediction, to control the optimization process against numerical outliers, etc.

Specifically in this work, for regularization, we consider label smoothing, input smoothing, and multi-agent mutual learning. Label smoothing [Szegedy & Liu$^+$ 15, Szegedy & Vanhoucke$^+$ 16], deliberately takes away some probability mass from the target word and redistribute the probability mass to other words according to some helper model. Input smoothing [Gao & Liao$^+$ 20] follows a similar concept, but operates at the input side of the network, smoothing out the input one-hot distributions over the words, which can also be viewed as a data augmentation methods. Multi-agent mutual learning [Liao & Gao$^+$ 20] is an approach where several agents are optimized for our primary goal of translation together, but are regularized with the motivation that agents should learn among themselves. In Chapter 4, we go into details of these methods, and show via extensive experiments that stronger machine translation performance can be achieved when the smoothing set up is carefully tuned.

### 1.5.3 Model Compactness

The third issue lies in the compactness of neural network architecture designs. When modeling sequence-to-sequence tasks like machine translation, the encoder-decoder architecture has been and is still the de facto standard. For instance, in early work on the recurrent continuous translation model [Kalchbrenner & Blunsom 13], the authors already distinguish their convolutional sentence model (encoder) and their recurrent language model (decoder), which is conditioned on the former. Likewise, the encoder and decoder separation is also seen in the state-of-the-art transformer [Vaswani & Shazeer$^+$ 17] models. At a high level, the job of the encoder is to encode source information into some hidden space, either into one continuous-valued vector [Kalchbrenner & Blunsom 13, Cho & van Merriënboer$^+$ 14b, Sutskever & Vinyals$^+$ 14] or into a sequence of vectors [Bahdanau & Cho$^+$ 15], which is then used for decoding given the target-side context. This process is straightforward and intuitive, and models very well the translation process of "read the source sentence, keep a memory of what it is, translate word-by-word from that memory". However, when humans perform the translation task, it is also possible to go through a process which can be described as "scan through the source sentence and what is translated, and decide which next target word should follow". When the

author does translations, often times it is a combination of the two, i.e. translate from the memory, but also occasionally scan the entire source sentence and the partial translation when deciding the next word. There is no reason to argue that either one is the correct way of doing translation, and the author knows no work in psychology or linguistics that clearly measures which would deliver a better translation result. That said, from the viewpoint of building a more compact and monolithic model, the field has clearly favored the former and separated the encoding and decoding steps.

In Chapter 5, detailed discussion and experiments are provided to further argue that it is possible to train an encoder-only, more monolithic language model for translation which performs on par with the usual encoder-decoder models, posing questions on the necessity of the strict separation. Considering the recent advancements of large language models, the architectures of which typically only consist of a single stack of transformer encoder layers, this question on the necessity to strictly separate the encoder and decoder is especially interesting. While the "machine translation with large language models" literature more often focus issues such as few-shot/zero-shot capabilities, prompting strategies, stylized and customized translation [Hendy & Abdelrehim$^+$ 23, Yao & Jiang$^+$ 23, Zhang & Haddow$^+$ 23, Ghazvininejad & Gonen$^+$ 23, Moslem & Haque$^+$ 23], this work is motivated mainly from an architecture point-of-view. That is, in this work, the training corpus and optimization parameters are mostly kept the same as in traditional encoder-decoder modeling, while in the large language modeling literature, additional training methods such as extensive unsupervised pre-training, instruction fine-tuning and reinforcement learning with human feedback are typically included.

# 2. Scientific Goals

The goal of this thesis revolves around efficient and effective training and modeling techniques for neural-network-based language modeling and machine translation:

1. How to address the large vocabulary issue (Section 1.5.1) for neural language modeling with **sampling-based training criteria** (Chapter 3)?
   - Which sampling-based training criterion variants are there? When are the theoretical optimums attained?
   - How to encourage self-normalization of neural sequence models with sampling-based training criterion?
   - What are the speedups and model performances when training with sampling-based training criteria?
2. How does one overcome the overfitting issue (Section 1.5.2) in neural machine translation with **smoothing techniques** (Chapter 4)?
   - What is a good way to smooth model outputs to regularize training? What about model inputs?
   - Do improvements from input smoothing and output smoothing stack?
   - What extensions to model combination help translation?
3. How can the machine translation model compactness issue (Section 1.5.3) be addressed with **a language modeling approach** (Chapter 5)?
   - What is a good training objective?
   - How does a monolithic encoder-only language model compare to a discriminative encoder-decoder model for translation, from a theoretical point of view?
   - Does the encoder-only language modeling approach perform as well as the baseline encoder-decoder model? Does the conclusion hold under extensive setups?

In the process of approaching the research questions above, the author found it increasingly difficult to implement ideas and concepts in existing software. Therefore, the author wrote some custom software, "`aseq`: a sequence toolkit". The toolkit is aimed to be flat, lightweight, and easily extendable, and focuses on the tasks of language modeling and machine translation (Chapter 6).

# 3. Sampling-Based Training Criteria for Neural Language Modeling

When training neural language models, the vocabulary is one of the first things that needs to be decided. As mentioned in Section 1.5.1, in order to account for those low-frequency words found in the tail of the Zipfian distribution [Powers 98] in natural languages, the vocabulary of modern word-level natural language models is becoming larger and larger. Although subword-level and character-level systems are around [Zeyer & Bahar$^+$ 19, Al-Rfou & Choe$^+$ 19], in this chapter, we focus on the word-level models, because they are still relevant in many automatic speech recognition systems that are based on hybrid deep neural networks and hidden Markov models [Lüscher & Beck$^+$ 19a, Gao & Thulke$^+$ 21, Yang & Gao$^+$ 22].

To highlight the problem at hand, we note that for large datasets and commercial systems, the order of magnitude of the vocabulary size $V$ can go up to several hundred thousands or even a million [Chelba & Mikolov$^+$ 13, Wang & Mohamed$^+$ 19, Lüscher & Beck$^+$ 19a, Gao & Thulke$^+$ 21, Yang & Gao$^+$ 22]. This has implications in terms of both space and time complexity:

- On one hand, the size of the $D \times V$ projection matrix (with $D$ being the hidden dimension of the model) in neural language models (as well as the $V \times D$ embedding matrix if the two are shared [Press & Wolf 17], see Section 1.4.1) scales linearly with the vocabulary size $V$.
- On the other hand, during the commonly used softmax-activated cross-entropy training, the traversal over $V$ is needed in order to calculate the denominator in the softmax function for normalization at each position, which leads to the training time ranging from days to weeks.

To resolve the space complexity issue, it is possible to apply methods based on e.g. factorization [Denton & Zaremba$^+$ 14, Yu & Liu$^+$ 17, Chen & Si$^+$ 18], pruning [Han & Pool$^+$ 15, Frankle & Carbin 19, Brix & Bahar$^+$ 20], and quantization [Hubara & Courbariaux$^+$ 17, Xu & Wang$^+$ 18, Choi & El-Khamy$^+$ 20] to reduce the model size, about which we will not go into details in this dissertation. To resolve the time complexity issue, many methods are proposed, of which, the most successful branch is arguably sampling-based methods, which is going to be the focus of this chapter.

## 3.1 Related Work

One classic idea to reduce the computational cost associated with the softmax calculation when $V$ is large, is to replace the "pick one word out of $V$ words" process with a "pick one

category out of $V'$ ($V' \ll V$) categories several times" process. This changes the one-time flat decision into a series of compact decisions in a hierarchical structure (a tree structure where nodes are decision points, branches lead to other decision points, and leaf nodes correspond to the actual word choices). Notably, the original softmax can be thought of as such a tree with one root node directly linked to $V$ leaf nodes, with a time complexity of $\mathcal{O}\{1 \cdot V\}$ (in total one decision to make, $V$ terms to sum over for normalization for that one decision). Intuitively, for trees with more depth and an average number of branches $b$ at each node, the time complexity changes to $\mathcal{O}\{\log_b(V) \cdot b\}$ (in total $\log_b(V)$ decisions to make, $b$ terms to sum over for normalization for each decision). One quickly realizes that there is an optimum choice of $b$ (or more accurately, the tree structure, because $b$ is only used here for a sloppy derivation), because in the other boundary case when $b$ approaches one, i.e. asking "is it this word?" $(V-1)$ times, the complexity is back up to $\mathcal{O}\{V\}$. Because of this, related work mainly differs in how the tree is formulated.

In the literature, the above-mentioned method is referred to as "hierarchical softmax". The concept was initially proposed by Morin and Bengio [Morin & Bengio 05], where a binary hierarchical clustering with prior knowledge from the WordNet semantic hierarchy [Miller 98] is used. They also propose to share the parameters across the hierarchy, which is conveniently possible because of the binary structure. Realizing that building the tree from prior knowledge and enforcing binary decisions at each node is not optimal, Mnih and Hinton [Mnih & Hinton 08] studied automatic feature-based algorithms to construct the tree from data, and explored the effects on training time and perplexity. When the depth of the tree is limited to two, i.e. asking "in which word category is the next word?" and "which word exactly in that category?", the method sometimes goes under the name of "word classes". For instance, this has roots back in Goodman's early work in maximum entropy training [Goodman 01], is seen in the early work on recurrent neural network language models by Tomáš Mikolov et al. [Mikolov & Kombrink$^+$ 11b], and the concept is further studied by Botros et al. [Botros & Irie$^+$ 15]. Another angle to build the tree is from the perspective of information entropy and entropy coding [Shannon 48, Huffman 52]. For example, in the study of the compositionality of word embeddings by Tomáš Mikolov et al. [Mikolov & Sutskever$^+$ 13], the authors used an underlying binary Huffman tree, and for the task of machine translation, Chitnis and DeNero [Chitnis & DeNero 15] also explored different variants of variable-length codes when constructing the tree structure.

Another branch of ideas is sampling, which is the focus of this chapter. The idea is straightforward: that is, if going over the entire vocabulary is costly, why not go over only part of the vocabulary in each training iteration? This at least will have the benefit of speeding up the training. If we are lucky, in the sense that if we also achieve self-normalization after training, it is possible to further speed up testing, because in that case we can simply take the raw output of the model without explicitly doing further normalization, and because of the self-normalization property of the model, the raw outputs are directly useful for downstream tasks, e.g. in second-pass lattice rescoring for automatic speech recognition.

In the literature, there exist plenty of works that develop upon the idea above. For instance, in the same year that Bengio et al. [Bengio & Ducharme$^+$ 03] proposed the early prototype of modern neural language models, Bengio and Senecal [Bengio & Senecal 03] proposed to apply sampling in order to speed up the training. In their work, the authors noted that we want to avoid directly sampling from the model's posterior distribution because that is as costly as performing the full summation over the vocabulary. Instead, they proposed to apply the Metropolis-Hastings algorithm [Metropolis & Rosenbluth$^+$ 53, Hastings 70] and importance sampling to speed up the training procedure. However, as will be shown later in this chapter,

because the model is not directly self-normalized, in the original paper, Bengio and Senecal further proposed to apply an adaptive sampling size during training, which increases as the training proceeds, in order to achieve good training accuracies at the end of the training. Similarly, in the work by Morin and Bengio [Morin & Bengio 05] where hierarchical softmax was proposed, the authors also noted that a training speedup is achievable with importance sampling, but the testing time complexity is still $\mathcal{O}\{V\}$ because one still has to go over the entire vocabulary for proper normalization. Another popular sampling-based method is proposed by Gutmann and Hyvärinen [Gutmann & Hyvärinen 10, Gutmann & Hyvärinen 12] and called noise contrastive estimation. The core concept of this method is to rewrite the original "one out of $V$" task into a new binary classification task of telling true samples from noisy ones. The method was quickly adapted to the task of neural language modeling by Mnih and Teh [Mnih & Teh 12]. As will be shown later in this chapter, noise contrastive estimation has the desired property of self-normalization, and is widely explored in the literature. For example, in the work by Jozefowicz et al. [Jozefowicz & Vinyals$^+$ 16], the authors discussed the intrinsic relationship between importance sampling and noise contrastive estimation. In Zoph et al. [Zoph & Vaswani$^+$ 16], the authors proposed an extension to noise contrastive estimation to achieve a further speedup. Later, Goldberger and Melamud [Goldberger & Melamud 18a] derived the self-normalization property of noise contrastive estimation, and proposed a regularization variant of it where self-normalization is further explicitly encouraged. In van den Oord et al. [van den Oord & Li$^+$ 18], the authors proposed the "InfoNCE" loss for the unsupervised learning of useful representations on high-dimensional data. This further motivated a line of research which maximizes mutual information for general representation learning [Poole & Ozair$^+$ 19, Tschannen & Djolonga$^+$ 20, Kong & de Masson d'Autume$^+$ 20]. Another important work that applied the sampling concept is by Mikolov et al. [Mikolov & Chen$^+$ 13]. In this paper, the authors proposed negative sampling. Because the goal there is to obtain meaningful hidden word representations and not the normalized conditional next-word probabilities, the authors simplified the importance sampling and noise contrastive estimation approaches, and kept only the positive log probability term on the target word and a sampled subset of rival word terms in the training criterion. The noise distribution from which the samples are drawn is a smoothed version of the empirical unigram distribution and, compared to noise contrastive estimation, only the sample indices are needed, and the sample probabilities in the noise distribution are dropped. This concept of contrasting positive examples against random negative samples was also explored in earlier work by Collobert and Weston [Collobert & Weston 08], where they proposed a maximum-margin hinge loss in order to rank the target words on top.

As briefly mentioned above concerning the work by Goldberger and Melamud [Goldberger & Melamud 18a], when self-normalization is desired, it is also possible to directly penalize the model during training. The idea is simple - we describe what we want with the model outputs within the training criterion, e.g. for a self-normalized model, the denominator in the softmax function should be: constant $\rightarrow$ constant at one $\rightarrow$ have a variance of zero when it is constant at one.

To this end, for the task of machine translation, Devlin et al. [Devlin & Zbib$^+$ 14] proposed a self-normalization loss that directly encourages the logarithm of the normalization term to be zero, the strength of which is further controlled by a hyperparameter when combining this loss with the positive log probability. To regularize the variance of the normalization term, Shi et al. [Shi & Zhang$^+$ 14a, Shi & Zhang$^+$ 14b] proposed to minimize the variance during training. This was further adapted by Chen et al. [Chen & Liu$^+$ 15], where the authors also discussed how a proxy normalization term can be calculated on the validation set during test time, in order to obtain pseudo normalized model outputs. In the same spirit, Andreas and

Klein [Andreas & Klein 15] proved the generalization bounds when the variance is regularized during training, theoretically showing that when a large enough proportion of training examples is normalized, the normalization on other training examples also has some guarantees.

Having discussed three main branches of works that deal with the large vocabulary problem in word-level neural language models, the rest of the chapter is arranged as follows. First, we revisit three traditional training criteria (Section 3.3), where we show why these classic training criteria give self-normalized models. Although this part is well known and is not new, it serves as the basis of further discussions on sampling-based training criteria. Next, several sampling-based training criteria are described (Section 3.4), and we formally derive their theoretical optimums. This step reveals why some criteria are self-normalized and others are not, and explains why the field has preferred one over another for certain tasks. Specifically, in this section, we propose a universal method to correct the raw model outputs even when they are not self-normalized by definition. Also, three variants of the importance sampling training criterion are further proposed, all of which possess the desired self-normalization property. For completeness of the discussion, explicit self-normalization and variance regularization are then formally defined (Section 3.5). Finally, experimental results are provided (Section 3.6), and it is shown that the proposed model output correction and self-normalized importance sampling work as expected in large-scale research and commercial tasks.

## 3.2 Notations

To clarify the notations, in this chapter:

- $n$ is a running index in the total number of target word positions $N$.
- $x$ denotes a certain context for the next word prediction, in all possible contexts $X$.
- $p(x)$ denotes the empirical prior distribution of the context $x$.
- $c$ and $c'$ are running indices in the target vocabulary size $C$[1], which is supposedly very large, e.g. in the order of several hundred thousands. $c_k$ denotes the $k$-th sampled class when drawing random samples from some noise distribution.
- $p(c|x)$ denotes the empirical class posterior probabilities on the training data.
- $\theta$ denotes the learnable parameters of the neural network model.
- $q(x, c)$ or $q(c|x)$ denotes the model outputs, either when they are unconstrained or when they are constrained to be normalized in $C$.
- $\hat{q}(x, c)$ denotes the model outputs when the optimum is attained.
- $\mathcal{D}$ is the noise distribution from which we sample, i.e. $c_k \sim \mathcal{D}$.
- $k$ is a running index in the total number of samples $K$.
- $\delta$ is the Kronecker delta to decide the identity of the prediction of the model and the ground truth target word, which evaluates to one when the model is correct, and zero when the model is wrong.

[1] Previously, we used $V$ to denote the vocabulary size. Here, we use $C$ to hint that it can also be interpreted as the total number of classes for a general classification task.

## 3.3 Traditional Training Criteria

To arrive at the correction method for sampling-based training criteria as well as self-normalized importance sampling training criteria, we start by revisiting three traditional training criteria, namely, mean squared error, binary cross entropy, and cross entropy.

### 3.3.1 Mean Squared Error

The Mean Squared Error (MSE) training criterion is commonly used for regression problems, but it is also applicable to classification problems by training the model output towards the empirical class posterior probabilities. Intuitively, the criterion can be thought of as counting errors in the continuous sense, and one tries to minimize the error counts:

$$
\begin{aligned}
L_{\mathrm{MSE}}(\theta) &\coloneqq \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\big(q_\theta(x_n,c)-\delta(c_n,c)\big)^2 \qquad (3.1)\\
&= \sum_{x}^{X} p(x)\sum_{c=1}^{C} p(c|x)\sum_{c'=1}^{C}\big(q_\theta^2(x,c')+\delta^2(c,c')-2q_\theta(x,c')\delta(c,c')\big)\\
&= \sum_{x}^{X} p(x)\left(\sum_{c'=1}^{C} q_\theta^2(x,c')+1-2\sum_{c=1}^{C} p(c|x)q_\theta(x,c)\right)\\
&= \sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} q_\theta^2(x,c)+\sum_{c=1}^{C} p^2(c|x)-2\sum_{c=1}^{C} p(c|x)q_\theta(x,c)+1-\sum_{c=1}^{C} p^2(c|x)\right)\\
&= \sum_{x}^{X} p(x)\left(\sum_{c=1}^{C}\big(q_\theta(x,c)-p(c|x)\big)^2+\text{const.}\right)\\
&\implies \hat{q}_\theta(x,c)=p(c|x)
\end{aligned}
$$

In the derivation above, the quadratic term $\big(q_\theta(x_n,c)-\delta(c_n,c)\big)^2$ is first expanded, the outer summation over $C$ is then expanded, the terms inside the big parentheses are rearranged by separating the ones dependent on $x$ and the ones not dependent on $x$, and finally, the quadratic terms $\sum_{c=1}^{C}(q_\theta(x,c)-p(c|x))^2$ are singled out. It is immediately obvious that the global optimum is attained when $\hat{q}_\theta(x,c)=p(c|x)$ for all $x,c$ pairs. This is an important observation, because whenever the model needs to satisfy this to arrive at the optimum, the model is conveniently self-normalized.

By definition, $q$ is unconstrained, but it is possible to constrain $q$ to be positive. In our preliminary experiments, using a sigmoid function to further constrain $q$ to be in $[0,1]$ can slightly boost the final model accuracy. That said, our models trained with MSE are significantly worse than those trained with binary cross entropy or cross entropy, despite our spending a great amount of effort to tune the MSE model. For instance, we tried to initialize the model with a converged model trained with cross entropy and then continue training with MSE, used different gradient optimizers, and grid-searched over the learning rates. Although in Hui and Belkin [Hui & Belkin 21], the authors showed slight improvements using the MSE loss over cross entropy loss on various tasks, we argue for word-level neural language modeling, where the number of target classes is large, and it is hard for an MSE model to converge with our training setups. Golik et al. [Golik & Doetsch$^+$ 13] derived an upper bound and a lower bound for the MSE loss and also showed empirically that the gradient under MSE tends to vanish

quickly, leading the model to a worse local optimum, with which our experience also agrees. We additionally found that out of the $C$ error counts that are summed up together, if we down-scale the penalization on rival classes where $c \neq c_n$, the final perplexity improves by a little, but the same trick does not bring any further improvements for models trained with binary cross entropy or cross entropy. Because of the empirically worse performance of MSE on word-level language modeling, we do not further mention it later, and only include this section here for the sake of completeness.

### 3.3.2 Binary Cross Entropy

The Binary Cross Entropy (BCE) training criterion is another classic training criterion that is applicable to classification problems. For each position, the criterion is made up of two terms, a positive term that encourages the model score on the target word, and $C-1$ negative terms that penalize the model scores on the rival words. Here, because of the logarithmic functions, $q$ is bounded within (0, 1), and is commonly activated with a sigmoid function:

$$
\begin{aligned}
L_{\mathrm{BCE}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log q_\theta(x_n, c_n) + \sum_{c'=1, c'\neq c_n}^{C} \log\big(1 - q_\theta(x_n, c')\big)\right) \qquad (3.2)\\
&= -\sum_{x}^{X} p(x) \sum_{c=1}^{C} p(c|x)\left(\log q_\theta(x,c) - \log\big(1-q_\theta(x,c)\big) + \sum_{c'=1}^{C}\log\big(1-q_\theta(x,c')\big)\right)\\
&= -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x)\log q_\theta(x,c) + \sum_{c=1}^{C}\big(1-p(c|x)\big)\log\big(1-q_\theta(x,c)\big)\right)\\
&= -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x)\log\frac{q_\theta(x,c)}{p(c|x)} + \sum_{c=1}^{C}\big(1-p(c|x)\big)\log\frac{1-q_\theta(x,c)}{1-p(c|x)} + \text{const.}\right)\\
&\implies \hat{q}_\theta(x,c) = p(c|x)
\end{aligned}
$$

In the derivation above, the first step supplements the inner summation in $C-1$ by further including the target word, the second step moves the outer summation in $C$ into the parentheses and groups the $\log q$ and $\log(1-q)$ terms, and the final step changes the surface form such that divergence inequality can be applied and the terms not dependent on $q$ are grouped in "const.". Then, applying divergence inequality for both summations in the parentheses, we find that for both summations the optimum is attained when $\hat{q}_\theta(x,c) = p(c|x)$ for all $c$, given $x$. As in the case of MSE, because of this, BCE is also self-normalized.

### 3.3.3 Cross Entropy

The third traditional criterion is Cross Entropy (CE). This is in most cases the default choice of training criterion for neural language models. Here, we explicitly write out the softmax operation to highlight the summation in the big vocabulary $C$ in the denominator. In this case, $q$ is the raw logit output from the model, unbounded, and the normalization of the class posterior probabilities is guaranteed by the softmax function:

$$
\begin{aligned}
L_{\mathrm{CE}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\log\frac{\exp q_\theta(x_n, c_n)}{\sum_{c'=1}^{C}\exp q_\theta(x_n, c')} \\
&= -\sum_{x}^{X} p(x)\sum_{c}^{C} p(c|x)\log\frac{\exp q_\theta(x,c)}{\sum_{c'=1}^{C}\exp q_\theta(x,c')} \\
&= -\sum_{x}^{X} p(x)\left(\sum_{c}^{C} p(c|x)\log\frac{\exp q_\theta(x,c)/\sum_{c'=1}^{C}\exp q_\theta(x,c')}{p(c|x)} + \text{const.}\right) \\
&\implies \frac{\exp \hat{q}_\theta(x,c)}{\sum_{c'=1}^{C}\exp \hat{q}_\theta(x,c')} = p(c|x)
\end{aligned}
\tag{3.3}
$$

Because the softmax function is expanded in the derivation above to make obvious the summation in $C$, the derivation looks more complicated than it actually is. Essentially, the normalized model output: $\mathrm{softmax}\big(q_\theta(x,c)\big) = \exp q_\theta(x,c)/\sum_{c'=1}^{C}\exp q_\theta(x,c')$ is treated as a whole, and the divergence inequality is applied by constructing the $\sum p_1 \log(p_2/p_1)$ term. As the result suggests, the cross entropy criterion is also self-normalized, because activating the unbounded model logits $\hat{q}_\theta(x,c)$ directly with the softmax function gives us the desired class posterior probabilities $p(c|x)$.

## 3.4 Sampling-Based Training Criteria

In the previous discussions about mean squared error, binary cross entropy and cross entropy, we saw a summation in $C$ in all three cases. As mentioned before, because $C$ is large, this summation is the core of the problem. Before going into the details of each sampling-based training criterion, we can first reconsider the summation.

Denoting some $c$-dependent quantity as $Q_c$, notice how a summation of $Q_c$ in $C$ can be rewritten into an expectation of $Q_c$ under the uniform distribution:

$$
\sum_{c=1}^{C} Q_c = C\sum_{c=1}^{C}\frac{1}{C}Q_c \tag{3.4}
$$

$$
= C\cdot\mathbb{E}(Q_c) \tag{3.5}
$$

That means, when the original summation $\sum_{c=1}^{C} Q_c$ is costly to compute, we can instead approximate the expectation $\mathbb{E}(Q_c)$. This is the core concept of the sampling-based training criterion that follow.

In the derivations that are to come, we often apply an approximation:

$$
\sum_{n=1}^{N}\sum_{k=1}^{K} Q_{n,k} \approx \sum_{n=1}^{N}\sum_{c=1}^{C} K\mathcal{D}(c)Q_{n,c} \tag{3.6}
$$

Here, the goal is to rewrite the summation in $K$ into a summation in $C$. We want to do this because handling the summation in $K$ is hard - it depends on the number of samples $K$, while the positive contribution in the training criterion, e.g. the positive term on the target word $\log q_\theta(x,c)$ in binary cross entropy, does not. Rewriting it into a summation in $C$ is beneficial, because then it is possible to merge it with the positive contribution term, which we will see

later in this section. Below, we give a brief explanation of why the approximation holds. If we ask ourselves the question, "how many terms show up if we expand the summation in $K$ on the left side", we can go over each class $c$ and answer, "for a given class $c$, e.g., if the noise distribution $\mathcal{D}$ says $\mathcal{D}(c)$ is 10%, then when $K$ is 1000, we roughly count 100 times that $c$ shows up during the sampling." That is, when $N$ and $K$ are large enough, we can approximate the summation in $K$ by binning the samples by class and go over all classes instead. This trick is applied several times in our derivations, and is the key step in many of them.

### 3.4.1 Monte Carlo Sampling

Monte Carlo Sampling (MCS) is a classic method that relies on the law of large numbers to approximate an expectation by repeatedly drawing random samples and calculating the empirical mean (sample mean). In the literature, negative sampling [Mikolov & Sutskever+ 13] is a prominent example of Monte Carlo sampling. In the original paper by Mikolov et al., the authors introduced negative sampling from the perspective of simplifying noise contrastive estimation and, arguably, their criterion has a surface form that is closer to the binary cross entropy. Here, we can directly apply Monte Carlo sampling on the summation in $C$ for both binary cross entropy and cross entropy. Specifically, instead of summing over $C$, we draw $K$ samples from some noise distribution, i.e. $c_k \sim \mathcal{D}$, and sum over $K$:

$$
\begin{aligned}
L_{\text{BCE-MCS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log q_\theta(x_n, c_n) + \sum_{k=1}^{K}\log\big(1 - q_\theta(x_n, c_k)\big)\right) \\
&\approx -\sum_{x}^{X} p(x)\sum_{c=1}^{C}\left(p(c|x)\log q_\theta(x,c) + K\mathcal{D}(c)\log\big(1 - q_\theta(x,c)\big)\right) \\
\frac{\partial L_{\text{BCE-MCS}}(\theta)}{\partial q_\theta(x,c)} &\approx p(x)\left(\frac{p(c|x)}{q_\theta(x,c)} - \frac{K\mathcal{D}(c)}{1 - q_\theta(x,c)}\right) \stackrel{!}{=} 0 \\
&\implies \hat{q}(x,c) \approx \big(1 + \frac{K\mathcal{D}(c)}{p(c|x)}\big)^{-1}
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
L_{\text{CE-MCS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - \log\sum_{k=1}^{K}\exp q_\theta(x_n, c_k)\right) \\
&\approx -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x) q_\theta(x,c) - \log\sum_{c=1}^{C} K\mathcal{D}(c)\exp q_\theta(x,c)\right) \\
\frac{\partial L_{\text{CE-MCS}}(\theta)}{\partial q_\theta(x,c)} &\approx -p(x)\left(p(c|x) - \frac{K\mathcal{D}(c)\exp q_\theta(x,c)}{\sum_{c'=1}^{C} K\mathcal{D}(c')\exp q_\theta(x,c')}\right) \stackrel{!}{=} 0 \\
&\implies \frac{\mathcal{D}(c)\exp \hat{q}_\theta(x,c)}{\sum_{c'=1}^{C}\mathcal{D}(c')\exp \hat{q}_\theta(x,c')} \approx p(c|x)
\end{aligned}
\tag{3.8}
$$

In the derivations above, the trick in Equation 3.6 is first applied. Without loss of generality, the partial derivatives with respect to $q_\theta(x, c)$ are then set to zero, and $\hat{q}$ is finally derived. Notice that the derivations are a bit sloppy here because we did not check the second-order derivative to verify if the optimum is the maximum or the minimum, and the boundary conditions, i.e.

$\lim_{q\to 0} L$ and $\lim_{q\to 1} L$, are not explicitly discussed. For $L_{\text{BCE-MCS}}$, $q$ is constrained between zero and one and activated by a sigmoid function. For $L_{\text{CE-MCS}}$, because $q$ denotes the raw logits and the softmax function is explicitly written out, $q$ is not constrained. Looking at the results, we quickly notice the difference compared to the traditional training criteria, that is, the model outputs are no longer self-normalized. Interestingly, for $L_{\text{CE-MCS}}$, adding the target-word-dependent bias $\log \mathcal{D}(c)$ to the raw model logits $\hat{q}_\theta(x, c)$ and then activating it with a softmax function will give us the class posterior probability $p(c|x)$.

Some other works that employ the Monte Carlo sampling concept include a TensorFlow function called "`tf.nn.sampled_softmax_loss`" [TensorFlow 22], and the work by Jean et al. [Jean & Cho$^+$ 15] where they try to address the large vocabulary problem in neural machine translation.

### 3.4.2 Compensated Partial Summation

Compensated Partial Summation (CPS) is a simple extension to Monte Carlo sampling with a straightforward motivation. That is, if we replace the summation of $C$ terms with a partial summation of $K$ terms, it might make sense to compensate the partial summation by a factor, $\alpha = C/K$. This idea is naive, because counter-examples are easy to find. Say $Q_1 = 1, Q_2 = 2, Q_3 = 3, K = 1$ and $Q_1$ is sampled out, the result of the correction $\alpha \cdot \sum_{k=1}^{K} Q_k = 3$ is still far off from the original sum $\sum_{c=1}^{C} Q_c = 6$. That said, we nonetheless visit this idea here:

$$
\begin{aligned}
L_{\text{BCE-CPS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log q_\theta(x_n, c_n) + \alpha \sum_{k=1}^{K} \log\big(1 - q_\theta(x_n, c_k)\big)\right) \qquad (3.9)\\
&\approx -\sum_{x}^{X} p(x) \sum_{c=1}^{C}\left(p(c|x)\log q_\theta(x,c) + \alpha K \mathcal{D}(c)\log\big(1 - q_\theta(x,c)\big)\right)\\
\frac{\partial L_{\text{BCE-CPS}}(\theta)}{\partial q_\theta(x,c)} &\approx p(x)\left(\frac{p(c|x)}{q_\theta(x,c)} - \frac{\alpha K \mathcal{D}(c)}{1 - q_\theta(x,c)}\right) \stackrel{!}{=} 0\\
&\implies \hat{q}(x,c) \approx \big(1 + \frac{\alpha K \mathcal{D}(c)}{p(c|x)}\big)^{-1}
\end{aligned}
$$

$$
\begin{aligned}
L_{\text{CE-CPS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - \log \alpha \sum_{k=1}^{K} \exp q_\theta(x_n, c_k)\right) \qquad (3.10)\\
&\approx -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x) q_\theta(x,c) - \log\alpha \sum_{c=1}^{C} K\mathcal{D}(c)\exp q_\theta(x,c)\right)\\
\frac{\partial L_{\text{CE-CPS}}(\theta)}{\partial q_\theta(x,c)} &\approx -p(x)\left(p(c|x) - \frac{\alpha K \mathcal{D}(c)\exp q_\theta(x,c)}{\sum_{c'=1}^{C} \alpha K \mathcal{D}(c')\exp q_\theta(x,c')}\right) \stackrel{!}{=} 0\\
&\implies \frac{\mathcal{D}(c)\exp \hat{q}_\theta(x,c)}{\sum_{c'=1}^{C}\mathcal{D}(c')\exp \hat{q}_\theta(x,c')} \approx p(c|x)
\end{aligned}
$$

The derivations are essentially the same as in Monte Carlo sampling. For $L_{\text{BCE-CPS}}$, the correction ratio $\alpha = C/K$ is kept in $\hat{q}$, and for $L_{\text{CE-CPS}}$, the two $\alpha$ scalars in the numerator and the denominator cancel each other in the derivative step, and are not retained in the final $\hat{q}$.

### 3.4.3 Importance Sampling

Importance Sampling (IS) introduces the noise distribution $\mathcal{D}$ into the expectation calculation[2] in Equation 3.4. As is the case for Monte Carlo sampling in Section 3.4.1, this can be applied to both binary cross entropy and cross entropy:

$$
\begin{aligned}
L_{\text{BCE-IS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log q_\theta(x_n, c_n) + \sum_{k=1}^{K}\frac{\log\left(1 - q_\theta(x_n, c_k)\right)}{K\mathcal{D}(c_k)}\right) \\
&\approx -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x)\log q_\theta(x, c) + \sum_{c=1}^{C}\log\left(1 - q_\theta(x, c)\right)\right) \\
\frac{\partial L_{\text{BCE-IS}}(\theta)}{\partial q_\theta(x, c)} &\approx -p(x)\left(\frac{p(c|x)}{q_\theta(x, c)} - \frac{1}{1 - q_\theta(x, c)}\right) \stackrel{!}{=} 0 \\
&\implies \hat{q}_\theta(x, c) \approx \left(1 + \frac{1}{p(c|x)}\right)^{-1}
\end{aligned}
\tag{3.11}
$$

$$
\begin{aligned}
L_{\text{CE-IS}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - \log\sum_{k=1}^{K}\frac{\exp q_\theta(x_n, c_k)}{K\mathcal{D}(c_k)}\right) \\
&\approx -\sum_{x}^{X} p(x)\left(\sum_{c=1}^{C} p(c|x) q_\theta(x, c) - \log\sum_{c=1}^{C}\exp q_\theta(x, c)\right) \\
\frac{\partial L_{\text{CE-IS}}(\theta)}{\partial q_\theta(x, c)} &\approx -p(x)\left(p(c|x) - \frac{\exp q_\theta(x, c)}{\sum_{c'=1}^{C}\exp q_\theta(x, c')}\right) \stackrel{!}{=} 0 \\
&\implies \frac{\exp \hat{q}_\theta(x, c)}{\sum_{c'=1}^{C}\exp \hat{q}_\theta(x, c')} = p(c|x)
\end{aligned}
\tag{3.12}
$$

To obtain $\hat{q}$ for both cases, we follow similar steps as before, i.e. first rewrite the summation in $K$ and then set the gradient to zero. For $L_{\text{BCE-IS}}(\theta)$, $\hat{q}$ has a curvature of the multiplicative inverse of $1 + p(c|x)$. For $L_{\text{CE-IS}}(\theta)$, activating $\hat{q}$ with the softmax function will yield $p(c|x)$. Note that this observation is also made in Jozefowicz et al. [Jozefowicz & Vinyals$^+$ 16] (Section 3.1 in their paper).

### 3.4.4 Noise Contrastive Estimation

Noise Contrastive Estimation (NCE) replaces the original task of "one out of $C$" next word prediction with a binary classification task of telling true samples from noisy ones. In the original papers by Gutmann and Hyvärinen [Gutmann & Hyvärinen 10, Gutmann & Hyvärinen 12], noise contrastive estimation was proposed in the context of binary cross entropy:

[2] For details, we refer the reader to Tom Kennedy's lecture notes on Monte Carlo methods [Kennedy 16], specifically, Equation 6.4.

$$L_{\text{BCE-NCE}} \coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log\frac{q_\theta(x_n,c_n)}{q_\theta(x_n,c_n)+K\mathcal{D}(c_n)}+\sum_{k=1}^{K}\log(1-\frac{q_\theta(x_n,c_k)}{q_\theta(x_n,c_k)+K\mathcal{D}(c_k)})\right) \tag{3.13}$$

$$\approx -\sum_{x}^{X}p(x)\sum_{c=1}^{C}\left(p(c|x)\log\frac{q_\theta(x,c)}{q_\theta(x,c)+K\mathcal{D}(c)}+K\mathcal{D}(c)\log\frac{K\mathcal{D}(c)}{q_\theta(x,c)+K\mathcal{D}(c)}\right)$$

$$\frac{\partial L_{\text{BCE-NCE}}(\theta)}{\partial q_\theta(x,c)} \approx -p(x)\left(\frac{p(c|x)}{q_\theta(x,c)}-\frac{p(c|x)+K\mathcal{D}(c)}{q_\theta(x,c)+K\mathcal{D}(c)}\right)\overset{!}{=}0$$

$$\implies \hat{q}_\theta(x,c)=p(c|x)$$

The derivation is similar to the previous, where the summation in $K$ is first rewritten into a summation in $C$, and the gradient with respect to $q_\theta(x,c)$ is set to zero to obtain $\hat{q}$. One thing worth noting about the noise contrastive estimation training criterion is its self-normalization property, as seen in the derivation above, $\hat{q}_\theta(x,c)=p(c|x)$, meaning that when the optimum is obtained the model outputs are the desired class posterior properties. In practice, if $K$ is set sufficiently large, and the model is trained well till convergence, the model can be as good as a model trained with cross entropy or binary cross entropy. For this reason, noise contrastive estimation enjoys great popularity in the literature [Gutmann & Hyvärinen 10, Gutmann & Hyvärinen 12, Mnih & Teh 12, Jozefowicz & Vinyals$^+$ 16, Chen & Liu$^+$ 16b, Goldberger & Melamud 18b, van den Oord & Li$^+$ 18, Poole & Ozair$^+$ 19, Tschannen & Djolonga$^+$ 20, Kong & de Masson d'Autume$^+$ 20].

### 3.4.5 Self-Normalized Importance Sampling

So far, we have discussed several sampling-based training criteria for neural language modeling, and derived their model outputs $\hat{q}_\theta(x,c)$ when the optimum is attained. We also made the observation that if $\hat{q}_\theta(x,c)$ is not strictly the intended class posterior probability $p(c|x)$, the model is not self-normalized. In other words, we prefer training criteria that fulfill $\hat{q}_\theta(x,c)=p(c|x)$, which is the case for mean squared error, binary cross entropy, cross entropy, and noise contrastive estimation. This observation is not new to the field, for example:

- In the work by Bengio and Senecal [Bengio & Senecal 03], the authors trained using the importance sampling criterion and noticed that the gradient tended to become less "accurate" as training went on, and the training diverged with a small sampling size. Therefore, they proposed an algorithm with dynamic sampling size to mitigate this problem, where the sampling size increases according to on-the-fly perplexity diagnostics on a tiny dataset. This observation of unstable training with importance sampling is further discussed in the follow-up work by Bengio and Senecal [Bengio & Senecal 08], where they attribute this to a too-large bias in the model output and/or a too-large variance in the gradient estimator.

- In the work by Mikolov et al. [Mikolov & Sutskever$^+$ 13] where they introduced negative sampling, they acknowledged that they only aim to learn high-quality vector representations with negative sampling, and not to "approximately maximize the log probability of the softmax".

- In a preprint note by Chris Dyer [Dyer 14] where he explained the difference between negative sampling and noise contrastive estimation (NCE), he wrote "if your goal is language modeling, you should use NCE; if your goal is word representation learning, you should consider both NCE and negative sampling".
- In the work by Jozefowicz et al. [Jozefowicz & Vinyals$^+$ 16], the authors noticed that it is possible to activate the logits of a model trained with importance sampling with the softmax function and obtain the desired class posterior probability, as is discussed in Section 3.4.3. However, they then argued that training with importance sampling might be better than training with noise contrastive estimation because the former is a multi-class classification task and the latter is a binary classification task.
- In the work by Oualil and Klakow [Oualil & Klakow 17], the authors summarized existing work related to importance sampling and noise contrastive sampling at the time, and noted that noise contrastive sampling is an attractive choice to train neural language models with a large vocabulary.

Despite the previous experience, here we make two contributions:

1. Although for different sampling-based training criteria, we may not directly have $\hat{q}_\theta(x, c) = p(c|x)$, there clearly exists a one-to-one mapping between the raw model outputs and the desired class posterior probabilities. In other words, once the exact form of $\hat{q}$ is derived, it is possible to solve for $p(c|x)$ to obtain an approximately self-normalized model, for all sampling-based training criteria examined in this chapter [Gao & Thulke$^+$ 21].
2. Considering the derivation for importance sampling, we show that it is possible to further adjust the importance sampling criterion to make it self-normalized, effectively arriving at a competitive alternative to the long-standing and popular noise contrastive estimation training criterion, when combating the large vocabulary problem [Yang & Gao$^+$ 22].

The first point is straightforward. If we think the model is optimized well during training, then at test time, given the model output, we can solve for $p(c|x)$ according to the derivations earlier, and use that as pseudo normalized class posterior probabilities for downstream tasks like second-pass rescoring for automatic speech recognition. Because we assume the models are trained well, we can expect that such an approach should perform equally well as training with the self-normalized training criteria such as cross entropy and noise contrastive estimation. Experimental results support the claims, as will be later discussed in Section 3.6. Below, we expand on the second point.

If we look closely at the derivations of binary cross entropy (BCE) and binary cross entropy with importance sampling (BCE-IS), i.e. Equation 3.2 and Equation 3.11, we quickly notice that they are very similar, with one small difference - the summation over negative contributions, i.e. the $\log(1-q)$ terms, includes the target class $c$ for BCE-IS, but not for BCE. For the sake of this discussion, we rewrite the two equations below:

$$L_{\text{BCE}}(\theta) = -\sum_{x}^{X} p(x) \sum_{c=1}^{C} p(c|x) \left( \log q_\theta(x, c) + \sum_{c'=1}^{C} \log \big(1 - q_\theta(x, c')\big) - \log \big(1 - q_\theta(x, c)\big) \right) \tag{3.14}$$

$$L_{\text{BCE-IS}}(\theta) \approx -\sum_{x}^{X} p(x) \sum_{c=1}^{C} p(c|x) \left( \log q_\theta(x, c) + \sum_{c'=1}^{C} \log \big(1 - q_\theta(x, c')\big) \right) \tag{3.15}$$

Clearly, there is a chance for $c$ to be sampled out when training BCE-IS, and this is the main cause of non-self-normalized model outputs. In other words, if we want self-normalization with importance sampling, we need the summation to only go over rival words and not the entire vocabulary - or, to exclude the contribution of $\log\big(1 - q_\theta(x, c)\big)$ when doing sampling. As discussed in the paper by Yang et al. [Yang & Gao$^+$ 22], there are different approaches to doing this, and we will only focus on one of them here (Mode 3 in their paper).

Formally, the self-normalized importance sampling (SNIS) can be defined to be:

$$L_{\text{SNIS}}(\theta) \coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(\log q_\theta(x_n, c_n) + \sum_{k=1}^{K} f_{c_n}\big(x_n, c_k, \theta, K, \mathcal{D}\big)\right) \tag{3.16}$$

$$\text{where} \quad f_{c_n}\big(x_n, c_k, \theta, K, \mathcal{D}\big) = \begin{cases} 0, & \text{if} \quad c_k = c_n \\ \frac{\log\left(1 - q_\theta(x_n, c_k)\right)}{K\mathcal{D}(c_k)}, & \text{else} \end{cases}$$

$$\implies \hat{q}_\theta(x, c) = p(c|x)$$

Simply put, we set the contribution to the summation in $K$ to zero whenever the target word $c$ gets sampled out. In practice, we perform sampling without replacement, because if replacement is allowed, one could be unlucky and sample the target class many times, which all evaluate to zero and contribute nothing to the summation, thus affecting the quality of the approximation. Reversing the discussion above, one quickly sees that the model output $\hat{q}_\theta$ is the desired class posterior $p(c|x)$, i.e., the model is self-normalized.

## 3.5 Self-Normalization and Variance Regularization

To briefly take a detour away from sampling-based training criteria, there are two related regularization tricks that can encourage self-normalization when training language models, namely self-normalization and variance regularization [Devlin & Zbib$^+$ 14, Shi & Zhang$^+$ 14a, Shi & Zhang$^+$ 14b, Chen & Liu$^+$ 15, Andreas & Klein 15]. If we reconsider and rewrite the cross entropy criterion (Equation 3.3):

$$\begin{aligned} L_{\text{CE}}(\theta) &\coloneqq -\frac{1}{N}\sum_{n=1}^{N}\log\frac{\exp q_\theta(x_n, c_n)}{\sum_{c'=1}^{C}\exp q_\theta(x_n, c')} \\ &= -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - \log\sum_{c'=1}^{C}\exp q_\theta(x_n, c')\right) \\ &= -\frac{1}{N}\sum_{n=1}^{N}\Big(q_\theta(x_n, c_n) - Z_n\Big) \end{aligned} \tag{3.17}$$

During training, both methods require the calculation of the normalization factor, or the denominator $Z_n$ (in actual implementations, because we usually do batched training, $Z_n$ is the normalization factor of the current batch), and are intended only to speed up the search process, i.e. not to calculate $Z_n$ during search and hope it is close enough to one. For the sake of the completeness of the discussion, in this section, we quickly visit these two methods.

### 3.5.1 Motivation and Definition

The idea of self-normalization (SN) is to explicitly encourage $Z_n$ to be one, i.e. $\log Z_n$ to be zero:

$$L_{\text{CE-SN}}(\theta) \coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - Z_n - \lambda_{\text{SN}} \log^2 Z_n\right) \tag{3.18}$$

Here, $\lambda_{\text{SN}}$ ($\lambda_{\text{SN}} \geq 0$) is a hyperparameter controlling the strength of the regularization, and the model learns to set $Z_n$ (for each batch) close to one because of the squared error term.

Similarly, when $Z_n$ is supposed to be constant at one, its variance should be as close to zero as possible. Therefore, variance regularization (VR) explicitly encourages this:

$$L_{\text{CE-VR}}(\theta) \coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - Z_n - \lambda_{\text{VR}} \left(\log Z_n - \log \bar{Z_n}\right)^2\right) \tag{3.19}$$

Here, $\bar{Z_n}$ is the mean of $\log Z_n$ of the batch, and $\lambda_{\text{VR}}$ ($\lambda_{\text{VR}} \geq 0$) controls the regularization strength.

Since the optimization direction of the two tricks is similar (i.e., towards constant $Z_n$ at one), it makes sense to combine the two:

$$L_{\text{CE-SNVR}}(\theta) \coloneqq -\frac{1}{N}\sum_{n=1}^{N}\left(q_\theta(x_n, c_n) - Z_n - \lambda_{\text{SN}} \log^2 Z_n - \lambda_{\text{VR}} \left(\log Z_n - \log \bar{Z_n}\right)^2\right) \tag{3.20}$$

Using this combined and regularized version of the cross entropy criterion, one can train a language model to be quasi-self-normalized, and speed up the search process by not explicitly calculating $Z_n$ on the test data.

### 3.5.2 Quality of Normalization

The hyperparameters $\lambda_{\text{SN}}$ and $\lambda_{\text{VR}}$ control the regularization strength, and sweeping through different values of them would reveal the impact on the normalization quality. Specifically, we experiment on the Switchboard language modeling task, use the same $\lambda$ values for both tricks (i.e. $\lambda = \lambda_{\text{SN}} = \lambda_{\text{VR}}$), calculate the mean and variance of $Z$, and also report the normalized (properly calculating $Z$ at test time) and unnormalized (using the model logits as a "pseudo model posterior probability distribution" at test time) perplexities on the test set.

The detailed results are given in Table 3.1. Looking at the mean values of $Z$ at different regularization strengths, it is clear that a larger $\lambda$ leads to a better self-normalized model, with the mean of $Z$ getting closer and closer to one. A similar observation can be made about the variance. As $\lambda$ goes to 100, the test $Z$ variance gets very close to zero, at 0.0004. This means that the self-normalization tricks can indeed control how the model generates the logits. Looking at the normalized test perplexities, in all cases, the perplexities are similar (the absolute values are not the best, because little effort is spent in tuning this model and it is not further interpolated with a count-based language model, which is the case for the experiments later in Section 3.6), but slightly degrades with increasing $\lambda$. On the other hand, when looking at the unnormalized test perplexities, some interesting observations can be made. When $\lambda$ is very small, the regularization is weak, the model has much freedom in enlarging its logits at the target words, and this leads to overconfident outputs (sometimes even larger than one). To avoid numerical problems, the outputs at these positions are capped at very close to one when

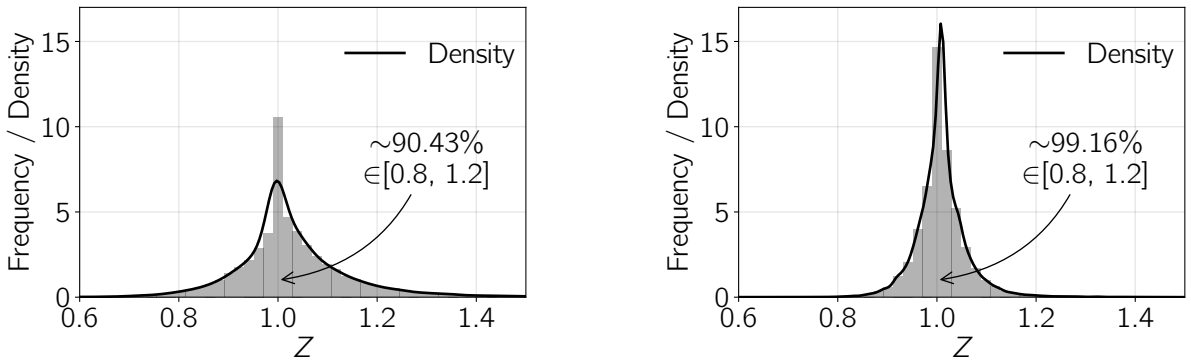
Table 3.1: Effect of self-normalization and variance regularization $\lambda$ on normalization quality and test perplexity (PPL) on Switchboard, $\lambda = \lambda_{\text{SN}} = \lambda_{\text{VR}}$. When poorly self-normalized (e.g. for $\lambda <$ 1e+1), the model is sometimes overconfident, assigning probability values larger than one to certain positions, and these outputs are capped at one for the calculation of unnormalized pseudo perplexities, leading to overconfident perplexities. As self-normalization improves (e.g. for $\lambda =$ 1e+1), this occurs less often, and the unnormalized pseudo perplexity is closer to the true perplexity. When the self-normalization is too strict (e.g. for $\lambda =$ 1e+2), the model is restricted to assign too large a probability value to positions it is confident about, thus resulting in the unnormalized pseudo perplexity being slightly larger than the true perplexity.

| $\lambda$ | $Z$ | | true PPL | pseudo PPL |
|---|---|---|---|---|
| | mean | variance | | |
| 1e−4 | 3842.7427 | 4635686454.8327 | 57.20 | 1.57 |
| 1e−3 | 75.5725 | 974413.4176 | 57.26 | 7.59 |
| 1e−2 | 2.0820 | 1163.5683 | 57.82 | 41.77 |
| 1e−1 | 1.1014 | 0.1316 | 57.55 | 54.46 |
| 1e+0 | 1.0334 | 0.0218 | 57.79 | 56.47 |
| 1e+1 | 1.0092 | 0.0029 | 58.48 | 58.03 |
| 1e+2 | 0.9961 | 0.0004 | 59.81 | 60.05 |

calculating the unnormalized perplexity. As $\lambda$ becomes larger, the self-normalization quality improves, and the unnormalized perplexity approaches the true perplexity. When $\lambda$ gets too large, the model needs to pay a big price when assigning too large logits to the target positions, and this leads to the unnormalized perplexity being a bit higher than the true perplexity. In real applications, e.g. second-pass rescoring for automatic speech recognition, the relative ranking among different words in the vocabulary is more important than the normalization itself, and language models regularized with self-normalization and variance regularization can become helpful in speeding up the search process [Chen & Liu$^+$ 15].

To further investigate the normalization quality, histograms as well as density curves of the denominator $Z$ are drawn in Figure 3.1, for when $\lambda$ is at 1 and 10. As seen, the model is capable at concentrating $Z$ at one when presented with unseen test data, and a stronger $\lambda$ indeed corresponds to a more concentrated distribution. When considering the range of [0.8, 1.2] (i.e. 20% error range), the majority of the $Z$ values lie in it, at roughly 90.43% and 99.16%, for $\lambda = 1$ and $\lambda = 10$ respectively. It is worth mentioning that when making these two figures, in both cases, there actually exists a long tail at the right-hand side, spanning to values even larger than 10. This means that despite the model being quasi-self-normalized, it still struggles to properly self-normalize at certain positions - the positions that it is very confident about and decides to assign logits larger than one to.

Self-normalization and variance regularization are simple and straightforward tricks to apply during the training phase to encourage self-normalized model outputs at test time. The same concept can be applied whenever a softmax-related normalization is too costly to compute, and is seen e.g. in works that try to make the attention calculation more efficient [Guo & Liu$^+$ 20].




(a) $\lambda$=1e+0 (b) $\lambda$=1e+1

Figure 3.1: Histograms and density curves of denominator $Z$ using self-normalization and variance regularization, with $\lambda$=1e+0 and $\lambda$=1e+1 ($\lambda = \lambda_{\text{SN}} = \lambda_{\text{VR}}$), on the test data of Switchboard. In both cases, 51 bins in the range of $[0.5, 1.5]$ are used for the histogram counts, and Gaussian kernels are used for kernel density estimation. As in Table 3.1, for (a) and (b), the test perplexities are 57.79 and 58.48, respectively.

Compared to sampling-based training criteria, self-normalization and variance regularization have the downside of not being able to speed up training (even slowing it down a little bit because of the overhead in calculating the regularization terms), and therefore we do not further discuss them and instead move on to the experimental results of sampling-based training criteria in the next section.

## 3.6 Experimental Results

### 3.6.1 Effect of Sampling Size

There is one hyperparameter that is common to all sampling-based training criteria discussed in Section 3.4, that is, the sampling size, $K$. As mentioned in Section 3.4, if $K$ is very small, we gain more speedup but potentially sacrifice some classification performance, and when $K$ is large, we do not have significant speedup, but the classification performance can be largely retained. In other words, in the speed-accuracy trade-off, there may be a sweet spot for $K$, where both the speed and the accuracy are good. Below, in Figure 3.2, we plot the training speed as well as the test perplexity while sweeping over different $K$ values, using the self-normalized importance sampling (Section 3.4.5) training criterion on Switchboard.

As can be seen from the figure, the overall trends match our expectations - small $K$ generally means faster training but worse perplexity, while large $K$ generally means slower training but better perplexity. An additional observation can be made: the dependency on $K$ is not linear, meaning that in both small $K$ and large $K$ regions, there exist "cutoffs" (around 400 to 600 for $K$ in Figure 3.2), beyond which the dependency on $K$ is not apparent but suddenly appears when the $K$ value moves past them. Here, we only experiment with one specific sampling-based training criterion on a specific dataset, and the exact curves may further depend on other factors such as the model architecture, the vocabulary size and so on, but the overall trends should be similar to what is shown here.

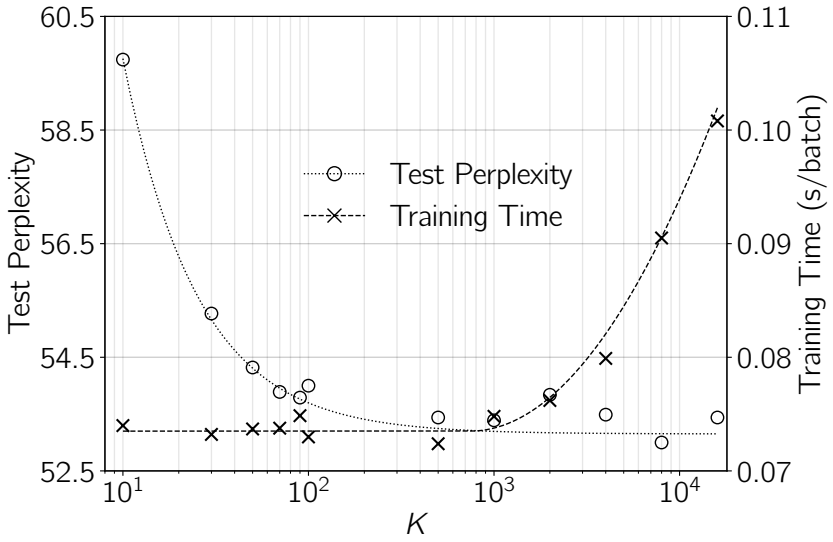



Figure 3.2: Influence of sampling size $K$ on training speed and test perplexity, with the self-normalized importance training criterion (Section 3.4.5), on Switchboard. The speed-accuracy trade-off shows that a small enough $K$ leads to faster training but worse perplexity, and a large enough $K$ leads to slower training but better perplexity. The dotted and dashed lines are curve fits done with non-linear least squares [SciPy 22].

### 3.6.2 Results with Explicit Normalization

In Section 3.4.5, we made the point that although we do not always have $\hat{q}_\theta(x, c) = p(c|x)$ for all sampling-based training criteria, because there exists a one-to-one mapping between the model outputs and class posterior probabilities, it is possible to explicitly solve for $p(c|x)$ and explicitly normalize it when necessary.

To this end, we conduct experiments on Switchboard and LibriSpeech. The vocabulary sizes are around 30k and 200k for Switchboard and LibriSpeech, respectively. For Switchboard, we train Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) language models [Sundermeyer & Schlüter$^+$ 12, Irie 20], and interpolate them with a count-based language model. For LibriSpeech, we train transformer language models [Irie & Zeyer$^+$ 19]. In both cases, the sampling-size is 8192, and the noise distribution from which we sample is a log-uniform one (in other words, probability decays exponentially with respect to word frequency rank). We perform second-pass lattice rescoring using lattices generated by strong baseline models [Beck & Zhou$^+$ 19, Kitza & Golik$^+$ 19]. For further information on model-related and training-related hyperparameters, as well as evaluation-related setups, we refer the reader to the paper by Gao et al. [Gao & Thulke$^+$ 21], where it is documented in more detail.

In Table 3.2, we show PerPLexities (PPLs) as well as Word Error Rates (WERs) on Switchboard by explicitly normalizing the language models. As seen, compared to the binary cross entropy (BCE) or the cross entropy (CE) baselines, all sampling-based language models speed up the training process by over 20%. The relative training speedup is consistent across all sampling-based training criteria. Considering the normalized PPL, all neural models are significantly better than the count-based model, which is confirmed in many works and expected. The CE baseline is slightly better than other neural models because a well-tuned recipe is used

Table 3.2: Sampling-based LSTM RNN language models on Switchboard, explicitly normalized after training. The 4-gram count-based Kneser-Ney (KN) language model is used to generate the first-pass lattices. The LSTM RNN language models are interpolated with the count-based model for second-pass rescoring. For sampling-based methods, language model outputs are corrected (Section 3.4.5) and normalized for both perplexity (PPL) and word error rate (WER) calculation.

| Model | Criterion | Sampling | Train Time (s/batch) | PPL (norm.) | WER (norm.) | | |
|---|---|---|---|---|---|---|---|
| | | | | | SWB | CH | All |
| 4-gram KN | - | - | - | 74.6 | 8.1 | 15.4 | 11.8 |
| LSTM RNN | BCE | - | 0.107 | 52.3 | 6.9 | 13.7 | 10.3 |
| | | MCS | 0.077 | 52.6 | 7.0 | 13.6 | 10.3 |
| | | CPS | 0.079 | 52.4 | 7.1 | 13.6 | 10.3 |
| | | IS | 0.079 | 51.5 | 7.0 | 13.7 | 10.3 |
| | | NCE | 0.078 | 51.4 | 7.0 | 13.6 | 10.3 |
| | CE | - | 0.100 | 49.9 | 6.8 | 13.4 | 10.1 |
| | | MCS | 0.078 | 52.4 | 7.0 | 13.7 | 10.4 |
| | | CPS | 0.077 | 52.3 | 7.0 | 13.6 | 10.3 |
| | | IS | 0.076 | 52.3 | 6.9 | 13.6 | 10.2 |

for this training run, and not much hyperparameter tuning is done for the rest of the criteria. In terms of WERs, all neural models are significantly better than the count-based model, which again verifies the empirical rule that better PPL generally leads to better WER [Bahl & Jelinek$^+$ 83, Chen & Beeferman$^+$ 98, Klakow & Peters 02, Sundermeyer & Ney$^+$ 15a, Irie 20]. However, among all neural models, the WERs are somewhat similar and it is not easy to conclude that one is significantly better than another. This result matches our theory, because for all sampling-based criteria considered here, $p(c|x)$ is obtainable from the model outputs and we should not expect much performance difference because of it.

Moving on to Table 3.3, similar observations can be made from experiments on the LibriSpeech dataset. Examining the training speed, compared to BCE and CE baselines, we can see that a significant relative speedup of over 40% is achieved across the board for sampling-based training criteria. The higher speedup over Switchboard is expected (40% versus 20%), because for Switchboard we sample around 8k samples from a 30k vocabulary, and for LibriSpeech we sample around 8k samples from a 200k vocabulary. Looking at the normalized PPLs, we observe some improvements over the LSTM baseline, which agrees with the results reported in other works [Lüscher & Beck$^+$ 19b, Irie & Zeyer$^+$ 19, Irie 20]. However, comparing the sampling-based criteria against the BCE and CE baselines, no significant differences can be found, and this again agrees with our expectation, as explained in the earlier paragraph. Finally, if we turn our attention to WERs, we see that improvements over the LSTM baseline is limited, and this is because the PPLs themselves are not sufficiently lower and we are rescoring on the given lattices. That said, an absolute WER improvement of 0.4 is still achieved with the

Table 3.3: Sampling-based transformer language models on LibriSpeech, explicitly normalized after training. The well-tuned baseline LSTM RNN language model is used to generate the first-pass lattices. The transformer language models are used for second-pass rescoring. For sampling-based methods, language model outputs are corrected (Section 3.4.5) and normalized for both perplexity (PPL) and word error rate (WER) calculation.

| Model | Criterion | Sampling | Train Time (s/batch) | PPL (norm.) | WER (norm.) | |
|---|---|---|---|---|---|---|
| | | | | | clean | other |
| LSTM RNN | - | - | - | 64.3 | 2.6 | 5.8 |
| Transformer | BCE | - | 0.358 | 58.5 | 2.5 | 5.4 |
| | | MCS | 0.213 | 58.0 | 2.6 | 5.4 |
| | | CPS | 0.205 | 58.4 | 2.5 | 5.4 |
| | | IS | 0.206 | 58.4 | 2.6 | 5.5 |
| | | NCE | 0.206 | 57.9 | 2.5 | 5.4 |
| | CE | - | 0.302 | 57.7 | 2.5 | 5.4 |
| | | MCS | 0.206 | 57.9 | 2.5 | 5.4 |
| | | CPS | 0.203 | 62.2 | 2.5 | 5.4 |
| | | IS | 0.201 | 58.7 | 2.5 | 5.4 |

transformer language models on test-other. If we further compare the sampling-based training criteria against BCE and CE baselines, we see no significant different differences in WERs, which is more evidence supporting the empirical law between PPL and WER [Bahl & Jelinek$^+$ 83, Chen & Beeferman$^+$ 98, Klakow & Peters 02, Sundermeyer & Ney$^+$ 15a, Irie 20] and our theory in Section 3.4.5.

Considering Table 3.2 and Table 3.3 together, we can conclude that experimental results match our expectation, that all sampling-based criteria considered in this work perform equally well, when properly corrected and normalized.

### 3.6.3 Results without Explicit Normalization

In this section, we further experiment with the sampling-based training criteria without performing explicit normalization. This is the more interesting case because while in the previous section we showed that all sampling-based training criteria perform equally well when properly corrected and normalized, we may not always want to do this because of the complexity of the process and computational effort associated with it. Instead, ideally, we want to train with a self-normalized sampling-based criterion and not have to do anything extra at test time.

In Table 3.4, we give the results without explicit normalization on Switchboard. Here, we only consider the BCE sampling-based training criteria variants and drop the CE variants because, as discussed in Section 3.6.2, they do not seem to differ much. The training time and the normalized PPL columns are the same as in Table 3.2, and rewritten here for easy reference. Looking at the column of WER with no normalization of the language model,

Table 3.4: Sampling-based LSTM language models on Switchboard, with no explicit normalization after training. The 4-gram count-based Kneser-Ney language model is used to generate the first-pass lattices. The LSTM language models are interpolated with the count-based model for second-pass rescoring. For sampling-based methods, language model outputs are corrected (Section 3.4.5), normalized for perplexity (PPL) calculation, but not normalized for word error rate (WER) calculation.

| Model | Criterion | Sampling | Train Time (s/batch) | PPL (norm.) | WER (no norm.) | | |
|---|---|---|---|---|---|---|---|
| | | | | | SWB | CH | All |
| 4-gram | - | - | - | 74.6 | 8.1 | 15.4 | 11.8 |
| LSTM RNN | BCE | - | 0.107 | 52.3 | 6.9 | 13.7 | 10.3 |
| | | MCS | 0.077 | 52.6 | 7.3 | 14.7 | 11.0 |
| | | CPS | 0.079 | 52.4 | 7.5 | 15.1 | 11.3 |
| | | IS | 0.079 | 51.5 | 6.9 | 13.6 | 10.2 |
| | | NCE | 0.078 | 51.4 | 6.9 | 13.6 | 10.2 |
| | | SNIS | 0.090 | 51.7 | 6.9 | 13.6 | 10.2 |
| | CE | - | 0.100 | 49.9 | 6.8 | 13.4 | 10.1 |

Table 3.5: Sampling-based transformer language models on LibriSpeech, without explicit normalization after training. The well-tuned baseline LSTM RNN language model is used to generate the first-pass lattices. The transformer language models are used for second-pass rescoring. For sampling-based methods, language model outputs are corrected (Section 3.4.5), normalized for perplexity (PPL) calculation, but not normalized for word error rate (WER) calculation.

| Model | Criterion | Sampling | Training (s/batch) | PPL (norm.) | WER (no norm.) | |
|---|---|---|---|---|---|---|
| | | | | | clean | other |
| LSTM RNN | - | - | - | 64.3 | 2.6 | 5.8 |
| Transformer | BCE | - | 0.358 | 58.5 | 2.5 | 5.4 |
| | | NCE | 0.206 | 57.9 | 2.5 | 5.4 |
| | | SNIS | 0.216 | 58.3 | 2.5 | 5.4 |
| | CE | - | 0.302 | 57.7 | 2.5 | 5.4 |

we again see significant improvements over the count-based model, and importance sampling (IS) and self-normalized importance sampling (SNIS) perform on par with the strong BCE and CE baselines, as well as the well-established noise contrastive estimation (NCE) method. However, the monte carlo sampling (MCS) and compensated partial summation (CPS) variants performance is notably worse, and we think this is because, unlike IS, SNIS and NCE, MCS and CPS require querying the unreliable noise distribution $\mathcal{D}$ during search (in Section 3.4

Table 3.6: Sampling-based LSTM RNN language models on AppTek En, without explicit normalization after training. The 4-gram count-based Kneser-Ney language model is used to generate the first-pass lattices. The LSTM RNN language models are used for second-pass rescoring. For sampling-based methods, language model outputs are corrected (Section 3.4.5), normalized for perplexity (PPL) calculation, but not normalized for word error rate (WER) calculation.

| Model | Criterion | Sampling | $K$ | Training (s/batch) | PPL (norm.) | WER (no norm.) |
|---|---|---|---|---|---|---|
| 4-gram | - | - | - | - | 82.3 | 13.7 |
| LSTM RNN | BCE | NCE | 100 | 0.092 | 65.9 | 13.3 |
| | | | 8000 | 0.098 | 55.9 | 13.1 |
| | | SNIS | 100 | 0.089 | 68.0 | 13.3 |
| | | | 8000 | 0.114 | 55.8 | 13.1 |

and Section 3.4.5, $\mathcal{D}$ shows up in the solution for MCS and CPS, but not for IS, SNIS and NCE). When plugging in a smoothed empirical unigram distribution instead of the log-uniform distribution for $\mathcal{D}$, the WERs of MCS and CPS improve a bit, but are still behind the other sampling-based methods. Additionally, considering that IS requires a correction step but SNIS does not, in the following discussions, we only compare SNIS against the strong NCE method.

In Table 3.5 and Table 3.6, we show results of SNIS compared to NCE on larger-scale datasets, namely LibriSpeech and AppTek En. AppTek En is an in-house language modeling dataset for English telephony speech recognition, the size of which is similar to LibriSpeech for the neural language model training. As shown in Table 3.5, both SNIS and NCE achieve around 40% speedup compared to the BCE and CE baselines, and give similar normalized PPLs as well as WERs without explicitly normalizing the language model outputs, validating their self-normalization capabilities. In Table 3.6, the vocabulary size is around 250k, and the dataset is production-oriented. Here, we directly compared SNIS against NCE. When $K$ is 100, the speed of SNIS is slightly faster than NCE, but when $K$ is 8000, the speed of SNIS is slightly slower. If we do not attribute this observation to random noise, we think the slight slowdown at $K = 8000$ might be because of our implementation of the sampling process not being optimal, while the sampling implementation of NCE sampling is a stable one from TensorFlow [Abadi & Agarwal$^+$ 15]. With sampling size $K$ at 100 and 8000, the SNIS method is on par with the strong NCE baseline, both in terms of the normalized PPL and the WER without explicitly normalizing the language model.

## 3.7 Summary

In this chapter, we looked at the large vocabulary issue when training word-level neural network language models. We discussed three traditional training criteria, namely mean squared error, binary cross entropy and cross entropy, and moved on to derive where the model optimums are obtained of various sampling-based training criteria. Contrary to previous belief, we made the important observation that although some sampling-based training criteria are not

self-normalized, i.e. $\hat{q}_\theta(x, c) = p(c|x)$ by definition, it is possible to solve for $p(c|x)$ and obtain comparable performance against strong baselines. We further proposed a novel self-normalized version of importance sampling, which is a strong contestant compared to the popular and well-established noise contrastive estimation method. In a short detour, we discussed the self-normalization and variance regularization methods, both of which are useful for regularizing the denominator in the softmax function, and showed the relationship between normalization quality and regularization strength. Coming back to the topic of sampling-based training criteria, through extensive language modeling and automatic speech recognition experiments, we showed that the sampling-based training criteria are on par with binary cross entropy and cross entropy as well as the noise contrastive estimation baselines.

# 4. Smoothing In Neural Machine Translation

When training deep neural networks, it is well known that regularization typically helps with generalization and avoiding overfitting (e.g. see Chapter 7 in [Goodfellow & Bengio$^+$ 16]). In this chapter, we focus on regularization techniques for neural machine translation systems. Although one may argue that with more and more data being available every day, the need for regularization might decrease, we take the position that given the combinatorial nature of natural languages, it is infeasible to collect statistics for the exponentially many events, and regularization still has an important role in building strong models in practice. For instance, when we train a model to predict the period "." as the next word given an English sentence "Thank you ." and the partial German translation "Danke", who is to say that the next word cannot be "schön"[1]? This is a valid point, because translation (or natural language processing in general) is not a deterministic task, i.e. there may be several equally good translations given a certain source sentence. In practice, we are not able to collect all possible combinations of source and target sentences, which means some kind of smoothing is necessary in order to avoid "over-trusting" the limited training corpus we have, no matter how large it may be. The term "smoothing" is traditionally used in the context of regularization of count-based language models [Kneser & Ney 95, Chen & Goodman 96]. In this chapter, we use the term to refer to methods like label smoothing [Szegedy & Vanhoucke$^+$ 15, Gao & Wang$^+$ 20], soft contextualized data augmentation [Gao & Zhu$^+$ 19, Gao & Liao$^+$ 20] and multi-agent learning [Bi & Xiong$^+$ 19, Liao & Gao$^+$ 20] seen in the literature for the task neural machine translation. Intuitively, the three types of methods aim to regularize the model by answering the following questions:

- Given the source sentence and the partial translation so far, my training example shows that the next word should be this one - could it be other words as well?
- Given a specific source and target sentence pair, I know how to train the model with this specific data point - can I also train the model to be robust when the sentence pair is slightly altered?
- I know that I can train a pretty good translation model, but if I perform another training run, the results are similar but slightly different - can I make use of this fact and fuse the models into a better-performing model?

In this chapter, we will visit the details of these concepts, and show that with good smoothing setups, the translation performances of strong baseline systems can be further improved.

[1] In other words, translating into "Thank you very much ." instead of "Thank you ."; "schön" in German in this context can be roughly translated to "a lot".

## 4.1 Related Work

The label smoothing method was first introduced by Szegedy et al. [Szegedy & Vanhoucke$^+$ 16], where they introduced several refinements to the Inception model [Szegedy & Liu$^+$ 15]. The method is straightforward: instead of fitting the model to hard labels, they train the model to generate a distribution which is a weighted average of the one-hot distribution and a uniform distribution over the output classes. The motivation for this change is to avoid overfitting and improve generalization. Intuitively, the method can be thought of as discounting some probability mass from the true label and redistributing it uniformly across all class labels. In other words, when the model is overly confident in a certain label, label smoothing penalizes the model. The method was quickly adopted and widely applied in tasks like image recognition, speech recognition and machine translation [Szegedy & Vanhoucke$^+$ 16, Chorowski & Jaitly 16, He & Zhang$^+$ 18, Vaswani & Shazeer$^+$ 17, Zhou & Michel$^+$ 22]. Because information entropy [Shannon 48] is a confidence measure of a probability distribution, to regularize the model in the same direction as label smoothing, one can also add a negative entropy term to the training loss to discourage overly confident outputs. Pereyra et al. [Pereyra & Tucker$^+$ 17] described such a method and also proposed that alternative distributions can be used when smoothing the one-hot target distribution. In Müller et al. [Müller & Kornblith$^+$ 19], the authors observed that distilling knowledge from a teacher trained with label smoothing seemed to under-perform distilling from one trained without, and provided analyses from the perspective of the model output at the penultimate layer, arguing that information is lost in the logits but does not hurt the generalization and calibration of the model predictions. Along this line, Shen et al. [Shen & Liu$^+$ 21] provided in-depth analyses of the incompatibility between knowledge distillation and label smoothing training. Extending further discussions to the apparently contradictory results between Müller et al. [Müller & Kornblith$^+$ 19] and Shen et al. [Shen & Liu$^+$ 21], Chandrasegaran et al. [Chandrasegaran & Tran$^+$ 22] concluded that using teacher models trained with label smoothing with a low temperature during distillation leads to better student models. In Xu et al. [Xu & Xu$^+$ 20], the authors gave an analysis of the convergence behavior of stochastic gradient descent training with label smoothing, and proposed to deactivate the regularization after the training had ran for a certain number of update steps. Zhang et al. [Zhang & Jiang$^+$ 21] also studied label smoothing and proposed to replace the uniform prior distribution with the soft labels from the model itself in the last training epoch. From the perspective of label noise, i.e. observed labels may be sampled from a distribution different from the ground truth, Lukasik et al. [Lukasik & Bhojanapalli$^+$ 20] argued that although label smoothing introduces symmetric noises to the labels, it is related to the loss correction methods in the label noise literature.

Applying label smoothing during training has further implications for the model outputs, especially for sequence generation tasks. For instance, when examining the problem that machine translation systems tend to generate empty outputs when the beam size is very large [Koehn & Knowles 17], Shi et al. [Shi & Xiao$^+$ 20] argued that label smoothing introduces a length bias preferring the empty translation, i.e. the artificial end-of-sentence token having a higher probability at the first position. Another work that put forward the same argument is Liang et al. [Liang & Wang$^+$ 22], where the authors showed that the length bias introduced by using label smoothing leads to models preferring shorter translations. The authors then proposed to use the debiased model outputs during search, and showed through experiments that this mitigates the "short translation problem" when a large beam size is used.

In Xie et al. [Xie & Wang$^+$ 16], the authors pointed out that as regularization techniques, dropout [Srivastava & Hinton$^+$ 14b] can be thought of as ensembling different model architec-

tures on the same data and DisturbLabel [Xie & Wang+ 16] can be thought of as ensembling the same model architecture on different data. Similarly, label smoothing can be thought of as estimating the marginalized label dropout during training [Pereyra & Tucker+ 17]. In our work, we consider two natural extensions to label smoothing, namely token selection and prior distribution. Effectively, we are trying to answer two questions: 1. Which positions should we smooth? 2. Which prior distribution should we smooth with? The former is similar to Salimans et al. [Salimans & Goodfellow+ 16] and Zhou et al. [Zhou & Cai+ 18], where they selected only positive examples when smoothing generative adversarial networks. The latter is similar to Pereyra et al. [Pereyra & Tucker+ 17] and Gao et al. [Gao & Zhu+ 19], where alternative distributions other than the uniform distributions are considered.

In Kobayashi [Kobayashi 18], the author proposed to randomly sample from the output distribution of a language model to perform data augmentation on natural language sequential data. In Gao et al. [Gao & Zhu+ 19], the authors extended the idea to generate "soft" distributions. Their idea is straightforward, that is, when we feed a certain sentence pair to the translation model, we can augment our data such that effectively our model is trained on more combinations of input data. Specifically, they pre-train neural language models on the source and target side, randomly pick positions on the source and target input, and feed the corresponding language model posterior distribution instead of the usual one-hot distribution as input to the machine translation model. Although their method requires the training of external neural language models, their results show that significant translation improvements can be achieved, even compared to other strong data augmentation baselines [Artetxe & Labaka+ 18, Lample & Conneau+ 18, Iyyer & Manjunatha+ 15, Xie & Wang+ 17, Kobayashi 18]. In our work, we extend the concept and ask ourselves three questions: 1. Why only select certain positions and not smooth all input positions? 2. Do we have to use external neural models, i.e. would simpler distributions suffice? 3. Does the improvement from input smoothing and output smoothing, i.e. label smoothing, stack?

Another method of regularization is to ensemble several models[2] after training, which can be separately trained or co-trained. The simple log-linear combination is commonly seen in "The Conference of Machine Translation" (WMT) shared tasks [Barrault & Biesialska+ 20, Akhbardeh & Arkhangorodsky+ 21, Kocmi & Bawden+ 22]. Here, we are more interested in more sophisticated methods such as knowledge distillation and multi-agent learning. The concept of knowledge distillation was introduced in Buciluǎ et al. [Buciluǎ & Caruana+ 06] and gained more popularity after the publication of Li et al. [Li & Zhao+ 14] and Hinton et al. [Hinton & Vinyals+ 15]. The existing methods mainly differ on whether the agents are separately trained [Li & Zhao+ 14, Hinton & Vinyals+ 15, Meng & Li+ 18] or co-trained [Zhang & Xiang+ 18, Bi & Xiong+ 19], and if further ensembling is done before distilling knowledge from the teacher to the student [Bi & Xiong+ 19]. That said, the Kullback-Leibler divergence [Kullback & Leibler 51] and cross entropy are common to most methods as a metric of differences in distributions.

## 4.2 Notations

To clarify the notations, in this chapter:

- $n$ is a running index in total positions $N$.
- $A$ and $B$ are some disjoint partitions of the total positions $N$.

[2]Or agents. In this context, we will use both terms interchangeably.

- $x$ is some context in all possible contexts $X$.
- $c$ is some class in all possible classes $C$.
- $f_1^J$ is a source sentence, with $1, 2, ..., j, ..., J$ denoting the positions for each word.
- $e_1^I$ is a target sentence, with $1, 2, ..., i, ..., I$ denoting the positions for each word.
- $\tilde{f}_j$ is the word vector of the word $f_j$.
- $\tilde{e}_i$ is the word vector of the word $e_i$.
- $\delta$ is the Kronecker delta function.
- $p$ is the empirical distribution from the training data.
- $q_\theta$ is the model distribution, with learnable parameters $\theta$.
- $\hat{q}_\theta$ denotes the model outputs when the optimum is attained.
- $r$ is an auxiliary prior distribution given by some external helper model.
- $\lambda$ or $\lambda'$ is some quantity that can be interpreted as probability mass.
- $\gamma$ denotes the percentage of source/target input positions undergoing input smoothing.
- $k$ is a running index in total number of agents $K$.

When there is no need to specify the source/target dependencies, and only the training criterion is in question, we will use the $q_\theta(c|x)$ notation instead of the more complicated $q_\theta(e_i|f_1^J, e_0^{i-1})$, to simplify the notations.

## 4.3 Label Smoothing

### 4.3.1 Traditional Label Smoothing

The unregularized training of a one-out-of-$C$ classification system typically learns from data points with hard labels. That is, a target vector with $C$ elements with all zeros but a one at the correct label (a "one-hot" vector) is used as the target for the model to fit to. Label smoothing (LS) [Szegedy & Vanhoucke$^+$ 16] modifies this process, and discounts some probability mass from the correct label, and uniformly redistributes it across the vocabulary:

$$L_{\text{CE}} = -\frac{1}{N}\sum_{n=1}^{N}\log q_\theta(c_n|x_n) \tag{4.1}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\delta(c, c_n)\log q_\theta(c|x_n)$$

$$L_{\text{LS}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\left((1-\lambda)\delta(c, c_n) + \lambda\frac{1}{C}\right)\log q_\theta(c|x_n) \tag{4.2}$$

Here, $\delta(c, c_n)$ is effectively the one-hot distribution, when we sum over $C$. As seen, when $\lambda$ goes from zero to one, the training criterion in Equation 4.2 goes from the traditional cross entropy

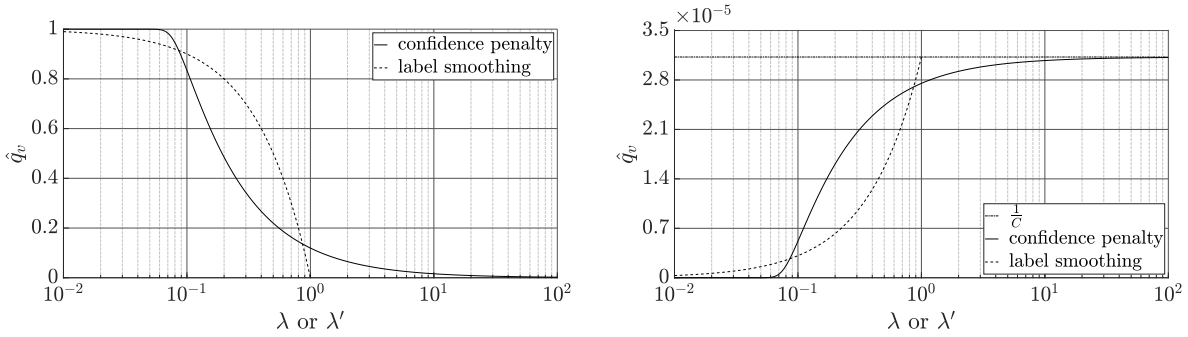
to an objective where the model should predict a uniform distribution regardless of the context. From the derivations in Chapter 3, we know that with cross entropy, the model optimum is attained when $\hat{q}_\theta(c|x) = p(c|x)$ (Note that in Chapter 3 the softmax function was explicitly written out to highlight the summation in $C$, but here it is hidden inside of $q_\theta(c|x)$). Due to the linear nature of the label smoothing formulation, following a similar derivation to earlier, we find that with label smoothing, the model optimum is attained at the linear interpolation between the empirical distribution and the uniform distribution, i.e. $\hat{q}_\theta(c|x) = (1-\lambda)p(c|x) + \lambda\frac{1}{C}$.

### 4.3.2 Confidence Penalty

Since the regularization direction is towards the uniform distribution, we can use the information entropy to achieve similar regularization effects, i.e. with confidence penalty [Pereyra & Tucker$^+$ 17]:

$$L_{\text{CP}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\Big(\delta(c, c_n) - \lambda' q_\theta(c|x_n)\Big)\log q_\theta(c|x_n) \tag{4.3}$$

We know that the information entropy takes its maximum at the uniform distribution, and that means when $\lambda'$ goes from zero to infinity, the training criterion goes from the cross entropy to an objective that almost exclusively trains the model towards the uniform distribution. Note that this point was known and was exactly the motivation of Pereyra et al. [Pereyra & Tucker$^+$ 17], but they did not give an exact solution to the training problem.




(a) smoothing effect on the correct label

(b) smoothing effect on the other labels

Figure 4.1: Graphs of when model optimums are attained, i.e. $\hat{q}$, with respect to $\lambda$ or $\lambda'$, for label smoothing and confidence penalty. The horizontal axes are logarithmic scale, with $\lambda \in [0, 1]$ and $\lambda' \geq 0$. To obtain the numerical solutions, we set $C$ to be 32k, which is a common vocabulary size for machine translation systems [Vaswani & Shazeer$^+$ 17].

In our work [Gao & Wang$^+$ 20], we showed that, ignoring the outer summation in $N$ and only considering a certain local $n$, it is possible to introduce the Lagrange multiplier $\lambda_0$ to solve for the optimum, but a transcendental equation would show up. It is then possible to make use of the Lambert $W$ function [Corless & Gonnet$^+$ 96] to express the local solution:

$$\hat{q}_\theta(c; n) = \frac{\delta(c, c_n)}{\lambda' W_0\left(\frac{\delta(c,c_n)}{\lambda'} e^{1+\frac{\lambda_0}{\lambda'}}\right)} \tag{4.4}$$

Here, $W_0$ denotes the principal branch of the Lambert $W$ function and $\lambda_0$ is the Lagrange multiplier. The detailed derivation can be found in the Appendix of Gao et al. [Gao & Wang$^+$ 20]. It is possible to numerically solve the equation above given non-negative $\lambda'$ and $\delta(c, c_n)$ using solvers for symbolic mathematics [MathWorks 23]. To give an intuitive understanding of how label smoothing and confidence penalty relate to and differ from one another, we ignore the outer summation in $N$ again, and plot the graphs of when model optimums are attained at, i.e. $\hat{q}$, with respect to $\lambda$ or $\lambda'$, for the two methods. As seen in Figure 4.1, both methods have similar curves, i.e. pushing down the model output at the correct label from one to zero (Figure 4.1a) and bringing up the model output at the other labels from zero to $\frac{1}{C}$ (Figure 4.1b), with increasing $\lambda$ and $\lambda'$.

### 4.3.3 Extensions

In the context of label smoothing, it makes sense to consider two questions:

1. Which positions should we smooth? How do we select them?
2. With which auxiliary distribution and how should we smooth the model?

This naturally leads to the following Extended Label Smoothing (ELS) training criterion:

$$L_{\text{ELS}} = -\frac{1}{N}\left(\sum_{n\in A}\log q_\theta(c_n|x_n) + \sum_{n\in B}\sum_{c=1}^{C}\big((1-\lambda)\delta(c,c_n) + \lambda r_c\big)\log q_\theta(c|x_n)\right) \tag{4.5}$$

Here, $A$ and $B$ are two disjoint subsets of the total target positions $N$, and $r_c$ is an auxiliary prior distribution over the vocabulary $C$ which is given by an external helper model, be it a uniform distribution or more complex one.

Regarding the first question, we consider two aspects. First, we make the distinction between randomly selecting positions and preferring those positions that the model is most uncertain about (those positions with higher information entropy). Second, we ask the question of how many positions should be smoothed? Should it be all target positions or some subset of them?

Regarding the second question, we also consider two aspects. First, we explore different prior distributions and see if there is anything to be gained beyond simple uniform prior distributions. Second, we sweep over different values of $\lambda$ in Equation 4.5, to see if there is a better hyperparameter beyond the usual $\lambda = 0.1$ used in many existing recipes.

## 4.4 Input Smoothing

### 4.4.1 Background and Formulation

As mentioned in Section 1.4.2, the step of embedding discrete tokens into continuous-valued vectors (i.e. for source side: $\tilde{f}_1, \tilde{f}_2, ..., \tilde{f}_j, ..., \tilde{f}_J$; for target side: $\tilde{e}_0, \tilde{e}_1, ..., \tilde{e}_i, ...\tilde{e}_{I-1}$) via the embedding matrix is ubiquitous in state-of-the-art neural-network-based machine translation models. Because only one word index corresponds to a token, essentially we perform a lookup operation in the matrix to obtain the word vector. In Kobayashi [Kobayashi 18] and Gao et al. [Gao & Zhu$^+$ 19], this norm is questioned and an alternative weighted-sum approach is proposed.

Shown in Figure 4.2 is an illustration of the input smoothing method. Assume that we have a source sentence "I like reading ." and select the third position for smoothing. We query

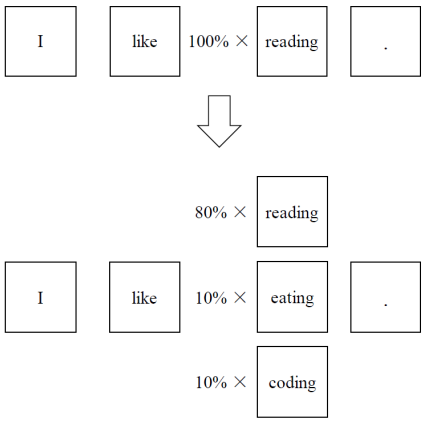



Figure 4.2: Illustration of input smoothing. The third word "reading" in the example sentence is selected for smoothing. Assume that the auxiliary prior distribution (denoted $r_c$) from some external helper model (e.g. a language model) gives 80%, 10% and 10% probability masses on three candidate words: "reading", "eating" and "coding". The input word vector of "reading" is then replaced with a weighted sum over the word vectors of these candidate words with the corresponding weights.

an external helper model for the auxiliary prior distribution $r_c$, and replace the original word vector in that position with a linear interpolation over all word vectors weighted by the auxiliary prior distribution. Here, the arbitrary prior distribution $r_c$ can be a one-hot distribution (which will default to the usual embedding-matrix-lookup case), a uniform distribution, the posterior distribution from an external language model, etc. The positions to smooth are randomly selected, and the word vector lookup procedure for those unselected positions is left intact. The original authors coined the name "soft contextualized data augmentation" for the method. In this dissertation, we simply call such methods "input smoothing" because essentially the previously hard input labels are replaced by soft input labels, which is somewhat similar to label smoothing, where hard one-hot target output labels are replaced by softer distributions.

The weighted-sum computation is also efficient in practical cases (when $C$ is not very large, which is the case for most neural machine translation system nowadays because they operate at the subword level), because it can be simply implemented as matrix multiplications. The immediate downside of the method is only the need to obtain the $r$ distribution, and potentially more memory usage because of the need to store and calculate with $r$. As will be shown later, the percentage of source/target positions that undergo input smoothing has a large impact on the translation performance; in this chapter, we devote a separate letter $\gamma$ to denote this quantity.

### 4.4.2 Extensions

In Gao et al. [Gao & Zhu$^+$ 19], the choice of $r$ is limited to posterior distributions from neural language models separately trained on the monolingual sides of the parallel data. They randomly select 15% of source/target input positions and do not enforce any "strength" of the smoothing (in analogy to $\lambda$ for label smoothing), but simply replace the one-hot distribution

completely with the prior distribution whenever the position is selected. Compared to label smoothing, aesthetically, the "symmetric" way of smoothing the inputs should be smoothing all input positions but with a hyperparameter controlling the smoothing strength: $\lambda$. In our work, we make this extension, and study whether the 15% position selection is necessary, and if a smoothing strength or mass concept makes any difference in practice. Specifically, for a certain selected input position, $\lambda$ controls to what extent the soft distribution contributes to the word vector.

Another natural extension is to explore alternative auxiliary prior distributions for $r$. While conceptually, contextual distributions from external language models are more expressive, we do need to pay the price of pre-training the language models, as well as keeping the language models and the posterior distributions in memory (on-the-fly implementation) or on disk (pre-calculated). In this work, we consider simpler and consequently cheaper distributions such as uniform and unigram distributions, as well as even more expressive distributions such as the posterior distribution from a back-translation model or a pre-trained BERT [Devlin & Chang$^+$ 19] model[3].

A final question we consider is whether or not the improvements from input smoothing and output smoothing stack. This is an important question because many new tricks and methods are published on a daily basis, with each one claiming somewhere around absolute +1% BLEU improvements. If all these tricks are orthogonal and bring incremental improvements, we would have had BLEU scores larger than 100% a long time ago. That said, it is not always an easy task, because of the combinatorial nature of the methods. Luckily, the input smoothing and output smoothing methods we discuss here are orthogonal by definition, and somewhat similar to each other, and therefore we expend the effort to answer the question.

## 4.5 Multi-Agent Mutual Learning

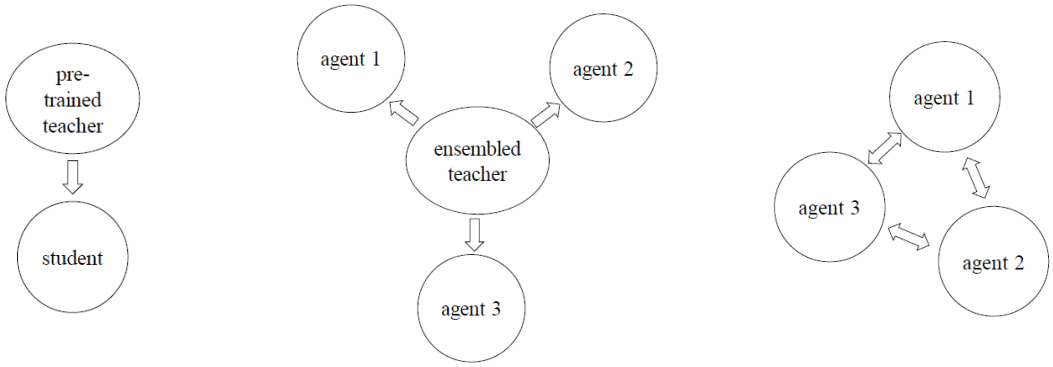
### 4.5.1 Background

When we have several probabilistic models for the same task, it is generally beneficial to ensemble the models to arrive at better predictions (see Chapter 16 in Hastie et al. [Hastie & Tibshirani$^+$ 09]). For the task of machine translation:

- If the models are of a heterogeneous nature, reranking or log-linear combination are typically used [Barrault & Biesialska$^+$ 20, Akhbardeh & Arkhangorodsky$^+$ 21, Kocmi & Bawden$^+$ 22].
- If the models are of a homogeneous nature, checkpoint averaging can be used [Vaswani & Shazeer$^+$ 17, Liu & Zhou$^+$ 18, Gao & Herold$^+$ 22b].
- If inter-model communication is further allowed, knowledge distillation and multi-agent learning can be of use [Hinton & Vinyals$^+$ 15, Wang & Yan$^+$ 21, Bi & Xiong$^+$ 19].

The core idea of multi-agent learning is to enable information flow among the agents during training (see Figure 4.3 for an intuitive understanding). This can be done e.g. in a two-step process, where an initial good teacher model is built, and then used to teach different agents [Bi & Xiong$^+$ 19], or via a dynamic mutual learning process, where the information flow among the agents happens during the from-scratch training [Liao & Gao$^+$ 20]. If we take the metaphor of

[3] After the publication of our work [Gao & Liao$^+$ 20], there was also a similar work [Cheng & Huang$^+$ 22], which explores the usage of BERT posteriors in input smoothing.




(a) knowledge distillation [Hinton & Vinyals+ 15]

(b) multi-agent training [Bi & Xiong+ 19]

(c) multi-agent mutual learning [Liao & Gao+ 20]

Figure 4.3: Illustration of multi-agent mutual learning. In traditional knowledge distillation [Hinton & Vinyals+ 15], pre-training a large teacher and distilling to a smaller student happens in two steps, with the goal of obtaining a smaller model with good performance. In multi-agent training [Bi & Xiong+ 19], multiple agents learn from an ensembled teacher. In our multi-agent mutual learning setup [Liao & Gao+ 20], the learning among the agents happens throughout the training process.

rolling a ball down the hills when training one agent using gradient descent, a similar metaphor for our multi-agent learning would be to roll several balls down the hills at the same time, but with strings attached to one another during the process, such that the movement of one ball also has influence on other balls. Technically speaking, the string metaphor may actually not be a bad one, because strings imply elasticity and potentially oscillation, both of which could be happening during the gradient learning process. Specifically, the "distance" measure of the balls can be done via the Kullback-Leibler divergence [Kullback & Leibler 51] or the cross entropy, because we are measuring the distances between model output probabilistic distributions. The oscillation aspect may exist because we only provide gradient signals dragging the balls together, but do not enforce any boundaries to avoid "overshooting".

### 4.5.2 Formulation

Denoting the model outputs as $q_{\theta_{k_1}}$ and $q_{\theta_{k_2}}$ and the respective label-smoothed cross entropy losses as $L^{\text{LS}}_{k_1}$ and $L^{\text{LS}}_{k_2}$ for two agents $k_1$ and $k_2$, the sentence-level inter-agent training objective can be written as:

$$L^{\text{SEN}}_{k_1,k_2} = \begin{cases} H(q_{\theta_{k1}}, q_{\theta_{k2}}), & \text{if} \quad L^{\text{LS}}_{k_1} \leq L^{\text{LS}}_{k_2} \\ H(q_{\theta_{k2}}, q_{\theta_{k1}}), & \text{else} \end{cases} \tag{4.6}$$

$$\text{where} \quad H(g,h) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} g(c|x_n)\log h(c|x_n)$$

In other words, we always use the better performing agent as the teacher in the sentence-level inter-agent training objective.

In a mixture of $K$ agents, for a certain agent $k$, in addition to the original label-smoothed cross entropy, all agent pairs are taken into consideration:

$$L_k^{\text{SEN}} = \lambda^{\text{SEN}} L_k^{\text{LS}} + (1 - \lambda^{\text{SEN}}) \frac{1}{K-1} \sum_{k'=1, k' \neq k}^{K} L_{k,k'}^{\text{SEN}} \tag{4.7}$$

Here, $\lambda^{\text{SEN}}$ controls the balance between learning from the data or from the other agents, and we can use static values or a dynamically decreasing $\lambda^{\text{SEN}}$ to gradually shift the training towards inter-agent learning, because at the start of the training, the training signal from other agents may not be very good, e.g.:

$$\lambda^{\text{SEN}} = 0.5 + \frac{0.5}{\#\text{epoch}} \tag{4.8}$$

After the initial convergence with the sentence-level objective, we further consider distilling knowledge among agents at the token-level. That is, for each agent, we consider it in conjunction with each of the other agents, and make decisions over our corpus, answering the question which agent does better on which token, in order to arrive at two disjoint subsets of all target positions. Specifically, for a certain sentence pair, and between two agents $k_1$ and $k_2$, we first determine the maximum ratio between the outputs over all positions. Then for each position $i$, we compare random dice rolls against the ratio between the current model outputs' ratio and this maximum to assign different positions $i$ into the two subsets (Algorithm 1 in Liao et al. [Liao & Gao$^+$ 20]). This process will yield two disjoint subsets $S_{k_1,k_2}$ and $S_{k_2,k_1}$, with the ordering of the agent indices denoting "who teaches whom" on this subset.

The token-level inter-agent objective then loops over all agent pairs as well as all "to-be-learned" target positions:

$$L_k^{\text{TOK}} = \lambda^{\text{TOK}} L_k^{\text{LS}} + (1 - \lambda^{\text{TOK}}) \frac{1}{K-1} \sum_{k'=1, k' \neq k}^{K} \left( - \frac{1}{\|S_{k',k}\|} \sum_{i \in S_{k',k}} \sum_{c=1}^{C} q_{\theta_{k'}}(c|x_i) \log q_{\theta_k}(c|x_i) \right) \tag{4.9}$$

The equation above appears complex, but expresses a very simple logic: for each agent $k$, in addition to the original label-smoothed cross entropy, we visit all other agents $k'$, normalize by the total number of tokens $\|S_{k',k}\|$ that we need to learn from $k'$, visit all those to-be-learned tokens $i$, use a local cross entropy objective, and balance between learning from data and learning from other agents with the hyperparameter $\lambda^{\text{TOK}}$. Because this token-level optimization is done after each agent is already trained pretty well, we use a static weight: $\lambda^{\text{TOK}} < 0.5$, i.e. focusing on learning the poorly predicted tokens from other agents.

## 4.6 Experimental Results

### 4.6.1 Label Smoothing

#### Token Selection and Prior Distribution

In our investigation of label smoothing in neural machine translation, we generally aim to arrive at a good training recipe. In this section, we are particularly interested in the optimal setup for token selection and prior distribution. To this end, we first consider the differences

between randomly selecting target positions and preferring those target positions where the model has a large information entropy, i.e. those positions that the model is most uncertain about. We sweep over different values of $\frac{\|B\|}{N}$, going from "smoothing none" to "smoothing all", with the two strategies.

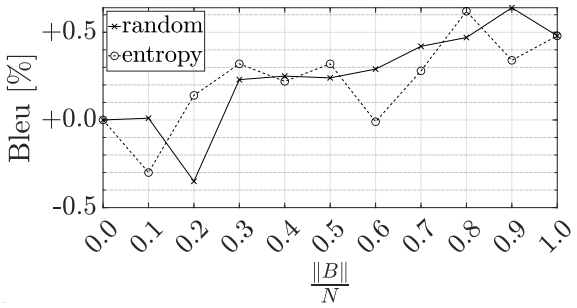



Figure 4.4: Label smoothing by randomly selecting target positions (random) or selecting target positions based on information entropy (entropy, "uncertain positions first"), on IWSLT2014 German-English. We sweep from "smoothing no positions" to "smoothing all positions", and absolute Bleu [%] improvements compared to the no label smoothing baseline are shown. The prior distribution for label smoothing comes from a pre-trained target-side neural language model, with a perplexity of 46.5 on the target text of the test data.

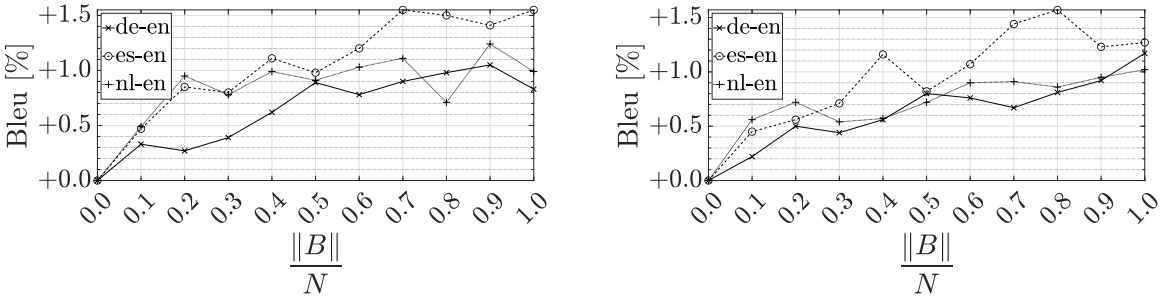
Shown in Figure 4.4 is the result. We make three observations:

- First, there is an overall increasing trend, showing that smoothing more target positions is beneficial.
- Second, there does not seem to be a significant difference between the two selection strategies, showing that the benefit from label smoothing is likely to be target-position- (and therefore target-word-) independent.
- Third, at $\frac{\|B\|}{N} = 0$ and $\frac{\|B\|}{N} = 1$, i.e. "smoothing none" and "smoothing all", the two curves coincide at the same points, which serves as a sanity check because the token selection strategy should not have an effect on the translation performance when none or all of the positions are selected.

These observations indicate that trying to be "smart" about the token selection process does not give substantial improvements. Rather, one should simply stick with the "smooth all target positions" strategy.

Notice that in Figure 4.4, we used a pre-trained external neural language model, which is trained on the target-side of the parallel data, as the helper model to retrieve the auxiliary prior distributions. In our experiments, we observed that using complex helper models underperforms using simpler prior distributions such as the uniform distribution and the unigram distribution. The third observation made above can to some degree explain this phenomenon. That is, if improvements from label smoothing are really target-position-independent, then the more complex models naturally do not bring much more to the table, because they are contextualized by nature.

For the reasons above, we further perform similar sweeping experiments on more datasets using simpler prior distributions, i.e. uniform and unigram distributions. In Figure 4.5, we

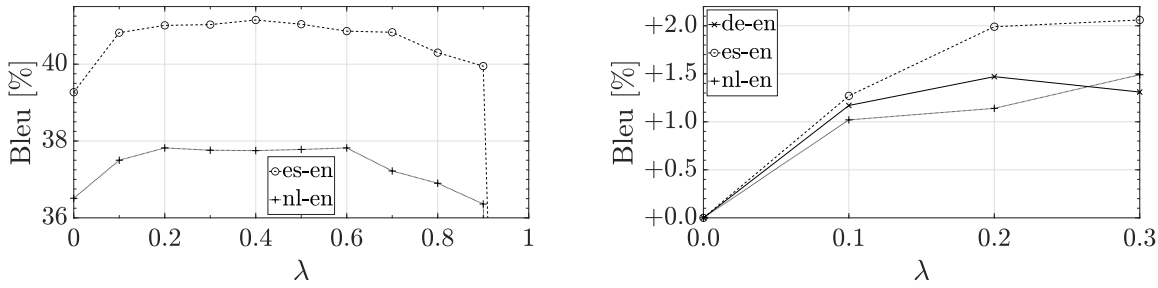



(a) uniform prior (b) unigram prior

Figure 4.5: Label smoothing by sweeping from "smoothing no positions" to "smoothing all positions" on IWSLT2014 German-English (de-en), Spanish-English (es-en) and Dutch-English (nl-en). The smoothed positions are randomly selected. The Y-axis shows the absolute Bleu [%] improvements compared to the no label smoothing baseline for the three translation directions, respectively. Simple prior distributions are used here, i.e. uniform distribution for (a) and unigram distribution for (b).

can see that the overall upward trends of the curves remain. In addition, the difference among the three translation directions is not big, mostly probably because of the fact that the three datasets are of similar domain and size. Finally, the difference between uniform and unigram priors is not significant, and we will see later, that the key hyperparameter for the overall performance is in fact the discounted probability mass $\lambda$.

### Probability Mass




(a) uniform prior (b) unigram prior

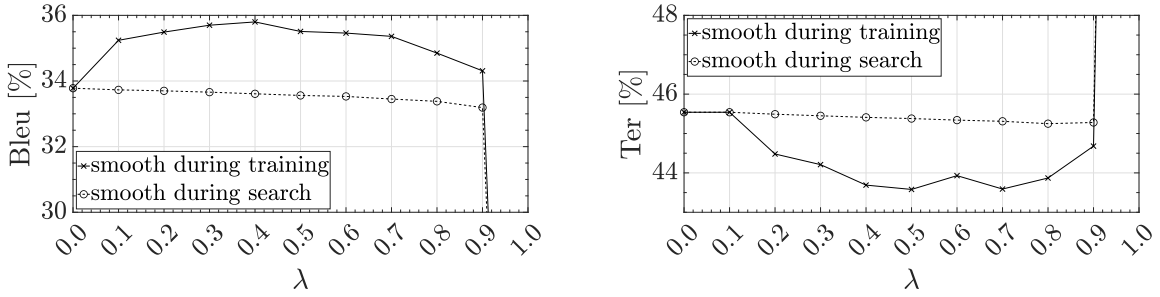
Figure 4.6: Label smoothing with different discounted probability masses $\lambda$, on IWSLT2014 German-English (de-en), Spanish-English (es-en) and Dutch-English (nl-en). (a): smoothing with uniform prior, and the Y-axis shows absolute Bleu [%] scores. (b): smoothing with unigram prior, and the Y-axis shows absolute Bleu [%] improvements (compared to the baseline without label smoothing).

As mentioned, we found that the key factor for good translation performance is the discounted probability mass $\lambda$. To reveal this point, it makes sense to train systems with different $\lambda$ settings. In Figure 4.6, we experiment with both the uniform and the unigram prior distributions, and vary $\lambda$ from zero to one.

As seen, when label smoothing is initially activated, there are significant improvements compared to the baseline without smoothing. Then, a wide plateau happens. This suggests that the neural network model is very good at picking out the correct training signals (the empirical translation posterior distributions) even when they are buried in much stronger noise signals (uniform or unigram). Only when $\lambda$ goes to extreme values close to one does the learning process break, which is expected, because we know that when $\lambda$ is strictly one, the model is trained to generate uniform distributions. In fact, inspection of the training logs shows that the model quickly learns the dummy task and gets stuck there (meanwhile learning nothing about the translation task, of course).

### Smoothing During Search

As mentioned in Section 4.3.1, we know that when the simple uniform prior is used in label smoothing during training, the training objective simply drags the model to generate a weighted average of the empirical distribution and the uniform distribution. Then, a natural question to ask is, why do we even bother? That is, one could simply do non-label-smoothing-regularized training, and manually adjust the model output during the beam search process according to the linear solution. If the model is trained perfectly towards the simple task of weighted average, then we should expect to see comparable performance with such a straightforward post-training-modification trick.




(a) Bleu[%] (b) Ter[%]

Figure 4.7: Label smoothing during training (only) versus label smoothing during search (only), on IWSLT2014 German-English. The uniform distribution is used as the prior and we sweep $\lambda$ from zero to one. Bleu[%] and Ter[%] results are plotted in (a) and (b), respectively. This set of experiments contains different training runs and the results are not directly comparable to other results presented so far in this chapter.

We implement the aforementioned idea and present the results in Figure 4.7. As expected, the solid curve in Figure 4.7a is similar to that in Figure 4.6a. However, examining the dotted curve, we observe a different trend. That is, with increasing $\lambda$, the Bleu score does not improve. We suspect that in hard-discounting, probability mass from the "argmax" label in each target position and redistributing them equally to other positions[4], we are not really altering the relative order of the output probability masses on different labels. In fact, assume for a certain position, the model assigns $q_1$ on the most-probable label and $q_2$ on the second-most-probable label, we need $\lambda > (q_1 - q_2)/(1 + 1/C)$ to change their order, in order to locally affect the beam search process. When $C$ is sufficiently large, this roughly corresponds to a $\lambda$ larger than the

[4]In different implementations, it could be redistribution to "all positions" or "all other positions". However, one can show that this simply corresponds to a redefinition of $\lambda$, which does not affect this discussion.

absolute difference between $q_1$ and $q_2$, which is probably the reason why we see a relatively flat dotted curve in Figure 4.7a.

Shifting our attention to Figure 4.7b, we observe a small contradiction between the BLEU and TER trends in the dotted curves. This can be explained by the changes in the lengths of the hypotheses. That is, with increasing $\lambda$, we see shorter and shorter hypotheses being generated. Considering that BLEU has a length penalty term, penalizing shorter hypotheses, while TER has a tendency to prefer shorter hypotheses[5], the small-sloped downward trend in the TER scores is not that surprising.

To quickly summarize, this set of experiments indicates that label smoothing is in fact a complicated regularization process, and the model trained with label smoothing via a stochastic gradient process does not simply generate a weighted average between the empirical distribution and the prior distribution. To preserve the improvements from label smoothing, one cannot simply train a non-smoothed model and make adjustments to model outputs during beam search.

### Improved Recipe

Table 4.1: BLEU [%] scores of systems trained without label smoothing, with default transformer label smoothing, and with our best recipe. It is verified that label smoothing brings decent improvements over the baseline without label smoothing, and further tuning the prior distribution as well as the discounted probability mass brings significant improvements over the default transformer setup.

| label smoothing | BLEU [%] | | | |
|---|---|---|---|---|
| | IWSLT2014 | | | WMT2014 |
| | de-en | nl-en | es-en | en-de |
| none | 34.5 | 37.3 | 40.5 | 28.0 |
| uniform prior, $\lambda$=0.1 [Vaswani & Shazeer$^+$ 17] | 34.9 | 37.7 | 41.3 | 28.4 |
| unigram prior, $\lambda$=0.3 (ours) | 35.6 | 39.1 | 41.5 | 29.0 |

After discussions on the token selection, prior distribution, discounted probability mass and whether or not to shift the label smoothing objective from training to search, we are in a position to discuss our improved training recipe. Overall, we find that label smoothing is a simple and effective method to boost the generalization capability of a neural translation model. Despite improvements from a simple uniform prior with $\lambda = 0.1$ setup, we found that smoothing with a unigram prior with $\lambda = 0.3$ consistently gives better performance, with $\lambda$ being the stronger factor affecting the end results. The results[6] are therefore summarized in Table 4.1. Notably, the improvements using our recipe that is carefully tuned on the small

[5] An easy way to think about it is to imagine two hypotheses with completely garbage words, with one being shorter than the reference and the other being much much longer than the reference. TER will assign 100% on the first hypothesis, while exceeding 100% on the second. Note that this is not a strict proof, but more leaning towards an upper bound of the error rates, because real hypotheses are not always fully made up of garbage words.

[6] The IWSLT results are from Gao et al. [Gao & Liao$^+$ 20], where improved baselines over that in Gao et al. [Gao & Wang$^+$ 20] are used. The WMT results are from Gao et al. [Gao & Wang$^+$ 20]. The results using the default transformer label smoothing are rerun by us and not numbers taken directly from the original paper [Vaswani & Shazeer$^+$ 17].

IWSLT2014 German-English dataset also transfer to other language pairs, as well as to the larger WMT2014 English-German dataset with the transformer big architecture.

### 4.6.2 Input Smoothing

#### Auxiliary Distribution and Token Selection Strategy

Similar to label smoothing, three hyperparameters are of interest when applying input smoothing to machine translation (MT) models:

- Auxiliary distribution $r$, which could be as simple as a uniform distribution, or as complex as the posterior distribution from an external BERT [Devlin & Chang$^+$ 19] model.
- Percentage of positions to smooth $\gamma$ (and of course how to select those positions), which corresponds to $\frac{\|B\|}{N}$ in label smoothing.
- Smoothing strength $\lambda$, which corresponds to the discounted probability mass in label smoothing.

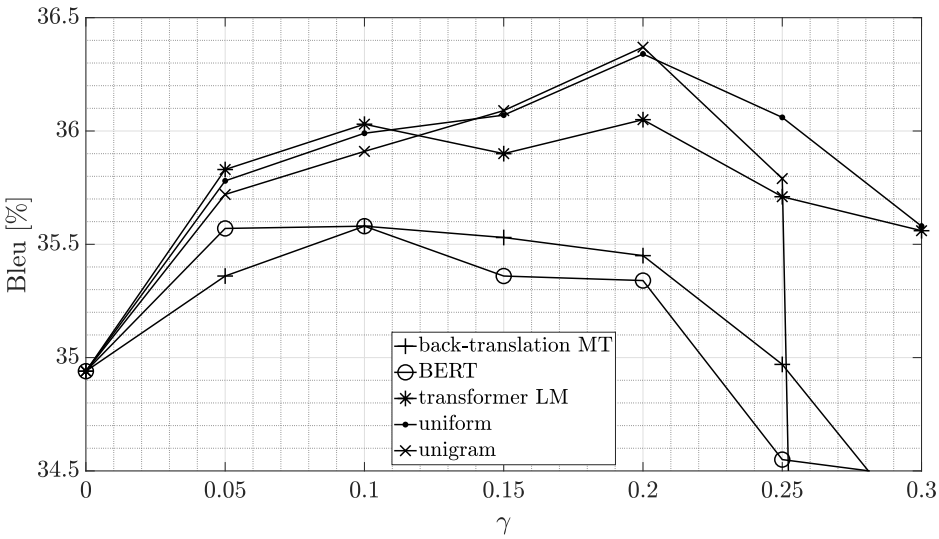
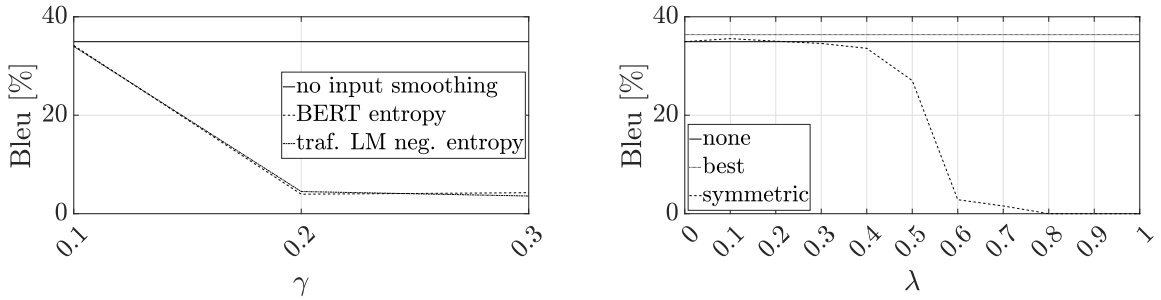



Figure 4.8: Input smoothing with different prior distributions, on IWSLT2014 German-English. $\gamma$ refers to the percentage of randomly selected positions that undergo input smoothing. Except for "back-translation MT", where input smoothing is only applied to the source inputs, in the other four cases, input smoothing is always applied to both source and target inputs. This figure can be thought of as an extension of Figure 2 in Gao et al. [Gao & Zhu$^+$ 19].

To decide good setups for these hyperparameters, we first extend Figure 2 in Gao et al. [Gao & Zhu$^+$ 19], by experimenting with alternative prior distributions other than those that come from external language models that were used in Gao et al. [Gao & Zhu$^+$ 19]. In Figure 4.8, five types of prior distributions are considered: from a pre-trained back-translation machine translation model and only applied to the source side, or from BERT [Devlin & Chang$^+$ 19], transformer language model, uniform distribution, unigram distribution, and applied to both the source and the target side (using the helper model at the corresponding side). In these

experiments, whenever an input position is randomly selected, the one-hot input is completely replaced by the soft prior distribution, and does not undergo a weighted sum. When sweeping the percentage of positions $\gamma$ to smooth, we quickly see that initial improvements are achieved with all prior distributions. However, when too large value is used for $\gamma$, degradation starts to appear. Overall, we verify the results from Gao et al. [Gao & Zhu$^+$ 19] that significant improvements can be achieved with smoothing, using external transformer language models. More complex models such as a pre-trained back-translation model or a BERT model which is even trained on additional data do not seem to bring any additional benefit. That said, simpler distributions such as the uniform or the unigram distribution with $\gamma = 20\%$ can bring slightly more improvements compared to the neural language models, while having the additional benefit that we do not need to pre-train and maintain external neural language models.




(a) Prioritizing the most (un)certain input tokens hurts input smoothing.

(b) Smoothing all input positions with controllable strength $\lambda$ hurts input smoothing.

Figure 4.9: Investigation into token selection strategies of input smoothing, on IWSLT2014. In (a), "BERT entropy" and "traf. LM neg. entropy" means that we first rank the input positions with the information entropy or negative information entropy of an external BERT or transformer language model, then we prioritize the top-ranked positions when performing input smoothing. In (b), "none" and "best" refer to the baselines without and with the best input smoothing setup, and "symmetric" refers to the case where all input positions are smoothed, but controlled with a hyperparameter $\lambda$ (symmetric to label smoothing, hence the name).

We then move on to answer the questions about the token selection strategy and the smoothing strength $\lambda$.

In Figure 4.9a, we rank the input positions with either the information entropy of the BERT model, or the negative information entropy of the transformer language model, on the source and the target side respectively. Then, when a certain percentage $\gamma$ is given, we prioritize the top positions according to the rank to smooth. Intuitively, this corresponds to prioritizing positions that the helper model is most (un)certain about. However, as Figure 4.9a shows, both ideas under-perform even when compared to the no input smoothing baseline. In other words, either focusing on easy word positions like "the" or focusing on hard word positions like "dog" versus "cat" (e.g. continuing the context "A common home pet is") does not give any benefit. Rather, randomly selecting input positions and introducing "background" noise seems to be a better option in this case. Note that we fix $\lambda = 1$ in the above-mentioned experiments.

When considering a symmetric case to label smoothing at the output side, i.e. smoothing all positions by tuning the smoothing strength $\lambda$, the result is also underwhelming. As shown in Figure 4.9b, a small initial improvement is obtained when $\lambda$ is small, but the overall performance drops very quickly as larger $\lambda$ values are used. In the extreme cases, e.g. when $\lambda = 0$ or $\lambda = 1$,

we observe either no difference to the "no input smoothing" baseline or completely garbage translation (because essentially no source sentence is given) respectively, both of which are to be expected. Despite the small improvement at small $\lambda$ values, the symmetric approach lags behind when compared to the best "randomly pick some positions, but smooth strongly with $\lambda = 1$" baseline. Therefore, we move on to study the combined effect between input smoothing and label smoothing, without further enforcing the aesthetically more attractive symmetric approach.

### In Conjunction With Label Smoothing

Table 4.2: Combining input smoothing with label smoothing, varying the auxiliary distribution, on IWSLT2014 German-English (de-en), Dutch-English (nl-en) and Spanish-English (es-en). BLEU [%] scores on the test are reported. Except for row 4 and 10, i.e. when using an external BERT model, only the parallel data is used. The asterisk in "uniform*" indicates that the label smoothing hyperparameter $\lambda$ is fixed at 0.1, in order to be comparable with the original transformer setup [Vaswani & Shazeer$^+$ 17]. Otherwise, we tuned the $\gamma$ and $\lambda$ hyperparameters for each setup on the de-en dataset and applied the best setup on nl-en and es-en.

| row id | auxiliary distribution | | | IWSLT2014 | | |
|---|---|---|---|---|---|---|
| | target output | source input | target input | de-en | nl-en | es-en |
| 1 | - | - | - | 34.5 | 37.3 | 40.5 |
| 2 | | back-translation MT | | 34.5 | 37.5 | 40.8 |
| 3 | | transformer LM | transformer LM | 34.6 | 37.3 | 41.0 |
| 4 | | BERT | BERT | 34.5 | 37.4 | 40.5 |
| 5 | | uniform | uniform | 34.8 | 37.9 | 41.0 |
| 6 | | unigram | unigram | 35.1 | 37.6 | 41.1 |
| 7 | uniform* | - | - | 34.9 | 37.7 | 41.3 |
| 8 | | back-translation MT | | 35.6 | 38.1 | 41.5 |
| 9 | | transformer LM | transformer LM | 36.1 | 38.2 | 41.6 |
| 10 | | BERT | BERT | 35.6 | 38.3 | 41.4 |
| 11 | | uniform | uniform | 36.3 | 38.9 | 42.4 |
| 12 | | unigram | unigram | 36.4 | 39.1 | 42.2 |
| 13 | unigram | - | - | 35.6 | 38.1 | 41.5 |
| 14 | | unigram | unigram | 36.3 | 39.1 | 42.4 |

In order to consider input smoothing in conjunction with label smoothing, we first experiment with extensive settings on three small IWSLT2014 datasets: German-English (de-en), Dutch-English (nl-en) and Spanish-English (es-en). In Table 4.2, the results are presented. First, if we consider rows 1 to 6, results similar to those in Figure 4.8 are observed. That is, despite

all helper models giving improvements to some degree, simple distributions like the uniform and the unigram are cheaper and should be preferred. Second, when comparing rows 1 to 6 against rows 7 to 12, i.e. applying the uniform label smoothing at the target output side, cumulative improvements are seen. This is an exciting result, because if the improvements from input smoothing and label smoothing do stack, then we can expect even further improvements when applying a better label smoothing recipe, as explained in the previous section. However, looking at rows 13 and 14, we can see that although input smoothing helps on top of only label smoothing, the benefit from applying a stronger label smoothing recipe disappears (row 13 minus row 7 versus row 14 minus row 12). In other words, it is important to apply smoothing to all three sides, i.e. source and target inputs, as well as target output, but the exact label smoothing recipe at the target output is not as crucial.

Table 4.3: Significant Bleu [%] score improvements are obtained from simple and effective smoothing recipes on WMT newstests. We follow the base and big architecture setups from the original transformer paper [Vaswani & Shazeer$^+$ 17] and train the baseline models ourselves. Under the smoothing column, we either use the original setup from Vaswani et al. [Vaswani & Shazeer$^+$ 17], or apply our best recipe to all three sides, i.e. source input, target input, and target output, using input smoothing and label smoothing, with either the uniform or the unigram prior distributions. Note that these results are from different runs compared to those in Table 4.1, but the numbers on the 2014 test set are directly comparable.

| architecture | #parameters | smoothing | WMT newstest | | |
|---|---|---|---|---|---|
| | | | 2014 | 2015 | 2016 |
| transformer base | 65M | [Vaswani & Shazeer$^+$ 17] | 27.4 | 29.4 | 33.8 |
| | | all uniform | 28.5 | 30.2 | 34.7 |
| | | all unigram | 28.5 | 30.6 | 34.8 |
| transformer big | 213M | [Vaswani & Shazeer$^+$ 17] | 28.0 | 30.3 | 33.6 |
| | | all uniform | 29.2 | 31.3 | 34.5 |

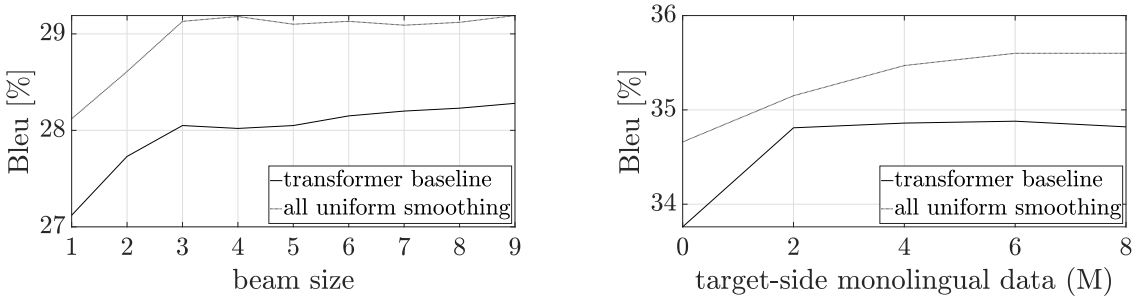
In order to further verify the results on larger datasets, we move on to train systems with different input smoothing and label smoothing setups on WMT2014 English-German, and evaluate on newstest 2014, 2015 and 2016 [Kocmi & Bawden$^+$ 22]. Because of limited computation resources, we only experiment with the transformer baseline and the "all uniform" and the "all unigram" smoothing setups. As seen from Table 4.3, for both the base and big architectures, significant improvements can be achieved when combining input smoothing and label smoothing, but the absolute improvements are smaller compared to those on smaller datasets. Another observation is that the difference between an "all uniform" and an "all unigram" smoothing strategy seems to be limited. Considering that obtaining the unigram distribution still requires counting the training data, while the uniform distribution can be directly applied without any pre-calculation, the "all uniform" recipe is probably sufficient as an "almost free" trick to boost the translation performance in real applications.

Given the improvements shown in Table 4.2 and Table 4.3, it makes sense to ask why there is improvement with input smoothing and label smoothing? We think that the main benefit comes from the more frequent updates of the word vectors. That is, when input smoothing and

label smoothing are applied, in each optimization step, those word vectors for words not seen in the current sentence pair are also updated. Compared to the transformer baseline, these vectors are either not updated at all (at the input side) or only updated implicitly via the denominator in the softmax function (at the output side). Although one may argue that the more frequent updates can be harmful, we argue that because we carefully tune the smoothing hyperparameters, and the smoothing setup is seldom strong enough to overwhelm the actual training signal from the empirical distribution, the overall regularization effect is beneficial for better generalization on unseen test sets.

#### Beam Search and Back-Translation

Finally, to understand the robustness of the input smoothing and label smoothing methods, we consider changing the beam size during beam search, and varying the amount of target-side monolingual data during back-translation.




(a) Beam search on newstest2014.

(b) Back-translation on newstest2016.

Figure 4.10: Robustness of combining input smoothing with label smoothing under different beam search and back-translation settings, on WMT German-English newstest 2014 and 2016. The "transformer baseline" refers to a transformer base model with the default label smoothing setup. The "all uniform smoothing" setup refers to our recipe where all source inputs, target inputs and target outputs are smoothed with uniform priors during training.

In Figure 4.10, we can clearly see the improvements from applying input smoothing and label smoothing. When either the beam size or the monolingual data varies, the improvements remain. One interesting observation is from Figure 4.10b, where the performance boost from back-translation starts to decrease for the transformer baseline when the monolingual data overwhelms the parallel data (around 4M in this case), but remains for the smoothed model. While this could very much be within normal statistical fluctuations, it could also be an indication that the smoothed model training is less susceptible to degradation caused by the training data change.

### 4.6.3 Multi-Agent Training

Having discussed the benefits from applying input smoothing and label smoothing in single-model training, we further shift our attention to the multi-agent case. From the definition, the multi-agent training method introduced in Section 4.5 is orthogonal to the smoothing methods discussed so far. Due to limited computation resources, and also to be comparable with the transformer baseline [Vaswani & Shazeer$^+$ 17], we do not further consider the effect of combining

multi-agent training with the input smoothing and label smoothing results earlier, but instead put our focus on studying the effectiveness of our sentence-level and token-level objectives, as well as the comparison against existing work and the ensembling baselines.

Table 4.4: Effectiveness of multi-agent training. BLEU [%] scores on the respective test sets are reported. $L^{\text{SEN}}$ and $L^{\text{TOK}}$ denote sentence-level and token-level losses respectively. $K$ denotes number of agents. "ens." is short for ensembling that takes the average of the log probabilities of the contributing models. Rows 1 and 2 are results from reference papers for comparison [Vaswani & Shazeer$^+$ 17, Bi & Xiong$^+$ 19]. Except for row 3, checkpoint averaging is not applied in our results because it does not seem to work well with multi-agent training. We only apply label smoothing as in the original transformer paper [Vaswani & Shazeer$^+$ 17], and no input smoothing is applied.

| row id | note | $K$ | $L^{\text{SEN}}$ | $L^{\text{TOK}}$ | ens. | IWSLT'14 | | WMT'16 | WMT'14 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | de-en | nl-en | ro-en | en-de | |
| | | | | | | | | | base | big |
| 1 | Vaswani et al. | 1 | - | - | no | - | - | - | 27.3 | 28.4 |
| 2 | Bi et al. | 4 | - | - | yes | 36.3 | - | - | - | 29.7 |
| 3 | ours | 1 | no | no | no | 34.7 | 37.7 | 34.1 | 27.1 | 28.3 |
| 4 | | 2 | yes | no | | 36.2 | 38.9 | 34.8 | 27.8 | 28.9 |
| 5 | | | | yes | | 36.6 | 39.5 | 35.3 | 28.3 | 29.4 |
| 6 | | 3 | yes | no | | 36.4 | 39.2 | 35.6 | 28.1 | 29.4 |
| 7 | | | | yes | | 36.8 | 39.9 | 35.9 | 28.4 | 29.8 |
| 8 | | 4 | no | no | yes | 36.8 | 38.2 | 35.8 | 28.6 | 29.2 |
| 9 | | | yes | no | no | 36.7 | 39.5 | 35.5 | 28.2 | 29.8 |
| 10 | | | | yes | | 37.0 | 39.8 | 36.1 | 28.7 | 29.9 |
| 11 | | | | | yes | 38.0 | 40.3 | 36.4 | 28.8 | 30.1 |

In Table 4.4, we vary the number of agents $K$, the training objectives, and the post-training ensembling setup. Considering $K$, we expect that with increasing number of agents, the system should perform more strongly. Results from rows 3, 4, 6 and 9 or from rows 3, 5, 7 and 10 confirm our expectation, but the improvements seem to diminish as more and more agents are introduced. Considering $L^{\text{SEN}}$ and $L^{\text{TOK}}$, we expect that both the sentence-level and the token-level objectives should give improvements on top of a non-mutual-learning baseline. This can be verified when comparing rows 3 to 7 and rows 9 and 10, i.e. $L^{\text{SEN}}$ brings a large improvement initially, and $L^{\text{TOK}}$ further boosts the performance by a little. Considering the ensembling trick, we expect that agents trained interactively should yield an overall ensemble. This statement can be checked when looking at rows 8 and 11 together, and we can see that agents trained separately under-perform agents trained jointly when ensembling is applied. Lastly, if we compare our best setup in row 11 against the previous work in row 2 [Bi & Xiong$^+$ 19] (recall from Section 4.5 that the main difference is that they distill the knowledge in a one-to-many manner, while in our case training signals flow in all agent pairs), we can observe that further improvements can be obtained, which are stronger on smaller datasets and relatively weaker on larger datasets. From these results, we summarize that, overall, multi-agent training is a costly but effective method to boost the translation performance.

## 4.7 Summary

In this chapter, we focused on the topic of smoothing in neural machine translation. Smoothing here is a general term referring to the regularization of the model. We started with label smoothing and confidence penalty, and gave analytical solutions to when the model optimums are obtained. We then moved on to input smoothing, where we perform weighted summation of word vectors instead of simple word embedding matrix lookups. In both label smoothing and input smoothing, we considered extensions such as where to smooth, how exactly to smooth, and with which prior distributions to smooth. Shifting our attention to multi-agent training, we proposed extensions to further perform token-level inter-agent learning after initial convergence of sentence-level co-learned agents. Experimentally, we found better-performing training recipes for label smoothing, confirmed the cumulative improvements from label smoothing and input smoothing, and showed that simple prior distributions such as the uniform or the unigram distribution are sufficient to achieve good translation performance. Our multi-agent training experiments gave significant improvements in automatic evaluation metrics against strong baselines, pushing forward the state-of-the-art methods to combine multiple models to achieve stronger prediction.

# 5. Language Modeling For Translation

The machine translation task has a sequence-to-sequence nature: given a source sequence, generate a target sequence accordingly. For this reason, the neural-network-based modeling of machine translation is primarily done with encoder-decoder models [Kalchbrenner & Blunsom 13, Cho & van Merriënboer$^+$ 14b, Sutskever & Vinyals$^+$ 14, Bahdanau & Cho$^+$ 15, Vaswani & Shazeer$^+$ 17]. Specifically, a dedicated encoder processes the entire source sequence by summarizing information from different source word positions, and generates some hidden representations for it. Then, a decoder goes through the target word positions, starting from an artificial sequence start symbol, and generates the next target word by considering what has been translated so far, as well as the relevant source hidden representations. Previous works differ mainly in how the encoding (e.g. using feedforward, uni- or bi-directional long short-term memory, convolutional or self-attention modules) and retrieval of source information from the hidden representations (attention mechanism) are done.

Recently, in light of the rapid development of large language models [Devlin & Chang$^+$ 19, Shoeybi & Patwary$^+$ 19, Brown & Mann$^+$ 20, Raffel & Shazeer$^+$ 22, BigScience Workshop 22, Zhang & Roller$^+$ 22, Schulman & Zoph$^+$ 22], the long-standing encoder-decoder approach for neural machine translation is now also in question. As mentioned in Section 1.3.1, the encoder-decoder model of translation performs discriminative modeling, and no reverse translation model or log-linear combination like in classic statistical machine translation are involved. However, one can also consider translation as the task of modeling the joint probability between the source and the target sentence, and perform generative modeling instead. In this way, the search process remains largely intact, because when the source sentence is given, the search for the target hypothesis does not depend on the probability of the source sentence. The benefit of such an approach mainly lies in its simplicity - one does not have to explicitly separate the encoder and decoder, and a monolithic model trained on concatenations of source and target sentences, as if it was a language modeling task, is therefore possible. Apart from the angle of discriminative modeling versus generative modeling, one can also think of this issue from the perspective of contextualized language modeling. That is, translation is simply language modeling in the target language with additional context from the source sentence. Because the entire source sentence is available at test time, sometimes people also refer to such contextual information as "prefix" [Liu* & Saleh*$^+$ 18b, Dong & Yang$^+$ 19, Raffel & Shazeer$^+$ 22, Scao & Wang$^+$ 22]. With this additional context, one needs to be careful with the internal self-attention mechanism as well as the auxiliary loss alongside the main language modeling objective at the target side, and we will delve into details of these topics in this chapter.

## 5.1 Related Work

Defining the translation task as a sequence-to-sequence task, the encoder-decoder architecture is a natural choice of neural network typology and has been the go-to-method since the early development of neural-network-based translation systems. For example, in the work by Kalchbrenner and Blunsom [Kalchbrenner & Blunsom 13], they proposed a convolutional sentence model to encode source information and a separate recurrent language model to decode target information, which is conditioned on the former. Although the terms "encoder" and "decoder" were not used in that paper, their model was effectively of an encoder-decoder nature. Since their work, the neural approach to translation has seen rapid developments. In the early work of sequence-to-sequence modeling for translation [Sutskever & Vinyals$^+$ 14, Cho & van Merriënboer$^+$ 14b, Cho & van Merriënboer$^+$ 14a], the terms "encoder" and "decoder" were used to denote the module that embeds source information into some continuous-valued hidden space and the module that decodes from it together with the target-side context, respectively. Despite further extensions to translation models, such as the attention mechanism [Bahdanau & Cho$^+$ 15], multi-task learning [Luong & Pham$^+$ 15], application of convolutional networks [Gehring & Auli$^+$ 17], and the later widely-used self-attention [Vaswani & Shazeer$^+$ 17], the separation of the encoder and decoder has remained the norm when it comes to translation modeling. Since the transformer architecture [Vaswani & Shazeer$^+$ 17] became popular, many works that aim to improve a specific component of the network maintained the overall encoder-decoder structure, and focused on the part that was interesting for them. Some examples include the improvement of the positional encoding [Shaw & Uszkoreit$^+$ 18], multi-head attention [Voita & Talbot$^+$ 19] and attention/alignment [Alkhouli & Bretschner$^+$ 18]. In works that go beyond certain model components but focus on the modeling itself, e.g., doing round-trip translation [Tu & Liu$^+$ 17] and performing non-autoregressive modeling [Gu & Bradbury$^+$ 18], the encoder-decoder architecture is also often kept. Notice that there have been other works that drifted away from the encoder-decoder regime and neglecting them in this discussion would not do them justice. This includes works such as the development of direct neural hidden Markov models for translation [Wang & Alkhouli$^+$ 17, Wang & Zhu$^+$ 18, Wang & Yang$^+$ 21a], completely dropping the attention and separating the encoding and decoding steps [Press & Smith 18], and simplifying the neural architecture by removing the entire encoder stacks [Tang & Sennrich$^+$ 19].

Although language models were traditionally used for statistical machine translation [Och & Tillmann$^+$ 99, Schwenk & Koehn 08], their application in the neural-network era is rather limited. They are commonly seen in fusion techniques [Stahlberg & Cross$^+$ 18] and back-translation [Sennrich & Haddow$^+$ 16a, Graça & Kim$^+$ 19]. That said, their development followed a similar route as in many other fields that widely adopted neural networks for modeling, i.e. starting from simple feedforward language models [Bengio & Ducharme$^+$ 03], to the application of recurrent neural networks such as the long short-term memory network [Sundermeyer & Schlüter$^+$ 12], to exploring the use of convolutional networks [Dauphin & Fan$^+$ 17], and to the widely-successful self-attentive transformer language models [Irie & Zeyer$^+$ 19]. In the course of this, the language models became stronger compared to the historical count-based models [Kneser & Ney 95], and their modeling capacities became more and more impressive. In the literature, examples that support this claim include works that push the limit of large-scale modeling [Radford & Wu$^+$ 19b, Brown & Mann$^+$ 20], aim to model context dependencies that are very long [Dai & Yang$^+$ 19], and the inspiring paradigm shift from the supervised training to the pre-training-and-finetuning scheme [Devlin & Chang$^+$ 19]. In light of the fast development of large-scale language models, and considering the fact that translation is indeed a language-modeling-style task with additional source-side information, there exists another line

of work that starts to shake the reign of encoder-decoder architectures for translation.

Dated to the year 2012, Mikolov and Zweig [Mikolov & Zweig 12] already mentioned the potential extension of contextualized language modeling to the translation task by treating the source sentence as additional context. In 2018, He et al. [He & Tan$^+$ 18] described a model where the parameters in the encoder and decoder stacks are shared, and the input to their model was the concatenation of source and target sentence. In their work, no explicit loss is used to reconstruct the source sentence at the output side. In the Ph.D. dissertation by Kazuki Irie [Irie 20], he described a similar approach, but observed a small degradation in the translation performance when using a causal mask in the attention mechanism. In the work by Raffel et al. [Raffel & Shazeer$^+$ 20], the authors looked into the details of attention masking, and discussed the implications of three different kinds of masks, which they call "fully-visible", "causal" and "causal with prefix". The aforementioned work paved the way for Wang et al. [Wang & Tu$^+$ 21], where they further introduced a two-step learning rate decay function on the source reconstruction loss, and showed competitive performance against strong encoder-decoder transformer-based translation baselines. More recently, several works further expanded on the concept. Specifically, Zhang et al. [Zhang & Ghorbani$^+$ 22] examined the scaling and cross-lingual behavior of such language models for translation. Hawthorne et al. [Hawthorne & Jaegle$^+$ 22] explored the capabilities of language models to model long-range dependencies on tasks in various modalities. Similarly, Hao et al. [Hao & Song$^+$ 22] developed a language model which serves as a general interface for language, vision and multilingual tasks. The most recent addition to this line of work as of the time of writing is by Fu et al. [Fu & Lam$^+$ 23], where the authors compared decoder-only and encoder-decoder models on sequence-to-sequence tasks, and argued that as the generation becomes longer, less and less attention is concentrated on the source side. Finally, in the context of large language models [Radford & Wu$^+$ 19b, Brown & Mann$^+$ 20, Ouyang & Wu$^+$ 22, Schulman & Zoph$^+$ 22], the emergent abilities [Wei & Tay$^+$ 22], zero-shot and few-shot translation capabilities [Moslem & Haque$^+$ 23], and even evaluation capabilities [Kocmi & Federmann 23] have been discussed[1].

## 5.2 Notations

To clarify the notations, in this chapter:

- $f_1^J$ denotes a source sentence, with $1, 2, ..., j, ..., J$ denoting the positions for each word.
- $e_1^I$ denotes a target sentence, with $1, 2, ..., i, ..., I$ denoting the positions for each word.
- $\tilde{f_j}$ is the word vector of the word $f_j$.
- $\tilde{e}_i$ is the word vector of the word $e_i$.
- In our approach of concatenating the source and the target sentences, we use $k$ as a running index in the concatenated sequence, i.e. $k \in \mathbb{Z}, 1 \leq k \leq J + I$.

[1] Depending on the languages and resources that are used to train such large language models, the multilingual capabilities of such models may be limited, in the sense that performance on translation directions involving English is better than without, because English text usually makes up the majority of the training corpus. Later in Section 5.4.6, we also make an argument that the emergent translation capabilities of large language models may be rooted from the inclusion of "pseudo parallel" sentences, despite some works in the literature claiming that the translation evaluation is done completely in a "zero-shot" manner.

- $(l)$ in the superscript of a quantity denotes the layer in the encoder or decoder stack. For instance, $h_j^{(l)}$ refers to the hidden representation output of source word $f_j$ in encoder layer $l$. $(l = 0)$ in the superscript therefore denotes the input to the encoder or decoder stack, and $(l = L)$ in the superscript denotes the outputs of the encoder or decoder stack.
- $h_j$ denotes the hidden representation for word $f_j$.
- $s_i$ denotes the hidden representation for word $e_i$. Notice that in the traditional encoder-decoder attention approach, $s_i$ is the "query" used to retrieve information from the source hidden representations $h_1, h_2, ..., h_j, ..., h_J$.
- $A[\cdot, \cdot]$ denotes the attention scores or energies between two positions. In the transformer approach, the scaled dot product attention is used for the actual calculation.
- $\alpha(j'|j)$ denotes the source-side self-attention weights in the transformer approach, i.e. a distribution over source positions given a certain position $j$.
- $\alpha(i'|i-1)$ denotes the target-side self-attention weights in the transformer approach, i.e. a distribution over target positions (more specifically, those positions that are prior to and including the last position $i-1$, because the attention over the future positions is by definition masked to be zero to avoid cheating) given the last position $i-1$.
- $\alpha(j|i-1)$ denotes the cross-attention weights in the transformer approach, i.e. a distribution over source positions given the last target position $i-1$.
- $c_{i-1}$ denotes the context vector summarizing all related source and target information in order to output the next word $e_i$. If it is in the first decoder layer, $c_{i-1}^{(l=0)}$ refers to the target word embedding $\tilde{e}_{i-1}$. Similarly, we use $c_{i-1}^{(l=L)}$ to denote it being in the last decoder layer.

For simplicity, we drop further notations on heads, layer-normalization, skip-connections, etc., the details of which can be found in the original transformer paper [Vaswani & Shazeer$^+$ 17].

With fast development of large language models as well as new models being reported and released frequently, there may exist some confusion regarding the naming of certain components. In this section, we wish to clarify some of the terminology often seen in the literature and position this work more accurately among existing and emerging works:

- Autoregressive versus non-autoregressive. For autoregressive models (e.g. in Irie et al. [Irie & Zeyer$^+$ 19]), the output words are shifted, and the model needs to predict the next word given the previous words. For non-autoregressive models (e.g. in Devlin et al. [Devlin & Chang$^+$ 19]), the output words are not shifted (i.e. are synchronous with the input words), and one needs to predict the masked word given surrounding words. The reason why non-autoregressive models are sometimes referred to as masked language models is that, if input words are not masked, the word classification becomes a trivial copying task. For the same reason, such tasks are also sometimes called "cloze" tasks in the literature.
- Causal versus non-causal. Both terms often refer to masks used in the attention mechanism. A causal mask means that for a certain position, it is allowed to "look at" (calculate attention energy against) all previous positions and itself but strictly not future positions.

It is often used in conjunction with autoregressive models to avoid the model cheating the next word prediction task by attending to future words. In contrast, a non-causal mask is one where all positions are allowed to "look at" all positions, and this is often used in either non-autoregressive models or for prefix positions [Wang & Roberts$^+$ 22].

- Encoder-decoder versus encoder-only versus decoder-only. Encoder-decoder models refer to those models where a separate encoder is used to encode information from the source sequence into some hidden representation and a separate decoder is used to decode information from previous target context as well as from the encoder hidden representation (e.g. in Vaswani et al. [Vaswani & Shazeer$^+$ 17]). On the other hand, encoder-only models are those models which drop the decoder and only rely on an encoder to aggregate information from context and make predictions with final word projection and softmax layers. In the literature, sometimes people refer to these encoder-only models as decoder-only models, to highlight that the final classification is also done with the model. However, this is not accurate, because traditionally, a decoder is different from the encoder, e.g. in transformer networks, the decoder has self-attention layers followed by cross-attention layers, but the encoder only contains self-attention layers, and in actual implementations, what is kept is the encoder. For this reason, in this work, we refer to such monolithic models as encoder-only models.

- Zero-shot versus few-shot versus finetuning. For zero-shot and few-shot, no further training is done. Zero-shot means no additional example is supplied to the model during evaluation. Few-shot means only presenting a handful of examples before querying the model on the downstream task. Commonly, the underlying bulk of model parameters are not updated, and the examples are only presented to the model as additional context. Finetuning refers to the case where the model parameters are further trained on task-specific data before the downstream evaluation.

- Prefix versus prompting. Prefix refers to the additional information presented to the model when querying its outputs. Although the generation of the response, or the continuation of the sequence, is typically autoregressive, the model may be allowed to process the entire prefix in a non-autoregressive manner, which correspondingly requires a non-causal attention mask on the prefix positions [Wang & Roberts$^+$ 22, Diao & Zhou$^+$ 23]. Prompting can be thought of as an extension of the prefix concept. When performing zero-shot, few-shot or supervised (finetuning the pre-trained language model on supervised data) tasks, it is possible to frame or phrase the task description as a prompt. It was shown that different choices of prompt have an impact on the final performance [Radford & Narasimhan 18, Radford & Wu$^+$ 19a, Sanh & Webson$^+$ 22].

Having clarified some terminology commonly seen in the literature, it is possible to more accurately position our work here. With the motivation to develop a more monolithic architecture for machine translation, we aim to develop an encoder-only model, which is trained on the concatenations of source and target sentences, with non-causal masking on the source positions and causal masking on the target positions, and perform translation tasks without further finetuning after the generic language-modeling-style pre-training. For this reason, we will refer to such models as translation language models (TLMs) from here on.

## 5.3 Methodology

### 5.3.1 Dependency Differences

Because we build the translation language models with the transformer [Vaswani & Shazeer$^+$ 17] encoder-decoder model as a baseline, it makes sense to compare the dependency differences between the two from a theoretical point of view. Below, for simplicity, we drop the notations for feedforward layers and residual connections in transformers [Vaswani & Shazeer$^+$ 17] and focus on attention-related computations.

The internal dependencies in the transformer baseline model can be expressed as follows:

$$
\begin{aligned}
&\text{source self-attention:}\\
&\qquad \alpha^{(l+1)}(j'|j) = \mathrm{softmax}\big(A[h_j^{(l)}, h_{j'}^{(l)}]\big)\\
&\qquad h_j^{(l+1)} = \sum_{j'=1}^{J} \alpha^{(l+1)}(j'|j) \cdot h_{j'}^{(l)}\\
&\text{target self-attention:}\\
&\qquad \alpha^{(l+1)}(i'|i-1) = \mathrm{softmax}\big(A[c_{i-1}^{(l)}, c_{i'}^{(l)}]\big)\\
&\qquad s_{i-1}^{(l+1)} = \sum_{i'=1}^{i-1} \alpha^{(l+1)}(i'|i-1) \cdot c_{i'}^{(l)}\\
&\text{cross-attention:}\\
&\qquad \alpha^{(l+1)}(j|i-1) = \mathrm{softmax}\big(A[s_{i-1}^{(l+1)}, h_j^{(l=L)}]\big)\\
&\qquad c_{i-1}^{(l+1)} = \sum_{j=1}^{J} \alpha^{(l+1)}(j|i-1) \cdot h_j^{(l=L)}
\end{aligned}
\tag{5.1}
$$

The internal dependencies in our encoder-only translation language model can be expressed as:

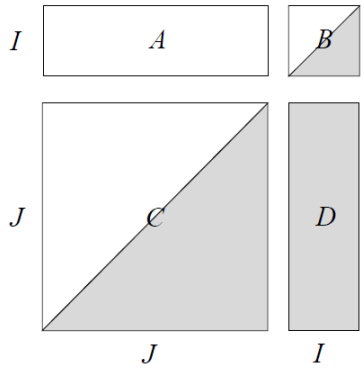
$$
\begin{aligned}
&\text{source self-attention:}\\
&\qquad \alpha^{(l+1)}(j'|j) = \mathrm{softmax}\big(A[h_j^{(l)}, h_{j'}^{(l)}]\big)\\
&\qquad h_j^{(l+1)} = \sum_{j'=1}^{J} \alpha^{(l+1)}(j'|j) \cdot c_{j'}^{(l)}\\
&\text{target self-attention:}\\
&\qquad \alpha^{(l+1)}(k|i-1) = \mathrm{softmax}\big(A[c_{J+i-1}^{(l)}, c_k^{(l)}]\big)\\
&\qquad s_{i-1}^{(l+1)} = \sum_{k=1}^{J+i-1} \alpha^{(l+1)}(k|i-1) \cdot c_k^{(l)}\\
&\text{context vector concatenation:}\\
&\qquad c_k^{(l+1)} = \begin{cases} h_k^{(l+1)}, & \text{if} \quad 1 \le k \le J \\ s_{k-J}^{(l+1)}, & \text{if} \quad J+1 \le k \le J+I \end{cases}
\end{aligned}
\tag{5.2}
$$

Comparing the two, it is clear that the source-side self-attention is preserved. What is different is in the target-side attention calculations. In the transformer case, a two-step calculation

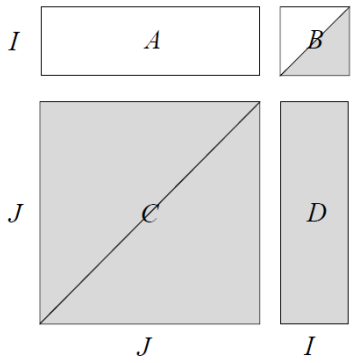
happens, where the target context is first summarized via target self-attention, and then used to query the source side context via cross-attention. In our approach, the previous two-step calculation is flattened into one step, i.e. a flat attention calculation over all source and target positions ($k$ running from 1 to $J + I$, with future target positions $k \geq i$ being masked out) and the dedicated cross-attention is therefore dropped. Correspondingly, the construction of the context vector is simplified, into a simple concatenation of the source hidden states $h$ and target hidden states $s$.

### 5.3.2 On the Attention Masking

As shown in Equation 5.1, the attention mechanism in the transformer is applied in three unique calculations: the source self-attention, the target self-attention, and the cross-attention. These correspond to three attention matrices, with the dimensions $J \times J$, $I \times I$, and $J \times I$. In Fig. 5.1, the three matrices are C, B, and D, respectively.




(a) source-side triangular (causal) mask




(b) source-side full (non-causal) mask

Figure 5.1: Attention masks in translation language models. We query along the horizontal direction against keys in the vertical direction. Grey areas indicate valid attention and white areas indicate invalid attention.

The meaning of each attention matrix can be explained as follows:

- A means attention from source positions to target positions. We whiten this area out because we do not want the model to cheat. That said, when decoding a certain target position $i$, conceptually, there is no problem for the source positions to attend to previous target positions up to $i - 1$. In other words, a target-position-dependent masking mechanism in $(J + I) \times (J + I) \times I$ might be possible, which may serve as an implicit fertility model.
- B means attention from target positions to target positions. We allow the model to look at current and past positions, but not future positions, to avoid cheating.
- C means attention from source positions to source positions. This is where the two subfigures vary. Either the model is restricted to look at future positions in the spirit of left-to-right language modeling (causal), or we say the source is just a prefix and the model should be allowed to attend to whichever source positions it wants (non-causal).

As discussed in Section 5.2, the distinction between causal and non-causal masking is an important point of how previous works differ from one another [Irie 20, He & Tan$^+$ 18, Raffel & Shazeer$^+$ 20, Wang & Tu$^+$ 21, Zhang & Ghorbani$^+$ 22].

- D means attention from target positions to source positions. This should of course be valid because we want to allow the model to look at any source words when deciding the next target word.

### 5.3.3 On the Source Reconstruction Loss

When training the translation language models, the training loss can be divided into two parts, namely a target-side machine translation loss ($L_{\text{MT}}$) and a source-side reconstruction loss ($L_{\text{RE}}$):

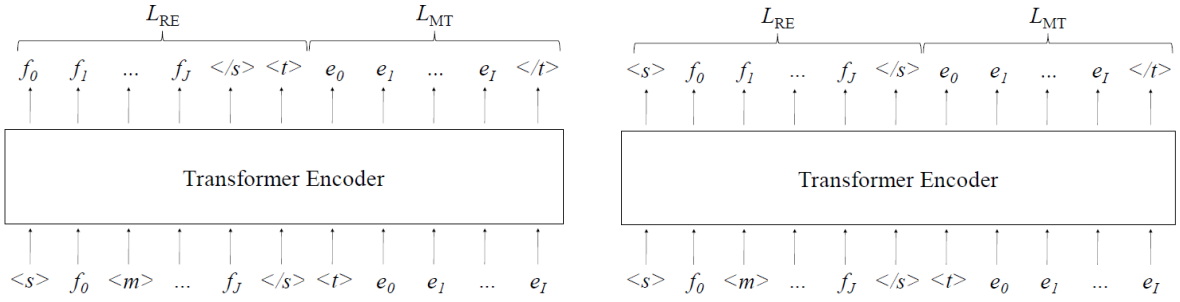
$$\begin{aligned} L_{\text{TLM}} &= \lambda L_{\text{RE}} + L_{\text{MT}} \\ &= -\frac{1}{N}\sum_{n=1}^{N}\left(\lambda\frac{1}{J_n}\sum_{j=1}^{J_n}\log P_\theta\left([f_n]_j \mid [f_n]_0^{j-1}\right) + \frac{1}{I_n}\sum_{i=1}^{I_n}\log P_\theta\left([e_n]_i \mid [e_n]_0^{i-1}, [f_n]_1^{J_n}\right)\right) \end{aligned} \tag{5.3}$$

In Equation 5.3, the combined TLM loss has a hyperparmeter $\lambda$, which controls the strength of the source reconstruction loss. As seen, the translation loss is simply the cross-entropy on the target words given the source words and the previous target words, and the reconstruction loss is a language modeling objective at the source side (for example, in Wang et al. [Wang & Tu$^+$ 21], they formulate the source reconstruction loss in such an autoregressive language modeling manner). This is, of course, just an example - in reality, one does not have to decompose the source sentence in an autoregressive manner, or restrict oneself to only attending to the past, as mentioned in Section 5.3.2. Notice that the source sentence probability does not have an effect during the search process:

$$\begin{aligned} \underset{e_1^I, I}{\operatorname{argmax}} P_\theta(f_1^J, e_1^I) &= \underset{e_1^I, I}{\operatorname{argmax}} \left\{ P_\theta(f_1^J) \cdot P_\theta(e_1^I | f_1^J) \right\} \\ &= \underset{e_1^I, I}{\operatorname{argmax}} P_\theta(e_1^I | f_1^J) \end{aligned} \tag{5.4}$$

For this reason, the source-side reconstruction loss used during the training stage only serves as a regularization loss or a second objective in a multi-task training setup. In other words, one may make design choices such as to shift (Figure 5.2a) or not to shift (Figure 5.2b) the source output for it to be a language modeling or autoencoding objective. When doing so, the attention masks need to be set accordingly, i.e. a triangular mask for language modeling and a full mask for autoencoding, as mentioned in Section 5.3.2. In addition, it is possible to further add BERT-style noise [Devlin & Chang$^+$ 19] to the source inputs. Specifically, we randomly pick 15% source positions, and replace the token at these positions with a mask token, a random token, or the original token with 80%, 10%, and 10% chances[2]. In the original BERT approach, there is no loss associated with the unmasked positions. Here, we nonetheless include them into $L_{\text{RE}}$. Our reasoning is simple: if these unmasked positions are included in the context, having them in the loss corresponds to a simple copying task and should at least not hurt the model, and if these unmasked positions are not included in the context, the loss becomes the usual cross-entropy training objective.

[2] Although there exists work that argues for other hyperparameter choices [Wettig & Gao$^+$ 23] or applies softer noises [Gao & Zhu$^+$ 19, Gao & Liao$^+$ 20], we do not further explore other setups for adding artificial noise here.




(a) source-side shift in TLM

(b) no source-side shift in TLM

Figure 5.2: Source-side reconstruction in translation language models (TLMs). Artificial start and end of sentence tokens, as well as BERT-style mask tokens [Devlin & Chang$^+$ 19] are included in the figure. When source output is shifted by one position and the source-side attention mask is triangular, it corresponds to a language modeling objective. When source output is not shifted and the source-side attention mask is full, it corresponds to an autoencoding objective.

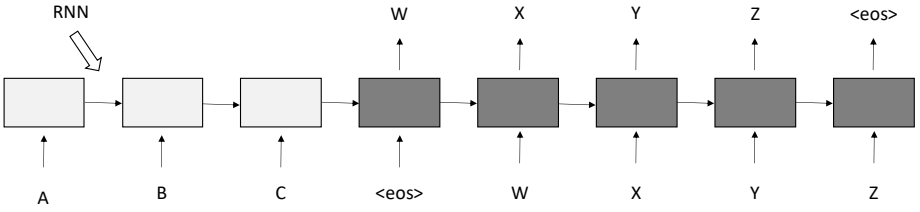
Finally, one more thing to consider is the learning rate schedule for the source-side reconstruction loss, i.e. $\lambda$ in Equation 5.3. In the work by Wang et al. [Wang & Tu$^+$ 21], they used a two-step linear decaying schedule to anneal the source-side loss, where the initial decay is fast, from 1.0 down to 0.1 at gradient update step $\tau$, and the later decay is slower. In this work, denoting the gradient update step as $t$ and with $t_0$ as the time step when $\lambda$ decays down to zero, we consider four variants of $\lambda_t$ to anneal the source reconstruction loss:

- $\lambda_t = 0$
- $\lambda_t = 1$
- $\lambda_t = \begin{cases} -\frac{0.9}{\tau}t + 1, & \text{if} \quad t \leq \tau \\ -\frac{0.1}{t_0 - \tau}t + \frac{0.1}{t_0 - \tau}t_0, & \text{if} \quad t > \tau \end{cases}$
- $\lambda_t = \exp(-\ln 0.1 \frac{t}{\tau})$

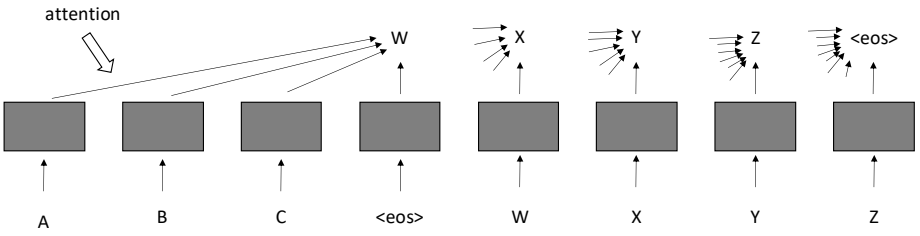
Wang et al. [Wang & Tu$^+$ 21] used the third variant, and our $\tau$ corresponds to $\beta$ in their paper.

### 5.3.4 Comparison to Traditional Sequence-to-Sequence Models

To avoid confusion with the traditional sequence-to-sequence models, in Figure 5.3, we illustrate and highlight the differences. Figure 5.3a is adapted from Figure 1 in Sutskever et al. [Sutskever & Vinyals$^+$ 14] and redrawn by us - it depicts the traditional sequence-to-sequence model. Notice how separate encoder and decoder are used (light grey and dark grey), and that recurrent neural network (RNN) modules are used to process the source and target inputs separately. In Figure 5.3b, our translation language model is depicted. Notice how a shared self-attentive module (dark grey) is used to process the entire source-target concatenation and that each target position has the possibility to attend to all previous positions via the attention mechanism. Comparing the two, we highlight that not only is our model more compact (no need to separate the encoder and the decoder), but also contains more modeling capacity, as the information flow from source to target is not bottlenecked by passing hidden states from one recurrent network to another.




(a) traditional sequence-to-sequence model (adapted and redrawn from Figure 1 in Sutskever et al. [Sutskever & Vinyals$^+$ 14])




(b) our translation language model; notice that there is no separation between encoder and decoder, and self-attentive modules are used

Figure 5.3: Comparison of our TLM to traditional sequence-to-sequence model. (a) a separate recurrent encoder (light grey) and a separate recurrent decoder (dark grey) are used to process the source and target inputs respectively [Sutskever & Vinyals$^+$ 14]. (b) a shared self-attentive model (dark grey) is used to process the concatenated source and target sequence, and all previous positions contribute to the prediction of the current output (some arrows denoting attention dependency are shortened for aesthetic reasons).

## 5.4 Experimental Results

### 5.4.1 Finding the Best Setup

As discussed in Section 5.3, we have to tune four hyperparameters to find a good setup for a monolithic translation language model: 1. source language modeling versus source autoencoding (Figure 5.2); 2. triangular mask versus full mask (Figure 5.1); 3. no artificial source noise versus BERT-style source noise (Section 5.3.3); 4. learning rate schedule of $\lambda_t$ (Section 5.3.3). We perform a grid search over all combinations of the four hyperparameters (see Table 5.1). The two datasets used here are IWSLT2014 German-English (de-en) and WMT2016 English-Romanian (en-ro), both of which are relatively small. The purpose of this grid search experiment is to quickly iterate over different setups, find a reasonable one, and draw preliminary conclusions about training translation language models (TLMs). Because Table 5.1 is rather big, in the following, we make interpretations by visiting the four hyperparameters one-by-one. For each hyperparameter we consider, we manually look up in the table for the best-performing combinations of the other three hyperparameters in terms of Bleu score - to effectively answer the question: "if we do the best we can elsewhere, what effects does this

Table 5.1: Grid search of four source-reconstruction-related hyperparameters on IWSLT2014 German-English (de-en) and WMT2016 English-Romanian (en-ro) translation tasks. LM means to shift the source-side outputs and train with an autoregressive language modeling objective, and AE means not to shift and corresponds to an autoencoding objective. The triangular and full masks are illustrated in Figure 5.1. The details of the noise setup and the source reconstruction loss scheduling can be found in Section 5.3.3. Our interpretations of the table are given in Section 5.4.1.

| architecture | source reconstruction variant | | | | de-en | | en-ro | |
|---|---|---|---|---|---|---|---|---|
| | task | mask | noise | sched. | BLEU[%] | TER[%] | BLEU[%] | TER[%] |
| MT (enc-dec) | - | - | - | - | 34.9 | 44.5 | 26.0 | 54.8 |
| LM (enc-only) | LM | tri. | none | 0 | 33.5 | 46.0 | 25.4 | 55.5 |
| | | | | 1 | 34.4 | 45.3 | 25.2 | 55.7 |
| | | | | lin | 34.2 | 45.2 | 25.5 | 55.4 |
| | | | | exp | 34.6 | 45.2 | 25.3 | 55.7 |
| | | | BERT | 0 | 33.6 | 45.7 | 25.4 | 55.6 |
| | | | | 1 | 34.4 | 45.1 | 25.2 | 55.8 |
| | | | | lin | 34.4 | 45.4 | 25.6 | 55.5 |
| | | | | exp | 34.2 | 45.8 | 25.4 | 55.3 |
| | | full | none | 0 | 34.5 | 44.9 | 25.8 | 55.4 |
| | | | | 1 | 34.5 | 44.8 | 25.9 | 55.0 |
| | | | | lin | 34.5 | 44.9 | 25.7 | 55.3 |
| | | | | exp | 34.4 | 44.8 | 26.1 | 54.8 |
| | | | BERT | 0 | 34.5 | 45.1 | 26.2 | 54.8 |
| | | | | 1 | 34.4 | 44.9 | 25.6 | 55.3 |
| | | | | lin | 34.7 | 44.5 | 25.8 | 55.3 |
| | | | | exp | 34.6 | 44.9 | 25.9 | 54.9 |
| | AE | tri. | none | 0 | 32.2 | 47.2 | 25.3 | 55.6 |
| | | | | 1 | 32.5 | 46.2 | 24.9 | 55.9 |
| | | | | lin | 32.0 | 46.3 | 25.2 | 55.8 |
| | | | | exp | 32.0 | 46.8 | 25.3 | 55.3 |
| | | | BERT | 0 | 30.8 | 47.9 | 25.1 | 56.1 |
| | | | | 1 | 33.5 | 45.9 | 25.1 | 56.0 |
| | | | | lin | 31.5 | 47.5 | 25.2 | 55.7 |
| | | | | exp | 33.6 | 45.9 | 25.5 | 55.6 |
| | | full | none | 0 | 34.4 | 45.1 | 25.8 | 55.2 |
| | | | | 1 | 34.0 | 45.3 | 25.9 | 55.1 |
| | | | | lin | 33.8 | 45.7 | 25.7 | 55.3 |
| | | | | exp | 34.0 | 45.5 | 25.7 | 55.3 |
| | | | BERT | 0 | 34.9 | 45.0 | 25.8 | 55.3 |
| | | | | 1 | 35.0 | 45.0 | 26.0 | 55.1 |
| | | | | lin | 34.7 | 45.0 | 25.7 | 55.4 |
| | | | | exp | 34.8 | 44.8 | 25.8 | 55.4 |

hyperparameter have?”

### Source Language Modeling Versus Source Autoencoding

Comparing the source language modeling (autoregressive) objective and the source autoencoding objective, there does not seem to be a significant difference in terms of BLEU scores. Specifically, on both IWSLT2014 German-English and WMT2016 English-Romanian datasets, the fluctuations of BLEU scores are within ±0.2% absolute BLEU scores. The fluctuations in TER appear to be bigger, and this might be because we chose "the best set of other hyperparameters" by selecting the setting with the highest BLEU scores. In fact, if we quickly revisit Table 5.1, overall, the choice of the source auxiliary task does not seem to play an important role in the final translation performance.

Table 5.2: Comparison between source language modeling and source autoencoding. The numbers are read off Table 5.1 by fixing the auxiliary task and picking the best set of other parameters in terms of BLEU score. Results suggest a weak correlation between the choice of the source training criterion and the final BLEU score.

| architecture | source training crit. | de-en | | en-ro | |
|---|---|---|---|---|---|
| | | BLEU[%] | TER[%] | BLEU[%] | TER[%] |
| transformer MT (enc-dec) | - | 34.9 | 44.5 | 26.0 | 54.8 |
| transformer LM (enc-only) | language modeling | 34.7 | 44.5 | 26.2 | 54.8 |
| | autoencoding | 35.0 | 45.0 | 26.0 | 55.1 |

### Triangular Versus Full Attention Masks

Comparing the triangular attention mask versus the full attention mask (Figure 5.1), the full attention mask clearly has an advantage. Intuitively, we should not limit ourselves and restrict the attention from early source words to later source words, because the entire source sentence is available at test time. If the future context proves not to be helpful, the model has the ability to fall back to a triangular mask. In other words, the triangular attention mask at the source side is a special case of using a full attention mask. As seen in Table 5.3, the translation performance is better when using a full attention mask compared to using a triangular attention mask. This suggests that the source hidden representations generated by allowing the model to attend to future source positions are beneficial.

Table 5.3: Comparison between triangular and full attention masks. The numbers are read off Table 5.1 by fixing the attention mask and picking the best set of other parameters in terms of BLEU score. Results suggest that a full attention mask is helpful in improving the translation performance of TLMs.

| architecture | source mask | de-en | | en-ro | |
|---|---|---|---|---|---|
| | | BLEU[%] | TER[%] | BLEU[%] | TER[%] |
| transformer MT (enc-dec) | - | 34.9 | 44.5 | 26.0 | 54.8 |
| transformer LM (enc-only) | triangular | 34.6 | 45.2 | 25.6 | 55.5 |
| | full | 35.0 | 45.0 | 26.2 | 54.8 |

To further highlight the importance of the full (non-causal) source-side mask, we scatter the BLEU scores of systems with triangular or full source mask under different experiment settings. As seen in Figure 5.4, under most experiment settings, the full source mask variant (non-causal)

outperforms the triangular source mask variant (causal). The results presented in Table 5.3 (corresponding to experiment setting #4 for triangular mask and #14 for full mask here) are essentially cheating experiments (in favor of the triangular mask variant) because we manually select the best setup by maximizing the Bleu score.

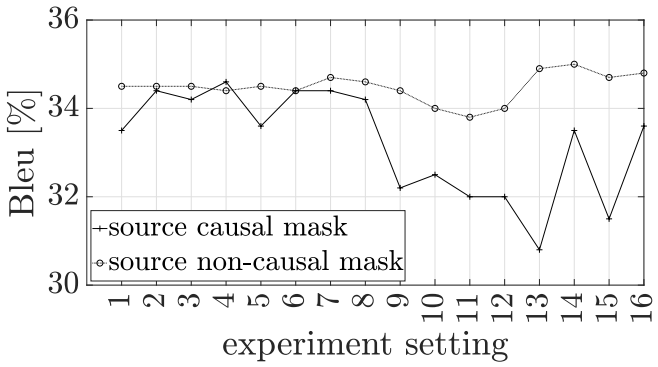



Figure 5.4: Scattering of Bleu [%] scores of systems with triangular (causal) or full (non-causal) source mask under different experiment settings, on IWSLT2014 German-English. Data points are from Table 5.1. Under the same experiment setting index, all hyperparameters are the same except for the source mask.

### Effect of BERT-Style Noise

Past experience tells us that augmenting the training data by adding artificial noise is helpful to improve the generalization capabilities of the model [Hill & Cho$^+$ 16, Kim & Geng$^+$ 18, Gao & Zhu$^+$ 19, Gao & Liao$^+$ 20]. Here, we expect that adding BERT-style noise [Devlin & Chang$^+$ 19] should also boost the performance of the models. As shown in Table 5.4, the best with-noise models are slightly better than the models without noise. Note that these results are cherry-picked by selecting the best set of other hyperparameters. Revisiting Table 5.1, in general, adding source-side noise is at least not harmful if only improving a little.

Table 5.4: Effect of source-side BERT-style artificial noise. The numbers are read off Table 5.1 by fixing the noise setting and picking the best set of other parameters in terms of Bleu score. Results suggest that adding BERT-style noise is slightly helpful in improving the translation performance.

| architecture | source noise | de-en | | en-ro | |
|---|---|---|---|---|---|
| | | Bleu[%] | Ter[%] | Bleu[%] | Ter[%] |
| transformer MT (enc-dec) | - | 34.9 | 44.5 | 26.0 | 54.8 |
| transformer LM (enc-only) | none | 34.6 | 45.2 | 26.1 | 54.8 |
| | BERT | 35.0 | 45.0 | 26.2 | 54.8 |

### Scheduling of Reconstruction Loss

Finally, we visit the hyperparameter choice of the reconstruction loss schedule. Contrary to Wang et al. [Wang & Tu$^+$ 21], where they find the two-step linearly decaying schedule for $\lambda_t$ to be important, our results suggest that the choice of the reconstruction loss schedule is

not critical. Shown in Table 5.5, the fluctuations in absolute BLEU are not large. The TER fluctuations are larger because we chose the other three hyperparameters based on the BLEU scores. Interestingly, even when completely zeroing out the reconstruction loss, i.e. $\lambda_t = 0$, the performance of TLMs can be good, if other hyperparameters are chosen carefully.

Table 5.5: Effect of reconstruction loss schedules. The numbers are read off Table 5.1 by fixing the corresponding schedule and picking the best set of other parameters in terms of BLEU score. Results suggest that the loss schedule is not important, and the reconstruction itself is even not necessary to achieve good translation performance.

| architecture | schedule | de-en | | en-ro | |
|---|---|---|---|---|---|
| | | BLEU[%] | TER[%] | BLEU[%] | TER[%] |
| transformer MT (enc-dec) | - | 34.9 | 44.5 | 26.0 | 54.8 |
| transformer LM (enc-only) | 0 | 34.9 | 45.0 | 26.2 | 54.8 |
| | 1 | 35.0 | 45.0 | 26.0 | 55.1 |
| | lin | 34.7 | 44.5 | 25.8 | 55.3 |
| | exp | 34.8 | 44.8 | 26.1 | 54.8 |

To quickly summarize the findings so far: the most critical hyperparameter in achieving good performance with TLMs seems to be the attention mask, i.e. properly modeling the internal dependencies and obtaining good source hidden representations. Performance-wise, the TLMs with a good hyperparameter setup are on par and even slightly better than the encoder-decoder baseline, showing that an encoder-only monolithic architecture is indeed a strong competitor against the encoder-decoder norm. Specifically, from here on, when we present results for TLMs, they refer to TLMs trained with a source autoencoding objective, with full attention mask, regularized with BERT-style noise and learned with a loss schedule of $\lambda = 1$. It is worth mentioning that the experiments presented in this chapter so far are biased experiments because we are making arguments about the hyperparameter choices based directly on test scores. However, as will be shown later, a good setup here also transfers to larger datasets and more extensive experimental settings.

### 5.4.2 Effect of Number of Parameters

Table 5.6: Effect of number of parameters on encoder-only translation language models. Varying only the number of encoder layers, the translation language models are trained under the same optimization setting. Results suggest that the parameter count needs to be similar to the encoder-decoder baseline to achieve comparable translation performance.

| architecture | # dec. layers | # enc. layers | # param. | de-en | |
|---|---|---|---|---|---|
| | | | | Bleu[%] | TER[%] |
| transformer MT (enc-dec) | 6 | 6 | 36.9M | 34.9 | 44.5 |
| transformer LM (enc-only) | - | 5 | 15.9M | 33.5 | 46.2 |
| | | 10 | 26.4M | 34.9 | 44.6 |
| | | 15 | 36.9M | 35.0 | 44.7 |
| | | 20 | 47.4M | 34.8 | 45.1 |

The four hyperparameters mentioned so far have different degrees of influence on the final

performance, and because they deal with data augmentation, modeling dependency, and reconstruction loss, the choice is parameter-count-independent. That said, we find that the overall model size, i.e. the parameter count, is important in achieving good performance. To this end, we vary the number of encoder layers in TLMs and compare them with the baseline encoder-decoder transformer model (see Table 5.6). Indeed, there are other ways to vary the total parameter count, e.g. changing the hidden dimension size. However, we have the intuition that the hidden dimension size has an impact on the expressiveness and separability of word vectors (think of the extreme case when we use only a scalar to represent a word[3]), therefore we decided to vary the number of hidden layers to reduce the chances of drawing wrong conclusions from phenomena with mixed causes. In Table 5.6, we train four encoder-only TLMs by varying the number of encoder layers while keeping the rest of the training settings fixed. As can be seen, when the parameter count is similar to the baseline encoder-decoder transformer model, the translation performance is comparable. Notice that the "15-layer" row here is a separate run from the "encoder-only, AE, full, BERT, 1" run in Table 5.1, and the results are comparable (Bleu: 35.0% versus 35.0% and Ter: 44.7% versus 45.0%). Because the IWSLT2014 German-English dataset used here is rather small, when the parameter count becomes larger, overfitting happens, and the performance slightly degrades. On the other hand, when the model is small and underfitted, e.g. with 10 and 15 encoder layers, the performance also deteriorated.

### 5.4.3 Beam Search

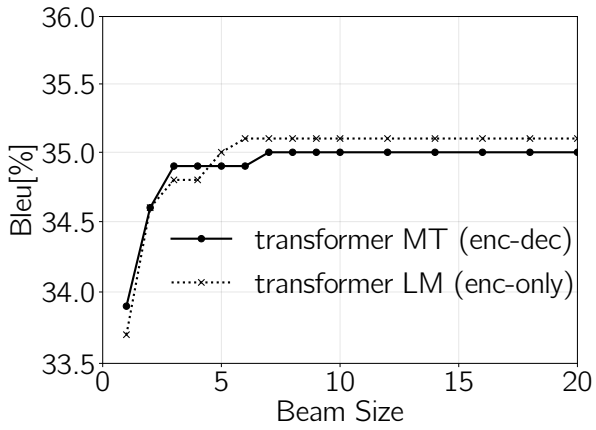



Figure 5.5: Behavior of beam search with respect to beam size for encoder-decoder transformer baseline and our encoder-only translation language model, on IWSLT2014 de-en. With beam size up to 20, as beam size increases, the Bleu[%] score first increases and then flattens in both cases, which is a typical curve seen for neural machine translation systems (e.g. Figure 7 in Gao et al. [Gao & Wang+ 20]). Results presented in Table 5.1 correspond to beam size 5 here.

The search steps in the experiments above were all done with beam search with a beam size of five. However, it also makes sense to verify the behavior with respect to changing beam sizes. To this end, we take the baseline encoder-decoder transformer baseline model and

[3]For instance, see Table VI in Sundermeyer et al. [Sundermeyer & Ney+ 15b] about how the hidden dimension size affects the perplexity of a neural language model. In addition, there exists work that explores how to explicitly encourage inter-class separability among words vectors [Huo & Gao+ 20], although in that work the hidden dimension size is kept fixed.

our encoder-only TLM, and sweep over different beam sizes and observe the BLEU[%] score changes. The results are shown in Figure 5.5. As seen, when the beam size increases, for both systems, the BLEU[%] initially increases and then flattens after beam size becomes larger than 7. The trend is similar for both systems, again exemplifying the translation capabilities of an encoder-only model. The similarity between the two curves can be expected, because the search processes are somewhat similar, hypothesizing one word at a time, updating the internal states with the hypothesized word, and repeating the process - the previously mentioned differences in dependencies, masking, etc. can all be conceptually thought of as internal workings in a black box. Notice that here we only go to beam sizes up to 20, because it is well-known in the literature that when the beam size becomes much larger, there is an unexpected degradation in the translation performance [Koehn & Knowles 17]. Although some works have tried to give an explanation [Shi & Xiao$^+$ 20, Liang & Wang$^+$ 22], it is not the focus of this work and we do not go further in this direction here.

### 5.4.4 Using Target-Side Monolingual Data

Table 5.7: Utilization of target-side monolingual data. For both encoder-decoder transformer model (traf.MT) and the encoder-only translation language model (traf.LM), target monolingual data is helpful in terms of development perplexity (devPPL). Multi-task training under-performs back-translation for the encoder-only model. With back-translation, the encoder-only model is on par with the encoder-decoder model. We also include results from Weiyue Wang [Wang 23] as a reference baseline.

| source | architecture | data condition | devPPL | zh-en | |
|---|---|---|---|---|---|
| | | | | Bleu[%] | TER[%] |
| [Wang 23] | traf.MT (enc-dec) | parallel only | - | 23.0 | 60.3 |
| ours | | | 6.91 | 23.2 | 60.5 |
| | | parallel + back-translated | 6.21 | 24.6 | 59.4 |
| | traf.LM (enc-only) | parallel only | 6.90 | 23.1 | 60.5 |
| | | parallel + target mono. | 6.70 | 23.0 | 61.4 |
| | | parallel + back-translated | 6.18 | 24.7 | 59.4 |

When training machine translation models, in addition to the supervised parallel data, it is common to make use of target-side monolingual data to improve the performance, thanks to its abundance [Koehn & Hoang$^+$ 07, Wuebker & Huck$^+$ 12, Freitag & Huck$^+$ 14, Sennrich & Haddow$^+$ 16a, Gulcehre & Firat$^+$ 17, Domhan & Hieber 17, Stahlberg & Cross$^+$ 18, Edunov & Ott$^+$ 18, Graça & Kim$^+$ 19]. The main approaches can be roughly categorized into three: 1. ensembling the translation model with an external language model; 2. training the translation model with an additional language modeling objective in a multi-task learning fashion; 3. augmenting the parallel data with synthetically generated back-translation data. The current understanding is that back-translation works best in practice [Barrault & Bojar$^+$ 19]. In principle, it is possible to apply ensembling on our translation language models. However, having to separately train and maintain a separate target language model does not align with our goal of achieving a compact and monolithic model. Therefore, we look at multi-task training and back-translation with translation language models.

In Table 5.7, we present comparisons between the encoder-decoder transformer baseline and our encoder-only model, in terms of utilization of target monolingual data, on WMT2017 Chinese-English. Specifically, the true parallel data contains about 17 million sentence pairs.

We sampled 5 million English sentences from the News crawl monolingual corpus[4], and we use the same separately pre-trained Chinese-English transformer model to do back-translation [Sennrich & Haddow+ 16a]. As seen in the table, the inclusion of additional target monolingual data is beneficial. However, compared to back-translation, directly training the TLM by iterating over translation and language modeling batches does not deliver good BLEU or TER scores. Looking at the development set perplexity (PPL), we observe that the monolingual data indeed improved the modeling, but not significantly enough to be reflected in the translation metrics. That said, turning our attention to the back-translation experiment, we clearly see that under a rich-resource setting, our TLM approach performs on par with the strong encoder-decoder baseline while maintaining a monolithic and streamlined architecture.

### 5.4.5 Multilingual Training

Next, we consider the performance of TLMs under multilingual training settings [Johnson & Schuster+ 17, Aharoni & Johnson+ 19]. With multilingual training, i.e. combining data from various language pairs, there are two major benefits:

- First, the model is more compact. Because model parameters are shared across more than one language pair, the need to maintain a quadratic number of systems with increasing numbers of supported languages is eased.
- Second, the transfer and zero-shot abilities of the model are improved. Because there may be intrinsic similarities among different languages, the model may benefit from seeing multiple languages during training.

Although it is possible to further define language-specific sub-networks to enable better learning of different languages [Lin & Wu+ 21], it is simpler and more common to simply train one joint model with all parameters being shared.

In the case of TLMs, multilingual training is straightforward. The handling of word embeddings is the same as in traditional methods, i.e. the vocabulary is shared across different languages. When the vocabulary is strictly disjoint (for instance, consider words made up of Chinese characters and the English alphabet respectively), this simply corresponds to concatenating two smaller word embedding matrices into a bigger one. We prepend language tags to source and target sentences before concatenating them and feeding them to the model. For example, the sentence:

$$\text{I like swimming .} \rightarrow \text{Ich gehe gerne Schwimmen .}$$

is fed to the system as:

<en> <s> I like swimming . </s> <de> <s> Ich gehe gerne Schwimmen . </s>

Of course, one may argue that the artificial start-of-sentence token is redundant in this case, but we decided to do it this way because of the simplicity of implementation[5] and also the results that will be shown later, i.e. it having minimal impact on final translation performance. Alternatively, one can also supply a translation direction tag like <en2de> before the source

[4] `https://data.statmt.org/news-crawl/`

[5] There is no need to change the tokenization code and we can simply prepend the corresponding language tags to the raw source and target sentences.

sentence, but because our setup is sufficient for the goal of comparing the multilingual setup between the baseline encoder-decoder transformer model and the TLM, we decided not to do this. Notice that when prepending language tags, the decoding of our system starts with the first actual target word, i.e. "Ich" in the example above.

To verify that TLMs are also on par with the baseline encoder-decoder transformer models under a multilingual setting, we create a custom multilingual dataset from the News-Commentary v16 dataset[6] [Tiedemann 12]. Specifically, we take all bilingual data in six directions involving German (de), Spanish (es) and French (fr): de-es, es-de, de-fr, fr-de, es-fr, fr-es, and for each direction we hold out the first 3000 translation sentence pairs as the test data and the last 3000 pairs as the development data. Then, we train both the baseline system and our system for the same number of optimization steps and pick the best checkpoint by comparing the development set perplexity. The results are presented in Table 5.8.

Table 5.8: BLEU[%] scores of multilingual transformer encoder-decoder system and our translation language model. We train both systems with the same number of optimization steps and select the best checkpoint with respect to development set perplexity (devPPL). The overall scores are recalculated by merging all test data and are not the average of previous columns.

| architecture | devPPL | de-es | es-de | de-fr | fr-de | es-fr | fr-es | overall |
|---|---|---|---|---|---|---|---|---|
| transformer MT (enc-dec) | 6.17 | 25.7 | 19.1 | 21.3 | 16.9 | 24.6 | 26.2 | 22.5 |
| transformer LM (enc-only) | 6.06 | 25.5 | 18.8 | 20.7 | 16.6 | 24.4 | 26.0 | 22.3 |

As seen, the encoder-only TLM performs similarly when compared to the baseline system. Although an absolute $-0.2$ BLEU[%] score difference can be observed, we think it is within the normal range of statistical fluctuations because the development perplexity shows that the encoder-only TLM is even slightly better than the baseline. Therefore, we conclude here that under a multilingual setup, the encoder-only TLM also performs on par with the encoder-decoder model, while maintaining a simpler and monolithic architecture.

### 5.4.6 Pseudo Parallel Data in Large Language Models

Recently, large language models [BigScience Workshop 22, Radford & Wu$^+$ 19b, Brown & Mann$^+$ 20], especially chatbots based on them [Adiwardana & Luong$^+$ 20, Schulman & Zoph$^+$ 22], have gained lots of attention and popularity. One impressive attribute of these models is their ability to perform zero-shot/few-shot/supervised translation. One central question revolving around their translation capability is: why can they do translation?

We know that to enable a statistical model to generalize on unseen data/tasks, we need both good learning of the empirical distribution on the training data and good smoothing/regularization. In the literature, it is not always clear to what extent parallel data is included or employed to train the large language models[7]. That said, it is likely that large quantities of "pseudo" parallel data is included in the training data[8]. With our translation language model, and data augmentation/alternation, it is possible to get a peek into the problem.

[6] The data is retrieved from `https://data.statmt.org/news-commentary/v16/`.

[7] A recent work that delves into this aspect is Briakou et al. [Briakou & Cherry$^+$ 23]

[8] A concrete example of a naturally occurring bilingual sentence is "I'm not the cleverest man in the world, but like they say in French: Je ne suis pas un imbecile [I'm not a fool]." (first example from Table 1 in Radford et al. [Radford & Wu$^+$ 19a])

Table 5.9: Effect of including (pseudo) machine translation data in translation language model (TLM) training on IWSLT2014 German-English. All models are trained till convergence with the same optimization parameters, and the checkpoints with the best development set perplexity (devPPL) are selected. The devPPL, BLEU [%] and TER [%] scores are reported on the translation test set. These experiments are different runs from Table 5.1, and therefore the results in row 1 and 2 are slightly different from before.

| row id | model | concat. | para. | | mono. | | devPPL | BLEU[%] | TER[%] |
|---|---|---|---|---|---|---|---|---|---|
| | | | true | pseudo | de | en | | | |
| 1 | enc-dec | 1-1 | 100% | - | - | - | 4.84 | 35.0 | 44.1 |
| 2 | enc-only | 1-1 | 100% | - | - | - | 4.69 | 34.9 | 45.1 |
| 3 | | | 50% | - | - | - | 6.34 | 31.2 | 48.2 |
| 4 | | | 50% | - | 50% | - | 6.12 | 31.9 | 48.6 |
| 5 | | | 50% | - | - | 50% | 5.83 | 31.6 | 50.3 |
| 6 | | | - | 50% | - | - | 3631.38 | 0.0 | 100.0 |
| 7 | | | - | 50% | 50% | - | 3426.84 | 0.1 | 100.0 |
| 8 | | | - | 50% | - | 50% | 3805.95 | 0.0 | 100.0 |
| 9 | | 2-2 | 99.9% | - | - | - | 5.05 | 33.3 | 46.8 |
| 10 | | 4-4 | 98.0% | - | - | - | 5.73 | 31.0 | 51.4 |
| 11 | | 8-8 | 59.3% | - | - | - | 6.28 | 13.6 | 100.0 |
| 12 | | 16-16 | 1.5% | - | - | - | 19.44 | 1.7 | 100.0 |

In Table 5.9, we perform a set of experiments changing the data condition and batch preparation. Specifically:

- We concatenate several sentences on the source and target side respectively, to simulate the task of learning long-range dependencies. The "concat." column shows how many sentences are concatenated into a longer sequence for TLM training. For instance, "2-2" means "ABXY" is fed to the model, where English sentences X and Y are translations of German sentences A and B, respectively. The maximum sequence length of both source and target is set at 1024.

- We alter the parallel training data by breaking up the parallelism, and simulate an unsupervised learning scenario where both source and target sentences are existent, but not parallel. The percentages under the "para." (parallel) and "mono." (monolingual) columns denote how much of the original data is used in training. The "pseudo" column indicates that the parallelism of the source and target text is broken, i.e. although bilingual, the source and target sentences are not direct translations of one another anymore.

- We combine parallel and monolingual data to simulate an implicit multi-task learning scenario, where both types of data are existent in the data mix. The monolingual data is the source/target side of the original parallel data, i.e. no additional monolingual data is used in these setups. When the monolingual data is used in the data mix, it is always used as the target sentence with no source context, resembling a monolingual language modeling task (i.e. "0-1", if one will).

The metrics are development perplexity, test BLEU score and test TER score.

As seen, when trained on the same parallel data, the encoder-only translation language model (TLM) performs on par with the encoder-decoder transformer (row 2 versus row 1, results agree with previous sections). When only half of the parallel data is utilized, there is a degradation in translation performance (row 3 versus row 2). Including further monolingual data, whether source or target, in a multi-task learning fashion, does not seem to significantly improve or hurt the translation performance[9] (row 4, 5 versus row 3). When the parallel data is of much lower quality, i.e. bilingual but not even maintaining the parallelism in our setup, the translation performance drops dramatically (row 6, 7, 8 versus row 3). We think the breakdown of the translation capability in our setup comes from the small scale of the experiment. In the literature, it is reported [Wei & Tay$^+$ 22] that large language models exhibit emergent abilities, meaning that their performance on downstream tasks suddenly improves after a certain model/data size threshold is met[10]. The poor translation qualities of rows 6, 7 and 8 in Table 5.9 could be simply because our data and model are too small. Finally, expanding the context to longer ranges also seems to hurt the translation performance a lot (row 9, 10, 11, 12 versus row 2). Our explanation for the degradation is twofold: 1. far fewer training sentence pairs are used because the length of the concatenated sequence exceeds the maximum sequence length processable by the model and 2. a mismatch in training and test conditions. For example, for row 9, we train with "2-2" but test with "1-1". Naively constructing the test prefix to be "2-1" and trying to search for the second target sentence sometimes leads to the model repeating the first target sentence. Alternatively, one may sample from the model output distributions and generate a sentence in the target language either until the artificial end token or a pre-defined maximum hypothesis length. In this case, we query the model with "1-0" and observe a degradation to a Bleu score of 17.9% and a Ter score of 67.5% using the sampling approach. The models are trained to process long context and predict long translations, but are only presented with short sentences and asked to predict short translations. In fact, examining the training perplexity shows that the models had no problem learning the training distributions. However, the generated hypotheses are significantly longer than they are supposed to be, meaning that the model tries to artificially prolong the hypothesis generation to the length that it is used to seeing during the training phase.

With these observations, our insights are:

1. Good-quality parallel data is key to good translation performance.
2. Mixing translation and language modeling data does not have a huge impact on the translation performance.
3. The model is able to learn long-range dependencies, but one has to be careful in designing the search process to properly query the model.

## 5.5 Summary

In this chapter, we visited the issues of model compactness for neural machine translation. For the task of machine translation we proposed a simple, monolithic model architecture resembling language models, which is trained on concatenations of source and target sentences and does not

[9] Note that in this setting we do not supply any "task token", meaning the model is free to decide on whether to do language modeling or translation. Such errors are potentially avoidable with additional task tokens, or a reinforcement learning approach [Ouyang & Wu$^+$ 22] where the task information is supplied to the model via natural language text.

[10] This property, however, may be an artifact of how the evaluation is done [Schaeffer & Miranda$^+$ 23].

require a strict separation between the encoder and the decoder. We discussed the differences in model dependencies between traditional encoder-decoder transformer models and our encoder-only translation language models. We further delved into the details of attention masks and the formulation of the source reconstruction loss. The experimental results confirmed our thinking, that a separate decoder is not necessary for good translation performance. We verified the claim under various experimental settings, such as traditional bilingual training, changing the model size, using different beam sizes, making use of additional target-side monolingual data, and multilingual training. In light of recent rapid developments of large language models, we also discussed why many of them exhibit zero-shot/few-shot translation capabilities. In conclusion, our results show that contrary to previous practice, the strict separation of an encoding and a decoding step is not necessary for good translation performance, and a joint, monolithic encoder-only model can perform as well as the state-of-the-art encoder-decoder models for translation.

# 6. aseq: a Sequence Toolkit

The fields of neural language modeling and neural machine translation are fast-moving. In recent years, the number of major conference papers has grown very rapidly [ACL 23]. This makes it increasingly hard to keep track of recent progress, as well as recreate interesting work in existing software. Therefore, the author wrote some custom software called "`aseq`", in order to be agile in writing prototypes for new ideas. In this chapter, we showcase a few features of the toolkit.

## 6.1 Related Work

During the period of this work, most software for neural language modeling and neural machine translation was based on generic neural network back-ends such as TensorFlow [Abadi & Agarwal$^+$ 15], PyTorch [Paszke & Gross$^+$ 19], Mxnet [Chen & Li$^+$ 15], etc., which provides essential utilities to build and evaluate neural-network-based models, such as tensors, neural network modules, automatic differentiation, training criteria, optimizers, schedulers, etc. While earlier pioneers did not have the benefit of such back-ends and had to resort to building their own tools, newcomers often provide simple interfaces and focus on extensions aiming at more efficient computation, easy extension, simple sharing etc. In no particular order, for the convenience of the interested reader, we attach a far-from-exhaustive list of some examples of these toolkits: RNNLM [Mikolov & Kombrink$^+$ 11a], rwthlm [Sundermeyer & Schlüter$^+$ 14], CUED-RNNLM [Chen & Liu$^+$ 16a], minGPT [Karpathy 23], Tensor2Tensor [Vaswani & Bengio$^+$ 18], Sockeye [Hieber & Domhan$^+$ 18], Joey NMT [Kreutzer & Bastings$^+$ 19], NMT-Keras [Peris & Casacuberta 18], fairseq [Ott & Edunov$^+$ 19], HuggingFace [Wolf & Debut$^+$ 20], OpenNMT [Klein & Kim$^+$ 17], etc. Along the spectrum from a toy example to industrial-level software, our toolkit lies somewhere between an end-of-semester big course project and an educational codebase for newcomers into the field.

## 6.2 Feature Showcase

### 6.2.1 uniblock

When cleaning up raw textual data for natural language processing applications, one often faces the task of getting rid of illegal or foreign characters. This is typically approached by scripting custom rules on a case-by-case basis. In Gao et al. [Gao & Wang$^+$ 19], we propose that this step can be simplified by learning a Bayesian Gaussian Mixture Model (BGMM) [scikit learn 23] on the Unicode block [Unicode 23] distribution on some "clean" data (e.g. the development set). The BGMM can then be used to score dirty training data, and the scores are

then useful for filtering before actual neural network training. We gave this method a simple name: "`uniblock`", to highlight how the features are constructed.

Table 6.1: Sample sentence pairs from the WMT2018 Chinese-English dataset and their `uniblock` scores. Notice how low scores are assigned to sentences with foreign characters and characters from rare Unicode blocks, and high scores are assigned to sentences with characters that are mostly in-block. Zero scores are artificially assigned to sentences with characters from unseen Unicode blocks. "Bad" characters are underlined for easy reading.

| side | score | parallel text |
|---|---|---|
| zh | -48523.3 | リリスノト |
| en | 53.8 | Open Directory - Unix |
| zh | 67.7 | 健康、整洁、卫生和清洁技术组织 |
| en | -158.6 | Salubrité, propreté, hygiène et techniques d'assainissement |
| zh | 0.0 | 从系统目录→位置→传送我的位置... |
| en | 0.0 | From System Menu → Location → Send My Location... |
| zh | 36.0 | 在25℃下测定了样品的总交换容量和平衡等温线. |
| en | 40.7 | ... and equilibria isotherm at 25 ℃ were determined. |
| zh | 68.6 | 财务政策及发展小组 |
| en | 53.8 | Financial Policy and Development Unit |

Table 6.2: Bleu [%] scores of neural machine translation systems trained with different amounts of data, filtered with `uniblock`, on WMT2018 Chinese-English (zh-en) and English-Chinese (en-zh). Without the need to script character-filtering rules, expected improvements from removing dirty data are observed.

| data | zh-en | | en-zh | |
|---|---|---|---|---|
| | test17 | test18 | test17 | test18 |
| 100% | 25.0 | 24.5 | 30.1 | 33.0 |
| 90% | 25.2 | 25.6 | 30.9 | 33.1 |
| 80% | 24.3 | 25.3 | 30.3 | 33.2 |
| 70% | 24.3 | 24.8 | 30.2 | 33.0 |

In Table 6.1, we showcase a few examples of scored parallel Chinese-English data, as well as their `uniblock` scores. As seen, sentences with "bad" characters are given low scores while sentences with "good" characters are scored higher. In Table 6.2, we show results of translation experiments done with data filtered with `uniblock` to different degrees. As seen, the method is effective in selectively getting rid of bad training pairs and the resulting smaller training data in turn slightly improves the test Bleu [%] scores.

### 6.2.2 Word Vector Norm Initialization

In Herold et al. [Herold & Gao+ 18], we noticed that the norms of word vectors learned by a neural language model follow a pattern that is related to the word counts in the training data. Specifically, the more frequent a word is, the longer the learned word vector norm is. Empirically, the norm and the word frequency rank seemed to have a logarithmic relationship.

This in turn motivated us to ask: if such a pattern is what the neural network learns, why not kickstart the training by initializing the word vectors to roughly follow this relationship?

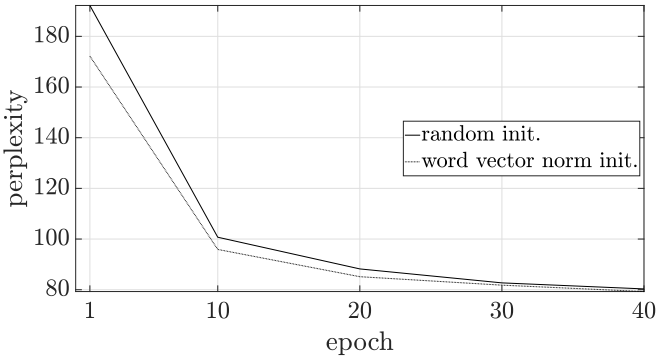



Figure 6.1: Comparison of word-count-based word vector norm initialization against random word vector initialization on WikiText-2 [Merity & Xiong+ 17]. Test perplexities are reported. Both models eventually converge to similar perplexities with extended training. The model whose word vector norms are initialized with word count information has a slight advantage in early epochs of training.

In Figure 6.1, we plot the test perplexities in early epochs of the training of such a model and a baseline model whose word vectors are initialized randomly. As seen, there is a clear benefit with such initialization when the training first starts, but both models eventually converge to a similar quality. This experiment was conducted on a very small dataset. However, if the training data is so large that e.g. 100 full epochs over the entire training data is simply infeasible, the method may prove useful in improving the training result.

### 6.2.3 Incremental Decoding

Table 6.3: Incremental decoding [Fairseq 23, Ott & Edunov+ 19] improves search speed without hindering translation performance. The results are reported on the WMT2014 English-German dataset.

| model | implementation | caching | search time (s) | BLEU (%) | TER (%) |
|---|---|---|---|---|---|
| LSTM RNN | [Luong & Pham+ 15] | - | - | 20.9 | - |
| | ours | none | 4077.21 | 21.1 | 64.9 |
| | | encoder | 1848.86 | 21.1 | 64.9 |
| | | encoder + decoder | 297.75 | 21.1 | 64.9 |
| transformer | [Vaswani & Shazeer+ 17] | - | - | 27.3 | - |
| | ours | none | 1166.99 | 27.1 | 55.2 |
| | | encoder | 797.48 | 27.1 | 55.2 |
| | | encoder + decoder | 710.72 | 27.1 | 55.2 |

Incremental decoding [Fairseq 23] is a trick to boost the decoding speed of neural machine translation models. The motivation is twofold: 1. The target sequence probability is factorized

autoregressively; 2. The calculation of the decoder-internal states only depends on left-side inputs. In other words, when decoding for the next step $i$, many of the previously calculated internal states do not have to be recalculated. The trickiness of the method mainly lies in the correct caching and retrieval of the internal states, because the active partial hypotheses in the beam may change during beam search. We implement the method and present the results in Table 6.3. As seen, compared to no caching at all, or simply caching the encoder outputs (because the source sentence does not change during decoding), incremental decoding gives significant speedups in search.

### 6.2.4 Grammar Sugar for Array Jobs

Many of the experiments involving neural networks can be thought of as a hyperparameter tuning problem. While sophisticated methods like Bayesian optimization [Snoek & Larochelle$^+$ 12] do exist, for fair comparisons and explanatory results, most experiments are done via a grid search in the combinatorial space of the interesting hyperparameters. While in the literature there exists sophisticated software [Peter & Beck$^+$ 18] that exactly deals with the complexity of the problem, here we provide an extremely lightweight solution which is essentially a grammar sugar.

Assume we have a certain job, and we want to loop over two options `opt1` and `opt2` for its hyperparameter `arg`, we can make use of the f-string [Python 23] in Python, and write both the bash submission logic as well as the grid search logic (notice how the f-string syntax is preserved in the definition of `t`) into a file named "generate.py":

```
t = r'''#!/usr/bin/env sh

dataset=/path/to/dataset
job=job.{arg}

submit -gpu 1 -t 24:00:00 -n $job -o $job.log "
    aseq job \
        --dataset $dataset \
        --arg {arg}
"
'''

import os
from aseq import tstr
for arg in ['opt1', 'opt2']:
    with open(f'job.{arg}.submit.sh', 'w') as fo:
        fo.write(eval(tstr(t)))
```

Running the code above will then result in a folder structure as below:

```
generate.py
job.opt1.submit.sh
job.opt2.submit.sh
```

And the content of `job.opt2.submit.sh` is:

```
#!/usr/bin/env sh

dataset=/path/to/dataset
job=job.opt2

submit -gpu 1 -t 24:00:00 -n $job -o $job.log "
    aseq job \
        --dataset $dataset \
        --arg opt2
"
```

The `t` variable is short for "template-string", and a `tstr` function needs to be implemented to wrap the template string with a literal `f` such that the formatting options can be effective[1]:

```
def tstr(s):
    return "fr'''" + s +  "'''"
```

The benefits of such a grammar sugar are two-fold: 1. The space, tab, newline, etc. symbols that one may desire in a bash job submission script are preserved; 2. The syntax of the f-string is preserved. The toy example may seem underwhelming, but the simplicity and the clarity of the method quickly becomes visible, when one needs to write submission scripts for a grid search involving e.g. three hyperparameters, each taking five possible values, ranging from numbers, strings, array-like structures, and all resulting in weird-formatted argument values.

[1] The author wrote an initial implementation, which was (thankfully) greatly simplified to a one-liner by David Thulke.

# 7. Scientific Achievements

In this dissertation, three topics related to improvements in training and modeling for neural language modeling and neural machine translation are discussed. In accordance with Chapter 2, we briefly outline the scientific achievements of this work:

1. Sampling-based training criteria for neural language modeling:
   - We revisited three traditional training criteria, namely mean squared error, binary cross entropy and cross entropy, and highlighted that the training and testing inefficiencies originate from the need to traverse over the large vocabulary.
   - We summarized existing sampling-based training criteria, and made the important observation that the desired class posterior distribution can be recovered via a correction step for the criteria considered.
   - We further proposed a self-normalized importance-sampling training criterion, and with extensive experiments, we were able to show that it can perform on par with the prominent noise contrastive estimation method.
2. Combating overfitting issues in neural machine translation:
   - We considered extensive experimental settings for label smoothing, such as which positions to smooth, with what auxiliary distribution to smooth, and how strongly to smooth, and found better training recipes that can improve the translation quality.
   - We further considered input smoothing and found that plugging in simple auxiliary distributions such as the uniform and the unigram distribution both at the input side and at the output side can significantly boost the translation performance.
   - We proposed a novel multi-agent mutual learning algorithm, which enables inter-agent supervised training, at both sentence-level and token-level, and showed that the ensemble of these agents is superior to simple log-linear model combinations.
3. More compact model via a language modeling approach for machine translation:
   - We developed a language modeling approach for machine translation where an encoder-only model is trained on concatenations of source and target sentences, and extensively studied different training settings.
   - We showed that the encoder-only language modeling approach performs on par with strong encoder-decoder transformer baseline models under various settings.
   - We discussed the implications of such a language modeling approach, and argued that the translation capabilities of recent large language models are likely rooted in the existence of parallel training data.

# 8. Individual Contributions

According to §5.6 of the doctoral guidelines of the RWTH Aachen University, Faculty of Mathematics, Computer Science and Natural Sciences, September 7, 2018, we list the following publications of the author, and highlight his individual contributions:

## Publications Directly Related to This Dissertation

- [Gao & Thulke$^+$ 21]: Y. Gao and D. Thulke put forward the theory and derived the model optimums together. A. Gerstenberger and K. Tran conducted the experiments. Experimental design was a joint effort.

- [Yang & Gao$^+$ 22]: Z. Yang initially came up with the self-normalization idea, and Y. Gao and Z. Yang worked out the theory together. Y. Gao, Z. Yang and A. Gerstenberger designed the experiments together and Z. Yang conducted the experiments.

- [Gao & Wang$^+$ 20]: Y. Gao came up with the initial extensions. Y. Gao and C. Herold came up with the solutions to the confidence penalty, and Z. Yang showed the detailed derivation. Y. Gao performed most experiments and analyses. W. Wang conducted the high-resource experiments.

- [Gao & Liao$^+$ 20]: Y. Gao proposed to explore alternative prior distributions and combine the smoothing methods. Y. Gao and B. Liao jointly designed the expriments. B. Liao conducted the experiments and Y. Gao provided the analyses.

- [Liao & Gao$^+$ 20]: B. Liao came up with the two-step mutual learning extensions after Y. Gao introduced the multi-agent work [Bi & Xiong$^+$ 19] to him. Y. Gao and B. Liao worked jointly on the methodology and experimental designs. B. Liao conducted the experiments.

- [Gao & Herold$^+$ 22a]: Y. Gao initially heard of the concept of training language models for translation on source and target concatenations from K. Irie [Irie 20]. Y. Gao proposed extensions in attention, auxiliary loss, etc. C. Herold and Z. Yang helped to polish the methodology. Y. Gao did the implementation and carried out most experiments. C. Herold conducted the Chinese-English experiments.

- [Herold & Gao$^+$ 18]: Y. Gao discovered the word vector norm distribution and proposed the initialization method. C. Herold and Y. Gao jointly refined the idea and worked on the regularization method. Implementation and experiments were joint efforts.

- [Gao & Wang+ 19]: Y. Gao came up with the idea, implemented the tool and did most experiments. W. Wang helped to refine the concept and conducted the translation experiments.
- [Gao & Herold+ 22b]: Y. Gao proposed the extensions, did implementations and conducted the experiments. C. Herold and Z. Yang helped to refine the extensions and to solve implementation issues.

## Other Publications

- [Rosendahl & Herold+ 19]: The work was a joint effort for the participation in the shared task in WMT. Y. Gao mainly contributed to the Chinese-English system, together with W. Wang.
- [Kim & Gao+ 19]: Y. Kim proposed the method and did most experiments. Y. Gao helped to refine the idea and conducted experiments in Belarusian-English and Slovenian-English.
- [Gao & Herold+ 19]: Y. Gao came up with the initial concept of context-dependent logit calculation and softmax. C. Herold and W. Wang helped polish the idea. Y. Gao and C. Herold implemented the kernels. Y. Gao ran the language modeling and machine translation experiments. C. Herold ran the analysis experiments.
- [Huo & Gao+ 20]: Y. Gao proposed the idea to apply large-margin softmax in language modeling. J. Huo and W. Wang contributed with the theory and methodology. J. Huo implemented the methods and conducted experiments.
- [Yang & Gao+ 20]: Y. Gao made the initial proposal to introduce a length-prediction loss in the encoder. Z. Yang, Y. Gao and W. Wang designed the sub-network and designed the experiments together. Z. Yang implemented the concept and conducted the experiments.
- [Huo & Herold+ 20]: The work was a joint effort from RWTH Informatik 6 and eBay in developing document-level translation systems. Y. Gao put forward the idea of using target-side context and contributed to the methodology and experimental designs.
- [Wang & Yang+ 21b]: W. Wang proposed the direct hidden Markov models with transformer. Z. Yang did the implementation and most experiments. Y. Gao contributed to the methodology, derivation and automatic differentiation discussions.
- [Tran & Thulke+ 22]: V. Tran, D. Thulke, Y. Gao and C. Herold all contributed to the Top-$K$-Train/Search extensions, as well as the experimental designs. V. Tran implemented and ran the experiments.

# A. Overview of the Corpora

## A.1 Switchboard 300h Language Modeling Task

The statistics of the Switchboard 300h language modeling task are shown in Table A.1. We follow Kazuki Irie [Irie 20] to prepare the validation set. The vocabulary size is 30K. Test results are reported on the Switchboard and CallHome partitioning of Hub5_00. Further detailed model parameters are documented in Gao et al. [Gao & Thulke+ 21].

Table A.1: Switchboard 300h language modeling task corpora statistics.

| | | # running words | OOV (%) | avg. length |
|---|---|---|---|---|
| train | | 26.7M | 1.6 | 11.2 |
| validation | | 133K | 0 | 12.8 |
| test | Switchboard | 22K | 0.7 | 12.3 |
| | CallHome | 23K | 1.6 | 9.1 |
| | all | 45K | 1.1 | 10.4 |

## A.2 LibriSpeech Language Modeling Task

The statistics of the LibriSpeech language modeling task are summarized in Table A.2. We follow Kazuki Irie [Irie 20] to prepare the dataset. Notably, the "clean" and "other" subsets are concatenated together to form the language modeling datasets. The vocabulary size is 200K. Further detailed model parameters are documented in Gao et al. [Gao & Thulke+ 21].

Table A.2: LibriSpeech language modeling task corpora statistics.

| | # running words | OOV (%) | avg. length |
|---|---|---|---|
| train | 813M | 0.18 | 19.97 |
| validation | 105K | 0.38 | 18.92 |
| test | 105K | 0.42 | 18.87 |

## A.3 AppTek English Language Modeling Task

The statistics of the AppTek English language modeling task are summarized in Table A.3. This dataset mainly contains textual data that is geared towards telephony speech. The vocabulary size is 250K. Note that the total text data contains 7.4B running words, and in Table A.3, the number 694M refers to the number of running words that was directly used in neural language model training. Further detailed model parameters are documented in Gao et al. [Yang & Gao+ 22].

Table A.3: AppTek English language modeling task corpora statistics.

| | # running words | OOV (%) | avg. length |
|---|---|---|---|
| train | 694M | 4.07 | 8.7 |
| validation | 18K | 0.33 | 13.8 |
| test | 142K | 0.15 | 18.0 |

## A.4 IWSLT2014 German-English Translation Task

The original task is from the 11th International Workshop on Spoken Language Translation[1] (IWSLT2014). The original data[2] is pre-processed with the script[3] from the FAIRSEQ [Ott & Edunov+ 19] toolkit. The texts are lower-cased and tokenized with the Moses [Koehn & Hoang+ 07] tokenizer. Training sentence pairs are additionally filtered according to source-to-target ratio and sentence lengths. A 22:1 split is done to obtain the training and validation data. All available development and test sets are concatenated to be used as the test data[4, 5]. The subword-level vocabulary is learned with joint-BPE [Sennrich & Haddow+ 16b] with 10k merge operations. The out-of-vocabulary (OOV) rates and the test scores are reported on tokenized data.

Table A.4: IWSLT14 German-English translation task corpora statistics.

| | | German | English |
|---|---|---|---|
| vocabulary | subword vocabulary | 10.1K | |
| train | sentence pairs | 160.2K | |
| | running subwords | 3.9M | 3.8M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 7.3K | |
| | running subwords | 175.3K | 171.3K |
| | OOV subwords | 0 (0.0%) | 1 (0.0%) |
| test | sentence pairs | 6.8K | |
| | running subwords | 155.1K | 150.2K |
| | OOV subwords | 47 (0.0%) | 5 (0.0%) |

[1] https://workshop2014.iwslt.org
[2] https://wit3.fbk.eu/2014-01
[3] https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-iwslt14.sh
[4] Felix Schmidt found that there are 72 misaligned sentence pairs in this test set.
[5] https://github.com/facebookresearch/fairseq/issues/4146

## A.5 IWSLT2014 Dutch-English Translation Task

The data sources and pre-processing are the same as that described in Section A.4.

Table A.5: IWSLT14 Dutch-English translation task corpora statistics.

| | | Dutch | English |
|---|---|---|---|
| vocabulary | subword vocabulary | 10.1K | |
| train | sentence pairs | 153.6K | |
| | running subwords | 3.5M | 3.6M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 7.0K | |
| | running subwords | 162.2K | 166.9K |
| | OOV subwords | 3 (0.0%) | 1 (0.0%) |
| test | sentence pairs | 5.4K | |
| | running subwords | 116.5K | 120.6K |
| | OOV subwords | 5 (0.0%) | 0 (0.0%) |

## A.6 IWSLT2014 Spanish-English Translation Task

The data sources and pre-processing are the same as that described in Section A.4.

Table A.6: IWSLT14 Spanish-English translation task corpora statistics.

| | | Spanish | English |
|---|---|---|---|
| vocabulary | subword vocabulary | 10.2K | |
| train | sentence pairs | 169K | |
| | running subwords | 4.0M | 4.0M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 7.7K | |
| | running subwords | 182.6K | 183.1K |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| test | sentence pairs | 6.8k | |
| | running subwords | 155.1k | 150.2k |
| | OOV subwords | 23 (0.0%) | 0 (0.0%) |

## A.7 WMT2016 English-Romanian Translation Task

We follow Wang [Wang 23] for the preparation of this dataset. The original task is from the First Conference On Machine Translation[6, 7]. The original data is taken from the Europarl [Koehn 05] corpus[8] and the OPUS [Tiedemann & Nygaard 04] corpus[9]. The pre-processing script is adapted from the tutorial[10] from the Marian [Junczys-Dowmunt & Grundkiewicz$^+$ 18] toolkit. The punctuation is normalized and Romanian diacritics are removed. The Moses [Koehn & Hoang$^+$ 07] tokenizer is used for tokenization. Training sentence pairs are additionally filtered according to source-to-target ratio and sentence lengths and true-cased. The `newsdev2016` set is used as validation data and the `newstest2016` set is used as test data. The subword-level vocabulary is learned with joint-BPE [Sennrich & Haddow$^+$ 16b] with 20k merge operations. The out-of-vocabulary (OOV) rates and the test scores are reported on tokenized data.

Table A.7: WMT2016 English-Romanian translation task corpora statistics.

| | | English | Romanian |
|---|---|---|---|
| vocabulary | subword vocabulary | 20.3K | |
| train | sentence pairs | 612K | |
| | running subwords | 17.6M | 18.8M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 2.0K | |
| | running subwords | 63.2K | 72.5K |
| | OOV subwords | 0 (0.0%) | 16 (0.0%) |
| test | sentence pairs | 2.0K | |
| | running subwords | 59.5K | 68.0K |
| | OOV subwords | 2 (0.0%) | 16 (0.0%) |

## A.8 WMT2014 English-German Translation Task

We follow Wang [Wang 23] for the preparation of this dataset. The original data is pre-processed with the script[11] from the FAIRSEQ [Ott & Edunov$^+$ 19] toolkit. The validation set is created by performing a 99:1 split on the original pre-processed training data. The `newstest2014` set is used as test data. The subword-level vocabulary is learned with joint-BPE [Sennrich & Haddow$^+$ 16b] with 40k merge operations. The out-of-vocabulary (OOV) rates are reported on tokenized data.

[6] Previously known as Workshop On Statistical Machine Translation.
[7] `https://www.statmt.org/wmt16/`
[8] `http://www.statmt.org/europarl/v7/ro-en.tgz`
[9] `http://opus.lingfil.uu.se/download.php?f=SETIMES2/en-ro.txt.zip`
[10] https://marian-nmt.github.io/examples/mtm2017/intro/
[11] `https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2de.sh`

Table A.8: WMT2014 English-German translation task corpora statistics.

| | | English | German |
|---|---|---|---|
| vocabulary | subword vocabulary | 43.6K | |
| train | sentence pairs | 4.0M | |
| | running subwords | 112M | 115M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 40.1k | |
| | running subwords | 1.14M | 1.17M |
| | OOV subwords | 8 (0.0%) | 7 (0.0%) |
| test | sentence pairs | 3.0K | |
| | running subwords | 59.5K | 68.0K |
| | OOV subwords | 0 (0.0%) | 768 (0.1%) |

## A.9 WMT2018 Chinese-English Translation Task

We follow Wang [Wang 23] for the preparation of this dataset. Uniblock [Gao & Wang+ 19] is used to filter the training data. The `newsdev2017` is used as the validation set. The `newstest2017` is used as the test set. The subword-level vocabulary is learned separately for Chinese and English with BPE [Sennrich & Haddow+ 16b] and 32k merge operations.

Table A.9: WMT2018 Chinese-English translation task corpora statistics.

| | | Chinese | English |
|---|---|---|---|
| vocabulary | subword vocabulary | 47.0k | 32.2k |
| train | sentence pairs | 17.0M | |
| | running subwords | 372M | 408M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 2.0K | |
| | running subwords | 58.7K | 65.1K |
| | OOV subwords | 9 (0.0%) | 17 (0.0%) |
| test | sentence pairs | 2.0K | |
| | running subwords | 55.9K | 59.7K |
| | OOV subwords | 8 (0.0%) | 12 (0.0%) |

## A.10 News-Commentary v16 Multilingual Translation Task

A custom multilingual dataset is created from the newscommentary v16[12] dataset [Tiedemann 12]. We take the data involving German, Spanish and French in all six directions. For the raw data in each direction, we take the first 3000 sentence pairs as test data and the last 3000 sentence pairs as development data. The subword-level vocabulary is learned with joint-BPE [Sennrich & Haddow$^+$ 16b] with 32k merge operations. In Table A.10, the statistics of the concatenated corpora, i.e. all six directions, are reported.

Table A.10: News-Commentary v16 multilingual translation task corpora statistics.

| | | source | target |
|---|---|---|---|
| vocabulary | subword vocabulary | 32.5K | |
| train | sentence pairs | 3.5M | |
| | running subwords | 54.9M | 54.9M |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| validation | sentence pairs | 18.0K | |
| | running subwords | 578.4K | 578.4K |
| | OOV subwords | 0 (0.0%) | 0 (0.0%) |
| test | sentence pairs | 18.0K | |
| | running subwords | 557.4K | 557.4K |
| | OOV subwords | 1 (0.0%) | 1 (0.0%) |

[12] `https://data.statmt.org/news-commentary/v16/`

# List of Figures

# List of Tables

# Bibliography


[Abadi & Agarwal+ 15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[ACL 23] ACL: ACL Conference Acceptance Rates, 2023.

[Adiwardana & Luong+ 20] D. Adiwardana, M.T. Luong, D.R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, Q.V. Le: Towards a Human-like Open-Domain Chatbot, 2020.

[Aharoni & Johnson+ 19] R. Aharoni, M. Johnson, O. Firat: Massively Multilingual Neural Machine Translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3874–3884, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Akhbardeh & Arkhangorodsky+ 21] F. Akhbardeh, A. Arkhangorodsky, M. Biesialska, O. Bojar, R. Chatterjee, V. Chaudhary, M.R. Costa-jussa, C. España-Bonet, A. Fan, C. Federmann, M. Freitag, Y. Graham, R. Grundkiewicz, B. Haddow, L. Harter, K. Heafield, C. Homan, M. Huck, K. Amponsah-Kaakyire, J. Kasai, D. Khashabi, K. Knight, T. Kocmi, P. Koehn, N. Lourie, C. Monz, M. Morishita, M. Nagata, A. Nagesh, T. Nakazawa, M. Negri, S. Pal, A.A. Tapo, M. Turchi, V. Vydrin, M. Zampieri: Findings of the 2021 Conference on Machine Translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pp. 1–88, Online, Nov. 2021. Association for Computational Linguistics.

[Al-Rfou & Alain+ 16] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.A. Côté, M. Côté, A. Courville, Y.N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov,

V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J.A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.A. Manzagol, O. Mastropietro, R.T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I.V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S.R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D.J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, Y. Zhang: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, Vol. abs/1605.02688, May 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 *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19, pp. 3159–3166. AAAI Press, Jan. 2019.

[Alkhouli & Bretschner+ 18] T. Alkhouli, G. Bretschner, H. Ney: On The Alignment Problem In Multi-Head Attention-Based Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 177–185, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Anastasopoulos & Chiang 18] A. Anastasopoulos, D. Chiang: Tied Multitask Learning for Neural Speech Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 82–91, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Andreas & Klein 15] J. Andreas, D. Klein: When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 244–249, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[Artetxe & Labaka+ 18] M. Artetxe, G. Labaka, E. Agirre, K. Cho: Unsupervised Neural Machine Translation. In *International Conference on Learning Representations*, pp. 1–12, April 2018.

[Ba & Kiros+ 16] J.L. Ba, J.R. Kiros, G.E. Hinton: Layer Normalization, 2016.

[Bahar & Bieschke+ 21] P. Bahar, T. Bieschke, R. Schlüter, H. Ney: Tight Integrated End-to-End Training for Cascaded Speech Translation. In *IEEE Spoken Language Technology Workshop*, pp. 950–957, Shenzhen, China, Jan. 2021.

[Bahar & Zeyer+ 19] P. Bahar, A. Zeyer, R. Schlüter, H. Ney: On Using SpecAugment for End-to-End Speech Translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, pp. 1–8, Hong Kong, Nov. 2-3 2019. Association for Computational Linguistics.

[Bahdanau & Cho+ 15] D. Bahdanau, K. Cho, Y. Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. In Y. Bengio, Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, pp. 1–15, May 2015.

[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. PAMI-5, No. 2, pp. 179–190, 1983.

[Baker & McCallum 98] L.D. Baker, A.K. McCallum: Distributional Clustering of Words for Text Classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, 96–103, New York, NY, USA, Aug. 1998. Association for Computing Machinery.

[Banar & Daelemans+ 21] N. Banar, W. Daelemans, M. Kestemont: Character-Level Transformer-Based Neural Machine Translation. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, NLPIR 2020, 149–156, New York, NY, USA, Feb. 2021. Association for Computing Machinery.

[Barrault & Biesialska+ 20] L. Barrault, M. Biesialska, O. Bojar, M.R. Costa-jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C.k. Lo, N. Ljubešić, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, M. Zampieri: Findings of the 2020 Conference on Machine Translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pp. 1–55, Online, Nov. 2020. Association for Computational Linguistics.

[Barrault & Bojar+ 19] L. Barrault, O. Bojar, M.R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, M. Zampieri: Findings of the 2019 Conference on Machine Translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pp. 1–61, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Beck & Zhou+ 19] E. Beck, W. Zhou, R. Schlüter, H. Ney: LSTM Language Models for LVCSR in First-Pass Decoding and Lattice-Rescoring, 2019.

[Bellard 23] F. Bellard: ts_zip: Text Compression using Large Language Models. `https://bellard.org/ts_server/ts_zip.html`, 2023. Accessed: 2023-11-26.

[Benegas & Batra+ 23] G. Benegas, S.S. Batra, Y.S. Song: DNA language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, Vol. 120, No. 44, pp. e2311219120, 2023.

[Bengio & Ducharme+ 03] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin: A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, Vol. 3, No. null, pp. 1137–1155, March 2003.

[Bengio & Senecal 03] Y. Bengio, J.S. Senecal: Quick Training of Probabilistic Neural Nets by Importance Sampling. In C.M. Bishop, B.J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Vol. R4 of *Proceedings of Machine Learning Research*, pp. 17–24. PMLR, 03–06 Jan 2003. Reissued by PMLR on 01 April 2021.

[Bengio & Senecal 08] Y. Bengio, J.S. Senecal: Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. *IEEE Transactions on Neural Networks*, Vol. 19, No. 4, pp. 713–722, 2008.

[Bérard & Besacier+ 18] A. Bérard, L. Besacier, A.C. Kocabiyikoglu, O. Pietquin: End-to-End Automatic Speech Translation of Audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6224–6228. IEEE Press, April 2018.

[Bi & Xiong+ 19] T. Bi, H. Xiong, Z. He, H. Wu, H. Wang: Multi-agent Learning for Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 856–865, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[BigScience Workshop 22] BigScience Workshop: BLOOM: A 176B-Parameter Open-Access Multilingual Language Model, 2022.

[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 *Interspeech*, pp. 1443–1447, Dresden, Germany, Sept. 2015.

[Briakou & Cherry+ 23] E. Briakou, C. Cherry, G. Foster: Searching for Needles in a Haystack: On the Role of Incidental Bilingualism in PaLM's Translation Capability, 2023.

[Brix & Bahar+ 20] C. Brix, P. Bahar, H. Ney: Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3909–3915, Online, July 2020. Association for Computational Linguistics.

[Brown & Cocke+ 88] P. Brown, J. Cocke, S.D. Pietra, V.D. Pietra, F. Jelinek, R. Mercer, P. Roossin: A Statistical Approach to Language Translation. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, 71–76, USA, Aug. 1988. Association for Computational Linguistics.

[Brown & Cocke+ 90] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin: A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, 1990.

[Brown & Della Pietra+ 93] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[Brown & Mann+ 20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei: Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin, editors, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877–1901. Curran Associates, Inc., Dec. 2020.

[Buciluǎ & Caruana+ 06] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil: Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, 535–541, New York, NY, USA, Aug. 2006. Association for Computing Machinery.

[Chandrasegaran & Tran+ 22] K. Chandrasegaran, N.T. Tran, Y. Zhao, N.M. Cheung: Revisiting Label Smoothing and Knowledge Distillation Compatibility: What was Missing?, 2022.

[Chelba & Mikolov+ 13] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, T. Robinson: One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. Technical report, Google, pp. 2635–2639, 2013.

[Chen & Beeferman+ 98] S.F. Chen, D. Beeferman, R. Rosenfeld: Evaluation Metrics For Language Models, 1998.

[Chen & Goodman 96] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pp. 310–318, Santa Cruz, California, USA, June 1996. Association for Computational Linguistics.

[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. 359–394, 1999.

[Chen & Li+ 15] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems, 2015.

[Chen & Liu+ 15] X. Chen, X. Liu, M. Gales, P.C. Woodland: Improving the training and evaluation efficiency of recurrent neural network language models. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5401–5405, Aug. 2015.

[Chen & Liu+ 16a] X. Chen, X. Liu, Y. Qian, M.J.F. Gales, P.C. Woodland: CUED-RNNLM — An open-source toolkit for efficient training and evaluation of recurrent neural network language models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6000–6004, March 2016.

[Chen & Liu+ 16b] X. Chen, X. Liu, Y. Wang, M.J.F. Gales, P.C. Woodland: Efficient Training and Evaluation of Recurrent Neural Network Language Models for Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 24, No. 11, pp. 2146–2157, 2016.

[Chen & Si+ 18] P. Chen, S. Si, Y. Li, C. Chelba, C.J. Hsieh: GroupReduce: Block-Wise Low-Rank Approximation for Neural Language Model Shrinking. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 31, pp. 1–11. Curran Associates, Inc., Dec. 2018.

[Cheng & Huang+ 22] Q. Cheng, J. Huang, Y. Duan: Semantically Consistent Data Augmentation for Neural Machine Translation via Conditional Masked Language Model. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 5148–5157, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics.

[Cherry & Moore+ 12] C. Cherry, R.C. Moore, C. Quirk: On Hierarchical Re-ordering and Permutation Parsing for Phrase-based Decoding. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pp. 200–209, Montréal, Canada, June 2012. Association for Computational Linguistics.

[Chiang & Knight+ 09] D. Chiang, K. Knight, W. Wang: 11,001 New Features for Statistical Machine Translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 218–226, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[Chitnis & DeNero 15] R. Chitnis, J. DeNero: Variable-Length Word Encodings for Neural Translation Models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2088–2093, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[Cho & van Merriënboer+ 14a] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio: On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[Cho & van Merriënboer+ 14b] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[Choi & El-Khamy+ 20] Y. Choi, M. El-Khamy, J. Lee: Universal Deep Neural Network Compression. *IEEE Journal of Selected Topics in Signal Processing*, Vol. 14, No. 4, pp. 715–726, 2020.

[Chomsky 57] N. Chomsky: *Syntactic Structures*. De Gruyter Mouton, Berlin, Boston, 1957.

[Chomsky 69] N. Chomsky: Quine's empirical assumptions. In *Words and objections: Essays on the work of WV Quine*, pp. 53–68. Springer, 1969.

[Chorowski & Jaitly 16] J. Chorowski, N. Jaitly: Towards Better Decoding and Language Model Integration in Sequence to Sequence Models. In *Interspeech*, pp. 523–527, Aug. 2016.

[Chung & Cho+ 16] J. Chung, K. Cho, Y. Bengio: A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1693–1703, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Church & Gale 91] K.W. Church, W.A. Gale: A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, Vol. 5, pp. 19–54, 1991.

[CireAan & Meier+ 12] D. CireAan, U. Meier, J. Masci, J. Schmidhuber: Multi-column deep neural network for traffic sign classification. *Neural networks*, Vol. 32, pp. 333–338, 2012.

[Collobert & Weston 08] R. Collobert, J. Weston: A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, 160–167, New York, NY, USA, July 2008. Association for Computing Machinery.

[Corless & Gonnet+ 96] R.M. Corless, G.H. Gonnet, D.E. Hare, D.J. Jeffrey, D.E. Knuth: On the Lambert W function. *Advances in Computational mathematics*, Vol. 5, pp. 329–359, 1996.

[Dai & Yang+ 19] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, R. Salakhutdinov: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[Dauphin & Fan+ 17] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier: Language Modeling with Gated Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 933–941. JMLR.org, Aug. 2017.

[Deng & Hsiao+ 22] L. Deng, R. Hsiao, A. Ghoshal: Bilingual End-to-End ASR with Byte-Level Subwords, 2022.

[Denton & Zaremba+ 14] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus: Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, 1269–1277, Cambridge, MA, USA, Dec. 2014. MIT Press.

[Devlin & Chang+ 19] J. Devlin, M.W. Chang, K. Lee, K. Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[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 *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1370–1380, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[Diao & Zhou+ 23] S. Diao, W. Zhou, X. Zhang, J. Wang: Write and Paint: Generative Vision-Language Models are Unified Modal Learners. In *The Eleventh International Conference on Learning Representations*, pp. 1–25, Feb. 2023.

[Domhan & Hieber 17] T. Domhan, F. Hieber: Using Target-side Monolingual Data for Neural Machine Translation through Multi-task Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1500–1505, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[Dong & Yang+ 19] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, H.W. Hon: Unified Language Model Pre-Training for Natural Language Understanding and Generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 1–13, Red Hook, NY, USA, Dec. 2019. Curran Associates Inc.

[Dosovitskiy & Beyer+ 21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, pp. 1–21, May 2021.

[Dyer 14] C. Dyer: Notes on Noise Contrastive Estimation and Negative Sampling, 2014.

[Edunov & Ott+ 18] S. Edunov, M. Ott, M. Auli, D. Grangier: Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural*

*Language Processing*, pp. 489–500, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[Fairseq 23] Fairseq: Documentation of Incremental Decoding in Fairseq, 2023.

[Flam-Shepherd & Zhu+ 22] D. Flam-Shepherd, K. Zhu, A. Aspuru-Guzik: Language models can learn complex molecular distributions. *Nature Communications*, Vol. 13, No. 1, pp. 3293, 2022.

[Frankle & Carbin 19] J. Frankle, M. Carbin: The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, pp. 1–42. OpenReview.net, Sept. 2019.

[Freitag & Al-Onaizan 17] M. Freitag, Y. Al-Onaizan: Beam Search Strategies for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 56–60, Vancouver, Aug. 2017. Association for Computational Linguistics.

[Freitag & Huck+ 14] M. Freitag, M. Huck, H. Ney: Jane: Open Source Machine Translation System Combination. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 29–32, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[Fu & Lam+ 23] Z. Fu, W. Lam, Q. Yu, A.M.C. So, S. Hu, Z. Liu, N. Collier: Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder, 2023.

[Gale & Sampson 95] W.A. Gale, G. Sampson: Good-turing frequency estimation without tears. *Journal of quantitative linguistics*, Vol. 2, No. 3, pp. 217–237, 1995.

[Galley & Manning 08] M. Galley, C.D. Manning: A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 848–856, Honolulu, Hawaii, Oct. 2008. Association for Computational Linguistics.

[Gao & Herold+ 19] Y. Gao, C. Herold, W. Wang, H. Ney: Exploring Kernel Functions in the Softmax Layer for Contextual Word Classification. In *Proceedings of the 16th International Conference on Spoken Language Translation*, pp. 1–6, Hong Kong, Nov. 2-3 2019. Association for Computational Linguistics.

[Gao & Herold+ 22a] Y. Gao, C. Herold, Z. Yang, H. Ney: Is Encoder-Decoder Redundant for Neural Machine Translation? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 562–574, Online only, Nov. 2022. Association for Computational Linguistics.

[Gao & Herold+ 22b] Y. Gao, C. Herold, Z. Yang, H. Ney: Revisiting Checkpoint Averaging for Neural Machine Translation. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pp. 188–196, Online only, Nov. 2022. Association for Computational Linguistics.

[Gao & Liao+ 20] Y. Gao, B. Liao, H. Ney: Unifying Input and Output Smoothing in Neural Machine Translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 4361–4372, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics.

[Gao & Thulke+ 21] Y. Gao, D. Thulke, A. Gerstenberger, K.V. Tran, R. Schlüter, H. Ney: On Sampling-Based Training Criteria for Neural Language Modeling. In *Interspeech*, pp. 1877–1881, Aug. 2021.

[Gao & Tow+ 21] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, A. Zou: A framework for few-shot language model evaluation, Sept. 2021.

[Gao & Wang+ 19] Y. Gao, W. Wang, H. Ney: uniblock: Scoring and Filtering Corpus with Unicode Block Information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1324–1329, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[Gao & Wang+ 20] Y. Gao, W. Wang, C. Herold, Z. Yang, H. Ney: Towards a Better Understanding of Label Smoothing in Neural Machine Translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 212–223, Suzhou, China, Dec. 2020. Association for Computational Linguistics.

[Gao & Zhu+ 19] F. Gao, J. Zhu, L. Wu, Y. Xia, T. Qin, X. Cheng, W. Zhou, T.Y. Liu: Soft Contextual Data Augmentation for Neural Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5539–5544, Florence, Italy, July 2019. Association for Computational Linguistics.

[Gehring & Auli+ 17] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin: Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1243–1252. JMLR.org, Aug. 2017.

[Gerstenberger & Irie+ 20] A. Gerstenberger, K. Irie, P. Golik, E. Beck, H. Ney: Domain Robust, Fast, and Compact Neural Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 7954–7958, Barcelona, Spain, May 2020.

[Ghazvininejad & Gonen+ 23] M. Ghazvininejad, H. Gonen, L. Zettlemoyer: Dictionary-based Phrase-level Prompting of Large Language Models for Machine Translation, 2023.

[Goldberger & Melamud 18a] J. Goldberger, O. Melamud: Self-Normalization Properties of Language Modeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 764–773, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[Goldberger & Melamud 18b] J. Goldberger, O. Melamud: Self-Normalization Properties of Language Modeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 764–773, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[Golik & Doetsch+ 13] P. Golik, P. Doetsch, H. Ney: Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison. In *Interspeech*, pp. 1756–1760, Lyon, France, Aug. 2013.

[Good 53] I.J. Good: THE POPULATION FREQUENCIES OF SPECIES AND THE ESTIMATION OF POPULATION PARAMETERS. *Biometrika*, Vol. 40, No. 3-4, pp. 237–264, 12 1953.

[Goodfellow & Bengio+ 16] I.J. Goodfellow, Y. Bengio, A. Courville: *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[Goodman 01] J. Goodman: Classes for fast maximum entropy training. *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, Vol. 1, pp. 561–564 vol.1, 2001.

[Graça & Kim+ 19] M. Graça, Y. Kim, J. Schamper, S. Khadivi, H. Ney: Generalizing Back-Translation in Neural Machine Translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pp. 45–52, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Grave & Bojanowski+ 18] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov: Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 3483–3487, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[Green & Wang+ 13] S. Green, S. Wang, D. Cer, C.D. Manning: Fast and Adaptive Online Training of Feature-Rich Translation Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 311–321, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics.

[Gu & Bradbury+ 18] J. Gu, J. Bradbury, C. Xiong, V.O. Li, R. Socher: Non-Autoregressive Neural Machine Translation. In *International Conference on Learning Representations*, pp. 1–13, Feb. 2018.

[Gulcehre & Firat+ 17] C. Gulcehre, O. Firat, K. Xu, K. Cho, Y. Bengio: On integrating a language model into neural machine translation. *Computer Speech & Language*, Vol. 45, pp. 137–148, 2017.

[Guo & Liu+ 20] L. Guo, J. Liu, X. Zhu, P. Yao, S. Lu, H. Lu: Normalized and Geometry-Aware Self-Attention Network for Image Captioning. *CoRR*, Vol. abs/2003.08897, 2020.

[Guta 20] V.A. Guta: *Search and Training with Joint Translation and Reordering Models for Statistical Machine Translation*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, Sept. 2020.

[Gutmann & Hyvärinen 12] M.U. Gutmann, A. Hyvärinen: Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research*, Vol. 13, No. 11, pp. 307–361, 2012.

[Gutmann & Hyvärinen 10] M. Gutmann, A. Hyvärinen: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh, M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Vol. 9 of *Proceedings of Machine Learning Research*, pp. 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[Hakimi Parizi & Cook 18] A. Hakimi Parizi, P. Cook: Do Character-Level Neural Network Language Models Capture Knowledge of Multiword Expression Compositionality? In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 185–192, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[Han & Pool+ 15] S. Han, J. Pool, J. Tran, W.J. Dally: Learning Both Weights and Connections for Efficient Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, 1135–1143, Cambridge, MA, USA, Dec. 2015. MIT Press.

[Hao & Song+ 22] Y. Hao, H. Song, L. Dong, S. Huang, Z. Chi, W. Wang, S. Ma, F. Wei: Language Models are General-Purpose Interfaces, 2022.

[Hassan & Aue+ 18] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, M. Zhou: Achieving Human Parity on Automatic Chinese to English News Translation, 2018.

[Hastie & Tibshirani+ 09] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman: *The elements of statistical learning: data mining, inference, and prediction*, Vol. 2. Springer, 2009.

[Hastings 70] W.K. Hastings: Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, Vol. 57, No. 1, pp. 97–109, 1970.

[Hawthorne & Jaegle+ 22] C. Hawthorne, A. Jaegle, C. Cangea, S. Borgeaud, C. Nash, M. Malinowski, S. Dieleman, O. Vinyals, M. Botvinick, I. Simon, H. Sheahan, N. Zeghidour, J.B. Alayrac, J. Carreira, J. Engel: General-purpose, long-context autoregressive modeling with Perceiver AR, 2022.

[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 S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 31, pp. 1–11. Curran Associates, Inc., Dec. 2018.

[He & Zhang+ 18] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, M. Li: Bag of Tricks for Image Classification with Convolutional Neural Networks, 2018.

[Hendy & Abdelrehim+ 23] A. Hendy, M. Abdelrehim, A. Sharaf, V. Raunak, M. Gabr, H. Matsushita, Y.J. Kim, M. Afify, H.H. Awadalla: How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation, 2023.

[Herold & Gao+ 18] C. Herold, Y. Gao, H. Ney: Improving Neural Language Models with Weight Norm Initialization and Regularization. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 93–100, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Hieber & Domhan+ 18] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, M. Post: Sockeye: A Toolkit for Neural Machine Translation, 2018.

[Hill & Cho+ 16] F. Hill, K. Cho, A. Korhonen: Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics.

[Hinton 09] G.E. Hinton: Deep belief networks. *Scholarpedia*, Vol. 4, No. 5, pp. 5947, 2009.

[Hinton & Osindero+ 06] G.E. Hinton, S. Osindero, Y.W. Teh: A fast learning algorithm for deep belief nets. *Neural computation*, Vol. 18, No. 7, pp. 1527–1554, 2006.

[Hinton & Vinyals+ 15] G. Hinton, O. Vinyals, J. Dean: Distilling the Knowledge in a Neural Network, 2015.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long Short-Term Memory. *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780, nov 1997.

[Hoffmann & Borgeaud+ 22] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D.d.L. Casas, L.A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G.v.d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J.W. Rae, O. Vinyals, L. Sifre: Training Compute-Optimal Large Language Models, 2022.

[Hopkins & May 11] M. Hopkins, J. May: Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[Hubara & Courbariaux+ 17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio: Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *J. Mach. Learn. Res.*, Vol. 18, No. 1, pp. 6869–6898, jan 2017.

[Huffman 52] D.A. Huffman: A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, Vol. 40, No. 9, pp. 1098–1101, 1952.

[Hugging Face H4 23] Hugging Face H4: Open LLM Leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023. Accessed: 2023-11-26.

[Hui & Belkin 21] L. Hui, M. Belkin: Evaluation of Neural Architectures trained with square Loss vs Cross-Entropy in Classification Tasks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, pp. 1–17. OpenReview.net, May 2021.

[Huo & Gao+ 20] J. Huo, Y. Gao, W. Wang, R. Schlüter, H. Ney: Investigation of Large-Margin Softmax in Neural Language Modeling. In *Interspeech*, pp. 1–5, Shanghai, China, Oct. 2020.

[Huo & Herold+ 20] J. Huo, C. Herold, Y. Gao, L. Dahlmann, S. Khadivi, H. Ney: Diving Deep into Context-Aware Neural Machine Translation. In *Proceedings of the Fifth Conference on Machine Translation*, pp. 604–616, Online, Nov. 2020. Association for Computational Linguistics.

[Hwang & Sung 16] K. Hwang, W. Sung: Character-Level Incremental Speech Recognition with Recurrent Neural Networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5335–5339. IEEE Press, March 2016.

[Inaguma & Duh+ 19] H. Inaguma, K. Duh, T. Kawahara, S. Watanabe: Multilingual End-to-End Speech Translation. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 570–577, Dec. 2019.

[Irie 20] K. Irie: *Advancing Neural Language Modeling in Automatic Speech Recognition*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, May 2020.

[Irie & Zeyer+ 19] K. Irie, A. Zeyer, R. Schlüter, H. Ney: Language Modeling with Deep Transformers. In *Interspeech*, pp. 3905–3909, Graz, Austria, Sept. 2019. ISCA Best Student Paper Award.

[Iyyer & Manjunatha+ 15] M. Iyyer, V. Manjunatha, J. Boyd-Graber, H. Daumé III: Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics.

[Jean & Cho+ 15] S. Jean, K. Cho, R. Memisevic, Y. Bengio: On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1–10, Beijing, China, July 2015. Association for Computational Linguistics.

[Jelinek 76] F. Jelinek: Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, Vol. 64, No. 4, pp. 532–556, 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: Perplexity—a 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 & Mercer 80] F. Jelinek, R.L. Mercer: Interpolated estimation of Markov source parameters from sparse data. In E.S. Gelsema, L.N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pp. 381–397. North Holland, Amsterdam, 1980.

[Johnson & Schuster+ 17] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, J. Dean: Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 339–351, 2017.

[Jozefowicz & Vinyals+ 16] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu: Exploring the Limits of Language Modeling, 2016.

[Junczys-Dowmunt & Grundkiewicz+ 18] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A.F. Aji, N. Bogoychev, A.F.T. Martins, A. Birch: Marian: Fast Neural Machine Translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pp. 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Kalchbrenner & Blunsom 13] N. Kalchbrenner, P. Blunsom: Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[Kano & Sakti+ 17] T. Kano, S. Sakti, S. Nakamura: Structured-Based Curriculum Learning for End-to-End English-Japanese Speech Translation. In *Interspeech*, pp. 2630–2634, Aug. 2017.

[Karpathy 23] A. Karpathy: minGPT, 2023.

[Katz 87] S.M. Katz: Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 35, pp. 400–401, 1987.

[Kennedy 16] T. Kennedy: Lecture Notes on Monte Carlo Methods - A Special Topics Course, April 2016.

[Kim & Gao+ 19] Y. Kim, Y. Gao, H. Ney: Effective Cross-lingual Transfer of Neural Machine Translation Models without Shared Vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1246–1257, Florence, Italy, July 2019. Association for Computational Linguistics.

[Kim & Geng+ 18] Y. Kim, J. Geng, H. Ney: Improving Unsupervised Word-by-Word Translation with Language Model and Denoising Autoencoder. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 862–868, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[Kim & Jernite+ 16] Y. Kim, Y. Jernite, D. Sontag, A.M. Rush: Character-Aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2741–2749. AAAI Press, Feb. 2016.

[Kim & Tran+ 19] Y. Kim, D.T. Tran, H. Ney: When and Why is Document-level Context Useful in Neural Machine Translation? In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pp. 24–34, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[Kitza & Golik+ 19] M. Kitza, P. Golik, R. Schlüter, H. Ney: Cumulative Adaptation for BLSTM Acoustic Models. In *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. 1-2, pp. 19–28, May 2002.

[Klein & Kim+ 17] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. Rush: OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*, pp. 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved Backing-off for M-gram Language Modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 181–184, Detroit, Michigan, USA, May 1995.

[Kobayashi 18] S. Kobayashi: Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 452–457, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Kocmi & Bawden+ 22] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, R. Knowles, P. Koehn, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, M. Novák, M. Popel, M. Popović: Findings of the 2022 Conference on Machine Translation (WMT22). In *Proceedings of the Seventh*

*Conference on Machine Translation (WMT)*, pp. 1–45, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics.

[Kocmi & Federmann 23] T. Kocmi, C. Federmann: Large Language Models Are State-of-the-Art Evaluators of Translation Quality, 2023.

[Koehn 05] P. Koehn: Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of Machine Translation Summit X: Papers*, pp. 79–86, Phuket, Thailand, Sept. 13-15 2005.

[Koehn 20] P. Koehn: *Neural machine translation*. Cambridge University Press, 2020.

[Koehn & Hoang+ 07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst: Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Koehn & Knowles 17] P. Koehn, R. Knowles: Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pp. 28–39, Vancouver, Aug. 2017. Association for Computational Linguistics.

[Koehn & Och+ 03] P. Koehn, F.J. Och, D. Marcu: Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 127–133, May 2003.

[Kong & de Masson d'Autume+ 20] L. Kong, C. de Masson d'Autume, L. Yu, W. Ling, Z. Dai, D. Yogatama: A Mutual Information Maximization Perspective of Language Representation Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, pp. 1–12. OpenReview.net, April 2020.

[Kreutzer & Bastings+ 19] J. Kreutzer, J. Bastings, S. Riezler: Joey NMT: A Minimalist NMT Toolkit for Novices. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 109–114, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[Krizhevsky & Sutskever+ 17] A. Krizhevsky, I. Sutskever, G.E. Hinton: Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, Vol. 60, No. 6, pp. 84–90, 2017.

[Krogh & Hertz 91] A. Krogh, J.A. Hertz: A Simple Weight Decay Can Improve Generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, 950–957, San Francisco, CA, USA, Dec. 1991. Morgan Kaufmann Publishers Inc.

[Kudo & Richardson 18] T. Kudo, J. Richardson: SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[Kullback & Leibler 51] S. Kullback, R.A. Leibler: On Information and Sufficiency. *Ann. Math. Statist.*, Vol. 22, No. 1, pp. 79–86, 1951.

[Lample & Conneau+ 18] G. Lample, A. Conneau, L. Denoyer, M. Ranzato: Unsupervised Machine Translation Using Monolingual Corpora Only. In *International Conference on Learning Representations*, pp. 1–14, April 2018.

[LeCun & Bottou+ 12] Y.A. LeCun, L. Bottou, G.B. Orr, K.R. Müller: *Efficient BackProp*, pp. 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Lee & Cho+ 17] J. Lee, K. Cho, T. Hofmann: Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 365–378, 2017.

[Levenshtein et al. 66] V.I. Levenshtein et al.: Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10, pp. 707–710. Soviet Union, Feb. 1966.

[Li & Zhao+ 14] J. Li, R. Zhao, J.T. Huang, Y. Gong: Learning small-size DNN with output-distribution-based criteria. In *Interspeech*, pp. 1–5, Sept. 2014.

[Li & Zhao+ 20] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, S. Chintala: PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *Proc. VLDB Endow.*, Vol. 13, No. 12, pp. 3005–3018, sep 2020.

[Liang & Wang+ 22] B. Liang, P. Wang, Y. Cao: The Implicit Length Bias of Label Smoothing on Beam Search Decoding, 2022.

[Liao & Gao+ 20] B. Liao, Y. Gao, H. Ney: Multi-Agent Mutual Learning at Sentence-Level and Token-Level for Neural Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1715–1724, Online, Nov. 2020. Association for Computational Linguistics.

[Lin & Wu+ 21] Z. Lin, L. Wu, M. Wang, L. Li: Learning Language Specific Sub-network for Multilingual Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 293–305, Online, Aug. 2021. Association for Computational Linguistics.

[Liu & Cao+ 18] X. Liu, D. Cao, K. Yu: Binarized LSTM Language Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2113–2121, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Liu & Saleh+ 18a] P.J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer: Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*, pp. 1–18, April 2018.

[Liu* & Saleh*+ 18b] P.J. Liu*, M. Saleh*, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer: Generating Wikipedia by Summarizing Long Sequences. In *International Conference on Learning Representations*, pp. 1–18, Feb. 2018.

[Liu & Zhou+ 18] Y. Liu, L. Zhou, Y. Wang, Y. Zhao, J. Zhang, C. Zong: A Comparable Study on Model Averaging, Ensembling and Reranking in NMT. In M. Zhang, V. Ng, D. Zhao, S. Li, H. Zan, editors, *Natural Language Processing and Chinese Computing*, pp. 299–308, Cham, Aug. 2018. Springer International Publishing.

[Lukasik & Bhojanapalli+ 20] M. Lukasik, S. Bhojanapalli, A.K. Menon, S. Kumar: Does Label Smoothing Mitigate Label Noise? In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, pp. 1–11. JMLR.org, July 2020.

[Luong & Pham+ 15] T. Luong, H. Pham, C.D. Manning: Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[Lüscher & Beck+ 19a] 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 *Interspeech*, pp. 231–235, Graz, Austria, Sept. 2019.

[Lüscher & Beck+ 19b] 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 *Interspeech*, pp. 231–235, Graz, Austria, Sept. 2019.

[Martinez & Bengio+ 13] H.P. Martinez, Y. Bengio, G.N. Yannakakis: Learning deep physiological models of affect. *IEEE Computational intelligence magazine*, Vol. 8, No. 2, pp. 20–33, 2013.

[MathWorks 23] MathWorks: Symbolic Math Toolbox in Matlab. `https://www.mathworks.com/products/symbolic.html`, 2023. Accessed: 2023-04-28.

[Meng & Li+ 18] Z. Meng, J. Li, Y. Gong, B.H. Juang: Adversarial Teacher-Student Learning for Unsupervised Domain Adaptation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, apr 2018.

[Merity & Keskar+ 18] S. Merity, N.S. Keskar, R. Socher: Regularizing and Optimizing LSTM Language Models. In *International Conference on Learning Representations*, pp. 1–13, April 2018.

[Merity & Xiong+ 17] S. Merity, C. Xiong, J. Bradbury, R. Socher: Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*, pp. 1–15, April 2017.

[Metropolis & Rosenbluth+ 53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller: Equation of state calculations by fast computing machines. *The journal of chemical physics*, Vol. 21, No. 6, pp. 1087–1092, 1953.

[Meyer & Michel+ 22] F. Meyer, W. Michel, M. Zeineldeen, R. Schlüter, H. Ney: Automatic Learning of Subword Dependent Model Scales. In *Interspeech*, pp. 1–5, Incheon, Korea, Sept. 2022.

[Mikolov & Chen+ 13] T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient Estimation of Word Representations in Vector Space. In Y. Bengio, Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, pp. 1–12, May 2013.

[Mikolov & Grave+ 18] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin: Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 52–55, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[Mikolov & Kombrink+ 11a] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, J.H. Cernocky: RNNLM - Recurrent Neural Network Language Modeling Toolkit. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 1–4, December 2011.

[Mikolov & Kombrink+ 11b] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, S. Khudanpur: Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, May 2011.

[Mikolov & Sutskever+ 13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean: Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119, Red Hook, NY, USA, Dec. 2013. Curran Associates Inc.

[Mikolov & Zweig 12] T. Mikolov, G. Zweig: Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pp. 234–239, Dec. 2012.

[Miller 98] G.A. Miller: *WordNet: An electronic lexical database*. MIT press, 1998.

[Mnih & Hinton 08] A. Mnih, G.E. Hinton: A Scalable Hierarchical Distributed Language Model. In D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, editors, *Advances in Neural Information Processing Systems*, Vol. 21, pp. 1–8. Curran Associates, Inc., Dec. 2008.

[Mnih & Teh 12] A. Mnih, Y.W. Teh: A Fast and Simple Algorithm for Training Neural Probabilistic Language Models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 419–426, Madison, WI, USA, June 2012. Omnipress.

[Morin & Bengio 05] F. Morin, Y. Bengio: Hierarchical Probabilistic Neural Network Language Model. In R.G. Cowell, Z. Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Vol. R5 of *Proceedings of Machine Learning Research*, pp. 246–252. PMLR, 06–08 Jan 2005. Reissued by PMLR on 30 March 2021.

[Moslem & Haque+ 23] Y. Moslem, R. Haque, J.D. Kelleher, A. Way: Adaptive Machine Translation with Large Language Models, 2023.

[Müller & Kornblith+ 19] R. Müller, S. Kornblith, G.E. Hinton: When does label smoothing help? In *Advances in Neural Information Processing Systems*, Vol. 32, pp. 1–10. Curran Associates, Inc., Dec. 2019.

[Ney & Essen 91] H. Ney, U. Essen: On smoothing techniques for bigram-based natural language modelling. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pp. 825–828 vol.2, April 1991.

[Nickolls & Buck+ 08] J. Nickolls, I. Buck, M. Garland, K. Skadron: Scalable parallel programming with CUDA. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, New York, NY, USA, 2008. Association for Computing Machinery.

[Niu & Zhong+ 21] Z. Niu, G. Zhong, H. Yu: A review on the attention mechanism of deep learning. *Neurocomputing*, Vol. 452, pp. 48–62, 2021.

[Norvig 12] P. Norvig: Colorless green ideas learn furiously: Chomsky and the two cultures of statistical learning. *Significance*, Vol. 9, No. 4, pp. 30–33, 2012.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 160–167, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[Och & Ney 02] F.J. Och, H. Ney: Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 295–302, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Och & Tillmann+ 99] F.J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 20–28, June 1999.

[Ott & Edunov+ 18] M. Ott, S. Edunov, D. Grangier, M. Auli: Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Ott & Edunov+ 19] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, M. Auli: fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Oualil & Klakow 17] Y. Oualil, D. Klakow: A Batch Noise Contrastive Estimation Approach for Training Large Vocabulary Language Models, 2017.

[Ouyang & Wu+ 22] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C.L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe: Training language models to follow instructions with human feedback, 2022.

[Papineni & Roukos+ 98] K.A. Papineni, S. Roukos, R.T. Ward: Maximum likelihood and discriminative training of direct translation models. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, Vol. 1, pp. 189–192. IEEE, May 1998.

[Papineni & Roukos+ 02] K. Papineni, S. Roukos, T. Ward, W.J. Zhu: Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Pascanu & Mikolov+ 13] R. Pascanu, T. Mikolov, Y. Bengio: On the difficulty of training recurrent neural networks. In S. Dasgupta, D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[Paszke & Gross+ 19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, pp. 1–12. Curran Associates Inc., Red Hook, NY, USA, 2019.

[Pereyra & Tucker+ 17] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, G.E. Hinton: Regularizing Neural Networks by Penalizing Confident Output Distributions. *CoRR*, Vol. abs/1701.06548, 2017.

[Peris & Casacuberta 18] Á. Peris, F. Casacuberta: NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning. *The Prague Bulletin of Mathematical Linguistics*, Vol. 111, No. 1, pp. 113–124, oct 2018.

[Peter & Beck+ 18] J.T. Peter, E. Beck, H. Ney: Sisyphus, a Workflow Manager Designed for Machine Translation and Automatic Speech Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 84–89, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[Poole & Ozair+ 19] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, G. Tucker: On Variational Bounds of Mutual Information. In K. Chaudhuri, R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 09–15 Jun 2019.

[Powers 98] D.M.W. Powers: Applications and Explanations of Zipf's Law. In *New Methods in Language Processing and Computational Natural Language Learning*, pp. 151–160, Jan. 1998.

[Press & Smith 18] O. Press, N.A. Smith: You May Not Need Attention, 2018.

[Press & Wolf 17] O. Press, L. Wolf: Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics.

[Provilkov & Malinin 21] I. Provilkov, A. Malinin: Multi-Sentence Resampling: A Simple Approach to Alleviate Dataset Length Bias and Beam-Search Degradation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 8612–8621, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[Python 23] Python: Documentation of Python f-string, 2023.

[Qin & Specia 15] Y. Qin, L. Specia: Truly Exploring Multiple References for Machine Translation Evaluation. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pp. 113–120, Antalya, Turkey, May 2015.

[Radford & Kim+ 22] A. Radford, J.W. Kim, C.M. Payne, P. Mishkin, T. Xu, G. Brockman, I. Sutskever: Introducing Whisper. `https://openai.com/blog/whisper/`, 2022. [Online; accessed 3-February-2023].

[Radford & Narasimhan 18] A. Radford, K. Narasimhan: Improving Language Understanding by Generative Pre-Training, 2018.

[Radford & Wu+ 19a] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever: Language Models are Unsupervised Multitask Learners, 2019.

[Radford & Wu+ 19b] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever: Language models are unsupervised multitask learners. *OpenAI blog*, Vol. 1, No. 8, pp. 9, 2019.

[Raffel & Shazeer+ 20] 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. *Journal of Machine Learning Research*, Vol. 21, No. 140, pp. 1–67, 2020.

[Raffel & Shazeer+ 22] 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. *J. Mach. Learn. Res.*, Vol. 21, No. 1, jun 2022.

[Rosendahl & Herold+ 19] J. Rosendahl, C. Herold, Y. Kim, M. Graça, W. Wang, P. Bahar, Y. Gao, H. Ney: The RWTH Aachen University Machine Translation Systems for WMT 2019. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pp. 349–355, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[Rosendahl & Tran+ 19] J. Rosendahl, V.A.K. Tran, W. Wang, H. Ney: Analysis of Positional Encodings for Neural Machine Translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, pp. 1–6, Hong Kong, Nov. 2-3 2019. Association for Computational Linguistics.

[Salimans & Goodfellow+ 16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, X. Chen: Improved Techniques for Training GANs. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 29, pp. 1–9. Curran Associates, Inc., Dec. 2016.

[Sanh & Webson+ 22] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M.S. Bari, C. Xu, U. Thakker, S.S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M.T.J. Jiang, H. Wang, M. Manica, S. Shen, Z.X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J.A. Fries, R. Teehan, T.L. Scao, S. Biderman, L. Gao, T. Wolf, A.M. Rush: Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*, pp. 1–216, April 2022.

[Scao & Wang+ 22] T.L. Scao, T. Wang, D. Hesslow, L. Saulnier, S. Bekman, M.S. Bari, S. Biderman, H. Elsahar, N. Muennighoff, J. Phang, O. Press, C. Raffel, V. Sanh, S. Shen, L. Sutawika, J. Tae, Z.X. Yong, J. Launay, I. Beltagy: What Language Model to Train if You Have One Million GPU Hours?, 2022.

[Schaeffer & Miranda+ 23] R. Schaeffer, B. Miranda, S. Koyejo: Are Emergent Abilities of Large Language Models a Mirage?, 2023.

[Schmidhuber 92] J. Schmidhuber: Learning complex, extended sequences using the principle of history compression. *Neural Computation*, Vol. 4, No. 2, pp. 234–242, 1992.

[Schulman & Zoph+ 22] J. Schulman, B. Zoph, C. Kim, J. Hilton, M. Jacob, J. Weng, J.F.C. Uribem, L. Fedus, L. Metz, M. Pokorny, R.G. Lopes, S. Zhao, A. Vijayvergiya, E. Sigler,

A. Perelman, C. Voss, M. Heaton, J. Parish, D. Cummings, R. Nayak, V. Balcom, D. Schnurr, T. Kaftan, C. Hallacy, N. Turley, N. Deutsch, V. Goel, J. Ward, A. Konstantinidis, W. Zaremba, L. Ouyang, L. Bogdonoff, J. Gross, D. Medina, S. Yoo, T. Lee, R. Lowe, D. Mossing, J. Huizinga, R. Jiang, C. Wainwright, D. Almeida, S. Lin, M. Zhang, K. Xiao, K. Slama, S. Bills, A. Gray, J. Leike, J. Pachocki, P. Tillet, S. Jain, G. Brockman, N. Ryder, A. Paino, Q. Yuan, C. Winter, B. Wang, M. Bavarian, I. Babuschkin, S. Sidor, I. Kanitscheider, M. Pavlov, M. Plappert, N. Tezak, H. Jun, W. Zhuk, V. Pong, L. Kaiser, J. Tworek, A. Carr, L. Weng, S. Agarwal, K. Cobbe, V. Kosaraju, A. Power, S. Polu, J. Han, R. Puri, S. Jain, B. Chess, C. Gibson, O. Boiko, E. Parparita, A. Tootoonchian, K. Kosic, C. Hesse: ChatGPT: Optimizing Language Models for Dialogue. `https://openai.com/blog/chatgpt/`, 2022. [Online; accessed 3-February-2023].

[Schwenk & Gauvain 02] H. Schwenk, J.L. Gauvain: Connectionist language modeling for large vocabulary continuous speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. I–765–I–768, May 2002.

[Schwenk & Koehn 08] H. Schwenk, P. Koehn: Large and Diverse Language Models for Statistical Machine Translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, pp. 661–666, Jan. 2008.

[scikit learn 23] scikit learn: Documentation of scikit-learn BayesianGaussianMixture, 2023.

[SciPy 22] SciPy: Documentation of `scipy.optimize.curve_fit`. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`, 2022. Accessed: 2023-03-22.

[Sennrich & Haddow$^+$ 16a] R. Sennrich, B. Haddow, A. Birch: Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Sennrich & Haddow$^+$ 16b] R. Sennrich, B. Haddow, A. Birch: Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Shannon 48] C.E. Shannon: A mathematical theory of communication. *The 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.

[Shaw & Uszkoreit$^+$ 18] P. Shaw, J. Uszkoreit, A. Vaswani: Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Shen & Liu$^+$ 21] Z. Shen, Z. Liu, D. Xu, Z. Chen, K.T. Cheng, M. Savvides: Is Label Smoothing Truly Incompatible with Knowledge Distillation: An Empirical Study. In *International Conference on Learning Representations*, pp. 1–17, May 2021.

[Shi & Xiao+ 20] X. Shi, Y. Xiao, K. Knight: Why Neural Machine Translation Prefers Empty Outputs, 2020.

[Shi & Zhang+ 14a] Y. Shi, W.Q. Zhang, M. Cai, J. Liu: Efficient One-Pass Decoding with NNLM for Speech Recognition. *IEEE Signal Processing Letters*, Vol. 21, No. 4, pp. 377–381, 2014.

[Shi & Zhang+ 14b] Y. Shi, W.Q. Zhang, M. Cai, J. Liu: Variance regularization of RNNLM for speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4893–4897, July 2014.

[Shoeybi & Patwary+ 19] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, B. Catanzaro: Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *CoRR*, Vol. abs/1909.08053, 2019.

[Snoek & Larochelle+ 12] J. Snoek, H. Larochelle, R.P. Adams: Practical Bayesian Optimization of Machine Learning Algorithms, 2012.

[Snover & Dorr+ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pp. 223–231, Cambridge, Massachusetts, USA, Aug. 8-12 2006. Association for Machine Translation in the Americas.

[Srivastava & Hinton+ 14a] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 56, pp. 1929–1958, 2014.

[Srivastava & Hinton+ 14b] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 56, pp. 1929–1958, 2014.

[Stahlberg & Cross+ 18] F. Stahlberg, J. Cross, V. Stoyanov: Simple Fusion: Return of the Language Model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 204–211, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

[Sundermeyer & Ney+ 15a] 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 & Ney+ 15b] 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, pp. 517–529, 2015.

[Sundermeyer & Schlüter+ 12] M. Sundermeyer, R. Schlüter, H. Ney: LSTM Neural Networks for Language Modeling. In *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 *Interspeech*, pp. 2093–2097, Singapore, Sept. 2014.

[Sutskever & Vinyals+ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, Vol. 27, 2014.

[Szegedy & Liu+ 15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich: Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[Szegedy & Vanhoucke+ 15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna: Rethinking the Inception Architecture for Computer Vision. *CoRR*, Vol. abs/1512.00567, 2015.

[Szegedy & Vanhoucke+ 16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna: Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, June 2016.

[Tang & Sennrich+ 19] G. Tang, R. Sennrich, J. Nivre: Understanding Neural Machine Translation by Simplification: The Case of Encoder-free Models. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 1186–1193, Varna, Bulgaria, Sept. 2019. INCOMA Ltd.

[TensorFlow 22] TensorFlow: Documentation of `tf.nn.sampled_softmax_loss`. `https://www.tensorflow.org/api_docs/python/tf/nn/sampled_softmax_loss`, 2022. Accessed: 2023-02-22.

[Tiedemann 12] J. Tiedemann: Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pp. 2214–2218, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[Tiedemann & Nygaard 04] J. Tiedemann, L. Nygaard: The OPUS Corpus - Parallel and Free: `http://logos.uio.no/opus`. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, pp. 2214–2218, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[Tillmann 04] C. Tillmann: A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pp. 101–104, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[Tran & Thulke+ 22] V.A.K. Tran, D. Thulke, Y. Gao, C. Herold, H. Ney: Does Joint Training Really Help Cascaded Speech Translation? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4480–4487, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[Tschannen & Djolonga+ 20] M. Tschannen, J. Djolonga, P.K. Rubenstein, S. Gelly, M. Lucic: On Mutual Information Maximization for Representation Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, pp. 1–15. OpenReview.net, April 2020.

[Tu & Liu+ 17] Z. Tu, Y. Liu, L. Shang, X. Liu, H. Li: Neural machine translation with reconstruction. In *Thirty-first AAAI conference on artificial intelligence*, pp. 3097–3103, Feb. 2017.

[Unicode 23] Unicode: Unicode Standard, 2023.

[van den Oord & Dieleman+ 16] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu: WaveNet: A Generative Model for Raw Audio. In *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 125, Sept. 2016.

[van den Oord & Li+ 18] A. van den Oord, Y. Li, O. Vinyals: Representation Learning with Contrastive Predictive Coding. *CoRR*, Vol. abs/1807.03748, March 2018.

[Vaswani & Bengio+ 18] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, J. Uszkoreit: Tensor2Tensor for Neural Machine Translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pp. 193–199, Boston, MA, March 2018. Association for Machine Translation in the Americas.

[Vaswani & Shazeer+ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, I. Polosukhin: Attention is All you Need. In I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 30, pp. 1–11. Curran Associates, Inc., Dec. 2017.

[Vogel & Ney+ 96] S. Vogel, H. Ney, C. Tillmann: HMM-Based Word Alignment in Statistical Translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*, pp. 836–841, Aug. 1996.

[Voita & Talbot+ 19] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, I. Titov: Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[Wang 23] W. Wang: *Neural Hidden Markov Model for Machine Translation*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, March 2023.

[Wang & Alkhouli+ 17] W. Wang, T. Alkhouli, D. Zhu, H. Ney: Hybrid neural network alignment and lexicon model in direct hmm for statistical machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 125–131, July 2017.

[Wang & Mohamed+ 19] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, M.L. Seltzer: Transformer-Based Acoustic Modeling for Hybrid Speech Recognition. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. , pp. 6874–6878, 2019.

[Wang & Mohamed+ 20] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, M.L. Seltzer: Transformer-Based Acoustic Modeling for Hybrid Speech Recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6874–6878, May 2020.

[Wang & Roberts+ 22] T. Wang, A. Roberts, D. Hesslow, T.L. Scao, H.W. Chung, I. Beltagy, J. Launay, C. Raffel: What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?, 2022.

[Wang & Tu+ 21] S. Wang, Z. Tu, Z. Tan, W. Wang, M. Sun, Y. Liu: Language Models are Good Translators, 2021.

[Wang & Wu+ 20] C. Wang, Y. Wu, S. Liu, Z. Yang, M. Zhou: Bridging the Gap between Pre-Training and Fine-Tuning for End-to-End Speech Translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 9161–9168. AAAI Press, Feb. 2020.

[Wang & Yan+ 21] F. Wang, J. Yan, F. Meng, J. Zhou: Selective Knowledge Distillation for Neural Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6456–6466, Online, Aug. 2021. Association for Computational Linguistics.

[Wang & Yang+ 21a] W. Wang, Z. Yang, Y. Gao, H. Ney: Transformer-Based Direct Hidden Markov Model for Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 23–32, Aug. 2021.

[Wang & Yang+ 21b] W. Wang, Z. Yang, Y. Gao, H. Ney: Transformer-Based Direct Hidden Markov Model for Machine Translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 23–32, Online, Aug. 2021. Association for Computational Linguistics.

[Wang & Zhu+ 18] W. Wang, D. Zhu, T. Alkhouli, Z. Gan, H. Ney: Neural Hidden Markov Model for Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 377–382, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Wei & Tay+ 22] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E.H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus: Emergent Abilities of Large Language Models, 2022.

[Werbos 74] P. Werbos: *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Ph.D. thesis, Harvard University, 1974.

[Wettig & Gao+ 23] A. Wettig, T. Gao, Z. Zhong, D. Chen: Should You Mask 15% in Masked Language Modeling?, 2023.

[Wolf & Debut+ 20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T.L. Scao, S. Gugger, M. Drame, Q. Lhoest, A.M. Rush: HuggingFace's Transformers: State-of-the-art Natural Language Processing, 2020.

[Wu & Schuster+ 16] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes,

J. Dean: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016.

[Wuebker & Huck+ 12] J. Wuebker, M. Huck, S. Peitz, M. Nuhn, M. Freitag, J.T. Peter, S. Mansour, H. Ney: Jane 2: Open Source Phrase-based and Hierarchical Statistical Machine Translation. In *Proceedings of COLING 2012: Demonstration Papers*, pp. 483–492, Mumbai, India, Dec. 2012. The COLING 2012 Organizing Committee.

[Wübker 17] J. Wübker: *Effective Training and Effcient Decoding for Statistical Machine Translation*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, Feb. 2017.

[Xie & Wang+ 16] L. Xie, J. Wang, Z. Wei, M. Wang, Q. Tian: DisturbLabel: Regularizing CNN on the Loss Layer. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4753–4762. IEEE Computer Society, June 2016.

[Xie & Wang+ 17] Z. Xie, S.I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, A.Y. Ng: Data Noising as Smoothing in Neural Network Language Models. In *International Conference on Learning Representations*, pp. 1–12, April 2017.

[Xu & Rudnicky 00] W. Xu, A. Rudnicky: Can artificial neural networks learn language models? In *Interspeech*, pp. 1–4, Oct. 2000.

[Xu & Wang+ 18] Y. Xu, Y. Wang, A. Zhou, W. Lin, H. Xiong: Deep Neural Network Compression with Single and Multiple Level Quantization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, pp. 4335—-4342. AAAI Press, Feb. 2018.

[Xu & Xu+ 20] Y. Xu, Y. Xu, Q. Qian, H. Li, R. Jin: Towards Understanding Label Smoothing, 2020.

[Yang & Gao+ 20] Z. Yang, Y. Gao, W. Wang, H. Ney: Predicting and Using Target Length in Neural Machine Translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 389–395, Suzhou, China, Dec. 2020. Association for Computational Linguistics.

[Yang & Gao+ 22] Z. Yang, Y. Gao, A. Gerstenberger, J. Jiang, R. Schlüter, H. Ney: Self-Normalized Importance Sampling for Neural Language Modeling. In *Interspeech*, pp. 3909–3913, Incheon, Korea, Sept. 2022.

[Yang & Huang+ 18] Y. Yang, L. Huang, M. Ma: Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3054–3059, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[Yao & Jiang+ 23] B. Yao, M. Jiang, D. Yang, J. Hu: Empowering LLM-based Machine Translation with Cultural Awareness, 2023.

[Yu & Liu+ 17] X. Yu, T. Liu, X. Wang, D. Tao: On Compressing Deep Models by Low Rank and Sparse Decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 67–76, Nov. 2017.

[Zens & Och+ 02] R. Zens, F.J. Och, H. Ney: Phrase-Based Statistical Machine Translation. In M. Jarke, G. Lakemeyer, J. Koehler, editors, *KI 2002: Advances in Artificial Intelligence*, pp. 18–32, Berlin, Heidelberg, Jan. 2002. Springer Berlin Heidelberg.

[Zeyer & Bahar+ 19] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, H. Ney: A comparison of Transformer and LSTM encoder decoder models for ASR. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 8–15, Sentosa, Singapore, Dec. 2019.

[Zhang & Ghorbani+ 22] B. Zhang, B. Ghorbani, A. Bapna, Y. Cheng, X. Garcia, J. Shen, O. Firat: Examining Scaling and Transfer of Language Model Architectures for Machine Translation, 2022.

[Zhang & Haddow+ 23] B. Zhang, B. Haddow, A. Birch: Prompting Large Language Model for Machine Translation: A Case Study, 2023.

[Zhang & Jiang+ 21] C.B. Zhang, P.T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, M.M. Cheng: Delving Deep Into Label Smoothing. *IEEE Transactions on Image Processing*, Vol. 30, pp. 5984–5996, 2021.

[Zhang & Roller+ 22] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X.V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P.S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer: OPT: Open Pre-trained Transformer Language Models, 2022.

[Zhang & Xiang+ 18] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu: Deep Mutual Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4320–4328, June 2018.

[Zhou & Cai+ 18] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, Y. Yu: Activation Maximization Generative Adversarial Nets. In *International Conference on Learning Representations*, pp. 1–24, April 2018.

[Zhou & Michel+ 22] W. Zhou, W. Michel, R. Schlüter, H. Ney: Efficient Training of Neural Transducer for Speech Recognition. In *Interspeech*, pp. 2058–2062, Incheon, Korea, Sept. 2022.

[Zhou & Zeineldeen+ 21] W. Zhou, M. Zeineldeen, Z. Zheng, R. Schlüter, H. Ney: Acoustic Data-Driven Subword Modeling for End-to-End Speech Recognition. In *Interspeech*, pp. 2886–2890, Aug. 2021.

[Zipf 49] G.K. Zipf: *Human behavior and the principle of least effort.* Addison-Wesley Press, 1949.

[Zoph & Vaswani+ 16] B. Zoph, A. Vaswani, J. May, K. Knight: Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1217–1222, San Diego, California, June 2016. Association for Computational Linguistics.