End-to-End Neural Speech Translation

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Matthias Sperber
aus Karlsruhe

Tag der mündlichen Prüfung: 9.1.2019

Erster Gutachter: Prof. Dr. Alexander Waibel

Zweiter Gutachter: Prof. Satoshi Nakamura

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe, sowie dass ich die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des KIT, ehem. Universität Karlsruhe (TH), zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 14.11.2018

Matthias Sperber

Abstract

The goal of this thesis is to develop improved methods for the task of speech translation. Speech translation takes audio signals of speech as input and produces text translations as output, with diverse applications that reach from dialog-based translation in limited domains to automatic translation of academic lectures. A main challenge and reason why current speech translation systems often produce unsatisfactory results is the division of the task into independent recognition and translation steps, leading to the well-known *error propagation* problem. In this thesis, we use recent neural modeling techniques to tighten the speech translation pipeline and, in the extreme, replace the cascade by models that directly generate translations from speech without an intermediate step. The overarching goal of all proposed approaches is the reduction or prevention of the error propagation problem.

As a starting point, we analyze the state-of-the-art for attentional speech recognition models, and develop a new model with a self-attentional acoustic model component. We show that this reduces training time significantly and enables inspection and linguistic interpretation of models that have previously often been described as black box approaches. Equipped with such a model, we then turn to the problem of speech translation, first from the viewpoint of a cascade, where we wish to translate the output of a speech recognizer as accurately as possible. This requires dealing with errors from the speech recognizer, which we achieve by noising the training data, leading to improved robustness of the translation component. A second approach is to explicitly handle competing recognition hypotheses during translation, which we accomplish through a novel lattice-to-sequence model, achieving substantial improvements in translation accuracy.

Direct models for speech translation are a promising new approach that does not divide the process into two independent steps. This requires direct data in which spoken utterances are paired with a textual translation, which is different from cascaded models in which both model components are trained on independent speech recognition and

machine translation corpora. Sequence-to-sequence models can now be trained on such direct data, and have in fact been used along these lines by some research groups, although with only inconsistent reported improvements. In this thesis, we show that whether such models outperform traditional models critically depends on the amount of available direct data, an observation that can be explained by the more complex direct mapping between source speech inputs and target text outputs. This puts direct models at a disadvantage in practice, because we usually possess only limited amounts of direct speech translation data, but have access to much more abundant data for cascaded training. A straight-forward potential solution for incorporating all available data is multi-task training, but we show that this is ineffective and not able to overcome the data disadvantage of the direct model. As a remedy, we develop a new two-stage model that naturally decomposes into two modeling steps akin to the cascade, but is end-to-end trainable and reduces the error propagation problem. We show that this model outperforms all other approaches under ideal data conditions, and is also much more effective at exploiting auxiliary data, closing the gap to the cascaded model in realistic situations. This shows for the first time that end-to-end trainable speech translation models are a practically relevant solution to speech translation, and are able to compete with or outperform traditional approaches.

Zusammenfassung

Diese Arbeit beschäftigt sich mit Methoden zur Verbesserung der automatischen Übersetzung gesprochener Sprache (kurz: Speech Translation). Die Eingabe ist hierbei ein akustisches Signal, die Ausgabe ist der zugehörige Text in einer anderen Sprache. Die Anwendungen sind vielfältig und reichen u.a. von dialogbasierten Übersetzungssystemen in begrenzten Domänen bis hin zu vollautomatischen Vorlesungsübersetzungssystemen.

Speech Translation ist ein komplexer Vorgang der in der Praxis noch viele Fehler produziert. Ein Grund hierfür ist die Zweiteilung in Spracherkennungskomponente und Übersetzungskomponente: beide Komponenten produzieren für sich genommen eine gewisse Menge an Fehlern, zusätzlich werden die Fehler der ersten Komponente an die zweite Komponente weitergereicht (sog. Error Propagation) was zusätzliche Fehler in der Ausgabe verursacht. Die Vermeidung des Error Propagation Problems ist daher grundlegender Forschungsgegenstand im Speech Translation Bereich.

In der Vergangenheit wurden bereits Methoden entwickelt, welche die Schnittstelle zwischen Spracherkenner und Übersetzer verbessern sollen, etwa durch Weiterreichen mehrerer Erkennungshypothesen oder durch Kombination beider Modelle mittels Finite State Transducers. Diese basieren jedoch weitgehend auf veralteten, statistischen Übersetzungsverfahren, die mittlerweile fast vollständig durch komplett neuronale Sequence-to-Sequence Modelle ersetzt wurden.

Die vorliegende Dissertation betrachtet mehrere Ansätze zur Verbesserung von Speech Translation, alle motiviert durch das Ziel, Error Propagation zu vermeiden, sowie durch die Herausforderungen und Möglichkeiten der neuen komplett neuronalen Modelle zur Spracherkennung und Übersetzung. Hierbei werden wir zum Teil völlig neuartige Modelle entwickeln und zum Teil Strategien entwickeln um erfolgreiche klassische Ideen auf neuronale Modelle zu übertragen.

Wir betrachten zunächst eine einfachere Variante unseres Problems, die Spracherkennung. Um Speech Translation Modelle zu entwickeln die komplett

auf neuronalen Sequence-to-Sequence Modellen basieren, müssen wir zunächst sicherstellen dass wir dieses einfachere Problem zufriedenstellend mit ähnlichen Modellen lösen können. Dazu entwickeln wir zunächst ein komplett neuronales Baseline Spracherkennungs-System auf Grundlage von Ergebnissen aus der Literatur, welches wir anschließend durch eine neuartige Self-Attentional Architektur erweitern. Wir zeigen dass wir hiermit sowohl die Trainingszeit verkürzen können, als auch bessere Einblicke in die oft als Blackbox beschriebenen Netze gewinnen und diese aus linguistischer Sicht interpretieren können.

Als nächstes widmen wir uns dem kaskadierten Ansatz zur Speech Translation. Hier nehmen wir an, dass eine Ausgabe eines Spracherkenners gegeben ist, und wir diese so akkurat wie möglich übersetzen wollen. Dazu ist es nötig, mit den Fehlern des Spracherkenners umzugehen, was wir erstens durch verbesserte Robustheit des Übersetzers und zweitens durch Betrachten alternativer Erkennungshypothesen erreichen. Die Verbesserung der Robustheit der Übersetzungskomponente, unser erster Beitrag, erreichen wir durch das Verrauschen der Trainings-Eingaben, wodurch das Modell lernt, mit fehlerhaften Eingaben und insbesondere Spracherkennungsfehlern besser umzugehen. Zweitens entwickeln wir ein Lattice-to-Sequence Übersetzungsmodell, also ein Modell welches Wortgraphen als Eingaben erwartet und diese in eine übersetzte Wortsequenz überführt. Dies ermöglicht uns, einen Teil des Hypothesenraums des Spracherkenners, in Form eines eben solchen Wortgraphen, an den Spracherkenner weiterzureichen. Hierdurch hat die Übersetzungskomponente Zugriff auf verschiedene alternative Ausgaben des Spracherkenners und kann im Training lernen, daraus selbständig die zum Übersetzen optimale und weniger fehlerbehaftete Eingabe zu extrahieren.

Schließlich kommen wir zum finalen und wichtigsten Beitrag dieser Dissertation. Ein vielversprechender neuer Speech Translation Ansatz ist die direkte Modellierung, d.h. ohne explizite Erzeugung eines Transkripts in der Quellsprache als Zwischenschritt. Hierzu sind direkte Daten, d.h. Tonaufnahmen mit zugehörigen textuellen Übersetzungen nötig, im Unterschied zu kaskadierten Modellen, welche auf transkribierte Tonaufnahmen sowie davon unabhängigen parallelen übersetzten Texten trainiert werden. Erstmals bieten die neuen end-to-end trainierbaren Sequence-to-Sequence Modelle grundsätzlich die Möglichkeit dieses direkten Weges und wurden auch bereits von einigen Forschungsgruppen entsprechend getestet, jedoch sind die Ergebnisse teils widersprüchlich und es bleibt bisher unklar, ob man Verbesserungen gegenüber kaskadierten Systemen erwarten kann. Wir zeigen hier dass dies entscheidend von der Menge der verfügbaren Daten abhängt, was sich leicht dadurch erklären lässt

dass direkte Modellierung ein deutlich komplexeres Problem darstellt als der Weg über zwei Schritte. Solche Situationen bedeuten im Maschinellen Lernen oftmals dass mehr Daten benötigt werden. Dies führt uns zu einem fundamentalen Problem dieses ansonsten sehr vielversprechenden Ansatzes, nämlich dass mehr direkte Trainingsdaten benötigt werden, obwohl diese in der Praxis sehr viel schwieriger zu sammeln sind als Trainingsdaten für traditionelle Systeme. Als Ausweg testen wir zunächst eine naheliegende Strategie, weitere traditionelle Daten ins direkte Modell-Training zu integrieren: Multi-Task Training. Dies stellt sich in unseren Experimenten allerdings als unzureichend heraus. Wir entwickeln daher ein neues Modell, das ähnlich einer Kaskade auf zwei Modellierungsschritten basiert, jedoch komplett durch Backpropagation trainiert wird und dabei bei der Übersetzung nur auf Audio-Kontextvektoren zurückgreift und damit nicht durch Erkennungsfehler beeinträchtigt wird. Wir zeigen dass dieses Modell erstens unter idealen Datenkonditionen bessere Ergebnisse gegenüber vergleichbaren direkten und kaskadierten Modellen erzielt, und zweitens deutlich mehr von zusätzlichen traditionellen Daten profitiert als die einfacheren direkten Modelle. Wir zeigen damit erstmals, dass end-to-end trainierbare Speech Translation Modelle eine ernst zu nehmende und praktisch relevante Alternative für traditionelle Ansätze sind.

Acknowledgements

First and foremost I would like to thank Prof. Alex Waibel for his guidance throughout my years as a PhD student, and for his research vision and ideas that made this thesis possible. I am also grateful for being given the opportunity to spend my early PhD years at NAIST in Japan. I am indebted to my co-advisor Prof. Satoshi Nakamura for much support and valuable advice, both during my stays at NAIST and afterwards. I would also like to express gratitude for countless discussions and practical help to Jan Niehues and Sebastian Stüker at KIT, and for all the support from Graham Neubig and Sakriani Sakti at both NAIST and CMU. Special thanks go to my colleagues from Karlsruhe: Eunah Cho, Stefan Constantin, Christian Fügen, Thanh-Le Ha, Michael Heck, Teresa Herrmann, Thilo Köhler, Narine Kokhlikyan, Mohammed Mediani, Markus Müller, Thai Son Nguyen, Quan Ngoc Pham, Kay Rottmann, Maria Schmidt, Mirjam Simantzik, Thomas Zenkel, Yuqi Zhang, as well as to much-needed support from Silke Dannenmaier, Sarah Fünfer, Bastian Krüger, Patricia Lichtblau, Margit Rödder, Virginia Roth, Franziska Vogel, Angela Grimminger, Tuna Murat Cicek, and Micha Wetzel. I would also like to thank Elizabeth Salesky, Zhong Zhou, Austin Matthews, and Paul Michel from Pittsburgh, as well as Philip Arthur, Oliver Adams, Nurul Lubis, Patrick Lumban Tobing, Hiroaki Shimizu, Manami Matsuda, Hiroki Tanaka, Takatomo Kano, and Do Quoc Truong from Nara for their friendship and support.

Last and certainly not least, I would like to thank my wife as well as my family for the unceasing encouragement throughout my years as a PhD student.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of this thesis is to enable machines to accurately translate speech by exploiting recent advances in neural computing. Speech is foundational to human communication: unlike written language, it is acquired naturally during childhood, does not require a writing system, and is readily accessible without any tools beyond the human body. Written language, in contrast, may not be accessible to every person, in every language, and in every circumstance. Being able to automatically *translate* speech would therefore be of tremendous usefulness toward an inclusive globalized world. Potential use cases include the following:

- Information access. For example, accessing the recorded speech of a speaker whose language one does not understand [CFH+13].
- Interpersonal communication. For example, enabling face-to-face dialog between speakers of different languages [WBW07].
- Information dissemination. For example, enabling speakers of a minority language without writing system to make their ideas known to others.

The task of speech translation, as considered in this thesis, takes an audio signal as input and outputs a translated written text. It combines a number of challenging aspects that all need to be addressed simultaneously, making the whole even more challenging than the sum of its parts:

- An acoustic signal is a continuous signal with a high amount of variability that needs to be abstracted from, including speaker characteristics, channel properties, dialects, and background noise.

- The translation of a sentence into another language involves challenges such as reordering of words and phrases, word sense disambiguation, pronoun resolution, and producing syntactically correct and semantically adequate outputs.

- Spoken language[1] tends to contain disfluencies, errors, casual style, and implicit communication, while written language is more formal, grammatically correct, and explicit. Speech translation involves moving from the spoken domain into the written domain and must therefore bridge this gap.

The ambiguity of language and difficulty to describe language using a formal system of rules, along with the aforementioned challenges, has led to a consensus in the natural language processing (NLP) community on data-driven approaches and statistical modeling techniques as preferable in most situations. This enables development of language-independent techniques, automated training, combination of several knowledge sources, and explicit treatment of uncertainty.

Traditionally, speech translation has been implemented through cascades of several statistical models, including a speech recognition component, a segmentation component, and a translation component. To optimize the accuracy of such a cascade, it is important to both improve the accuracy of the individual components and to tightly integrate all components across the cascade. An inherent defect to the traditional cascading approach is the **propagation of errors**. Because of the high degree of ambiguity in natural language, every involved component produces a certain number of errors, which are then propagated through the cascade and lead to compounding follow-up errors. The cascade violates an important principle in statistical modeling, according to which any hard decisions are to be delayed as long as possible [Hun87, MD14].

Attentional encoder-decoder models have recently emerged as a powerful model to flexibly transduce a given sequence into another sequence [KB13, SVL14, BCB15]. These models consist of recurrent encoder and decoder sub-components and an attention sub-component, which can intuitively be thought of as source-side language model, target-side language model, and alignment model. However, a main reason for their success has been the ability to train these sub-components jointly, thereby somewhat blurring this clear-cut division of labor. Attentional encoder-decoder models have been found flexible enough to handle a variety of NLP tasks, including machine

[1] We shall henceforth write *speech* to denote audio signals containing spoken utterances, *spoken language* to denote a verbatim textual representation of spoken utterances, and *written language* as the contrasting case of text originally composed as text without a spoken utterance as immediate basis.

translation [BCB15], speech recognition [CBS+15, CJLV16], constituency parsing [VKK+15], and punctuation insertion [CNW17], with high accuracy.

This paradigm shift leads us to reconsider the traditional cascading approach to spoken language translation in this thesis and to develop new approaches, with the main motivation of preventing the error propagation problem. New approaches are important both because attentional encoder-decoder models offer exciting new modeling opportunities, and because some of the traditional techniques are no longer applicable. In this thesis, we propose improvements to modern spoken language translation along the dimensions of improving individual components, improving integration across the cascade, and replacing the cascade by one-model approaches. The latter idea of performing speech translation with only a single model is particularly appealing because it has the potential to eliminate the problem of propagation of errors altogether and because all model parameters can be estimated jointly. However, it also introduces new challenges because combining several transformation steps into a single component increases the transformation complexity between input sequences and output sequences, and may therefore make models harder to train or require more training data.

1.1 Contributions

To tackle the problem of speech translation, we start with a simpler problem, speech recognition: In order to develop neural speech translation models, we must first ensure to be able to solve this simpler task with an all-neural model. To this end, we establish a state-of-the-art **all-neural baseline speech recognition system** based on prior methods from the literature (Section 4.1), which we then extend through a novel **self-attentional architecture** (Section 4.2), improving both its training speed and interpretability.

We next turn to developing tightly integrated cascaded speech translation models using neural approaches. To this end, we assume to be given the output of a speech recognizer, which we desire to translate as accurately as possible. Our first contributed method (Section 5.1) improves **noise robustness** of the subsequent translation model. In particular, we induce noise to the training data so that the translation model learns to handle the speech recognition errors gracefully. As our next contribution (Section 5.2), we devise a **lattice-to-sequence** translation model that is able to directly consume the decoding graph of the speech recognizer, passing on information on recognition uncertainty through the hidden states and thereby avoiding some of the early decisions normally taken in cascaded approaches. We empirically demonstrate the effectiveness

of the proposed methods.

Our final contribution is a one-model approach where all parameters are trained jointly (Chapter 6). As a first attempt, we extend our all-neural speech recognition model such that it is able to output translations instead of transcripts. We further devise **multi-task training strategies** to improve the model, as well as to exploit auxiliary data such as speech recognition and machine translation data. However, we find that results are mixed when compared to a cascaded model, in line with similar experiments in related works. Worse, the gap to the cascaded model only grows when adding auxiliary speech recognition or machine translation data, putting such models at a severe disadvantage in practical situations. We therefore introduce a novel **attention-passing model** that eliminates the direct model's weaknesses by using two attention mechanisms, while still supporting efficient training via back-propagation and avoiding early decoding decisions. An empirical evaluation shows that this substantially outperforms the cascaded and direct approaches and a previously used two-stage model in favorable data conditions, and is moreover able to exploit auxiliary speech recognition and machine translation data much more effectively than the direct one-model approach. This shows for the first time that end-to-end trainable speech translation models are practically relevant and able to compete with and outperform traditional approaches.

We have also published many of our findings in conference and journal papers [SNNW17, SNW17, PSS+17, SNN+18, ZSN+18, SPN+18, SNNW19].

Chapter 2

Background

The traditional approach to speech translation uses a cascade of an automatic speech recognition component, a machine translation component, and some "speech translation glue" to make both compatible. Naturally, the cascade can benefit from advancements within the individual components. For instance, the recent paradigm shift in machine translation from statistical machine translation (SMT) to neural machine translation (NMT) has improved not only translation of text but also the overall quality of a speech translation cascade in which the SMT component is replaced by NMT.

This chapter establishes the background for this thesis. It describes a speech translation cascade that is traditional in its overall approach of chaining independent components, but does consider state-of-the-art modeling techniques for each component. We start by describing a machine translation system using a modern attentional encoder-decoder architecture. Next, two approaches to solving the speech recognition stage are described: The HMM-based approach that has been the dominant approach for many years, and an attentional encoder-decoder variant that is one of several methods that have recently started to approach the accuracy of HMM systems. All of these are used and extended in later chapters of this thesis. We therefore describe each model to the level of detail necessary for later explorations. We continue on to describing several basic challenges that need to be considered in cascaded systems, and conclude with an overview of prior research on the specific topic of speech translation.

2.1 Neural Machine Translation

Machine translation is the task of translating a given sentence in the source language into a new sentence in the target language. Source and target sentences can be thought

of as sequences of discrete word tokens, but can also be broken down into smaller sub-word units or even character sequences. Neural machine translation imposes a conditional probability distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ for input sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ and output sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_M)$. The model parameters θ are directly estimated in a supervised fashion using a parallel corpus of paired sentences in the source and target language.

The most successful approach to machine translation is currently the *attentional encoder-decoder model* [KB13, SVL14, BCB15], which we will sometimes refer to simply as *encoder-decoder model* for brevity. These models are also called sequence-to-sequence models in literature. We describe this approach in detail because subsequent chapters will introduce extensions to it. Figure 2.1 illustrates the model architecture.

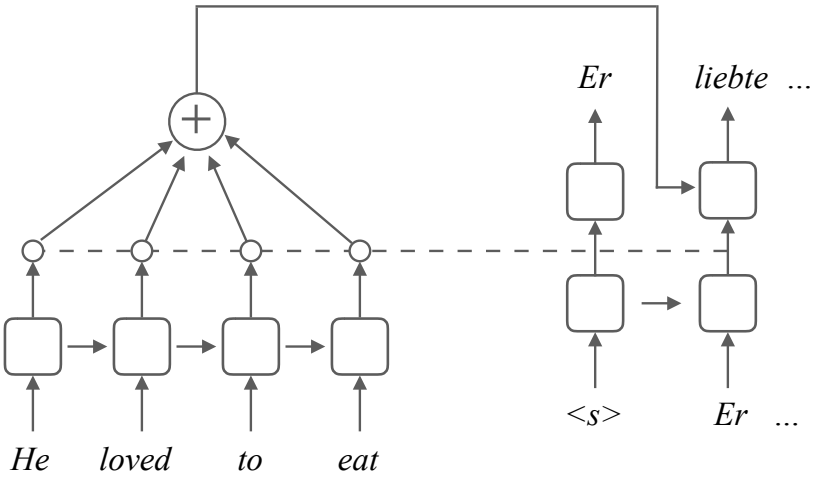


Figure 2.1: Encoder-decoder model architecture.

We begin by factorizing the conditional probability as the product of conditional probabilities of each token to be generated:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{M} p(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}).$$

The training objective is to estimate parameters θ that maximize the log-likelihood of the sentence pairs in a given parallel training data set D:

$$J(\theta) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \log p(\boldsymbol{y} \mid \boldsymbol{x}; \theta).$$

2.1.1 Encoder

The encoder is a bi-directional recurrent neural network (RNN), following Bahdanau et al. [BCB15]. Here, the source sentence is processed in both the forward and backward directions with two separate RNNs. For every input x_i, two hidden states are generated as

$$\overrightarrow{\mathbf{h}}_i = \text{LSTM}\left(E_{fwd}(x_i), \overrightarrow{\mathbf{h}}_{i-1}\right) \tag{2.1}$$

$$\overleftarrow{\mathbf{h}}_i = \text{LSTM}\left(E_{bwd}(x_i), \overleftarrow{\mathbf{h}}_{i+1}\right), \tag{2.2}$$

where E_{fwd} and E_{bwd} are source embedding lookup tables. A typical choice of RNN is long short-term memory (LSTM) [HS97, Section 2.1.4] which has been demonstrated to achieve high accuracy in many situations. Multiple LSTM layers can be stacked, with bidirectional outputs concatenated either after every layer or only after the final layer. We obtain the final hidden encoder state $\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \mid \overleftarrow{\mathbf{h}}_i$, where layer indices are omitted for simplicity.

2.1.2 Attention

We use an attention mechanism [LPM15] to summarize the encoder outputs into a fixed-size representation. At each decoding time step j, a context vector $\mathbf{c}_j$ is computed as a weighted average of the source hidden states:

$$\mathbf{c}_j = \sum_{i=1}^{N} \alpha_{ij} \mathbf{h}_i.$$

The normalized attentional weights α_{ij} measure the relative importance of the source words for the current decoding step and are computed as a softmax with normalization factor Z summing over i:

$$\alpha_{ij} = \frac{1}{Z} \exp\left(\text{s}\left(\mathbf{s}_{j-1}, \mathbf{h}_i\right)\right)$$

$\text{s}(\cdot)$ is a feed-forward neural network with a single layer that estimates the importance of source hidden state $\mathbf{h}_i$ for producing the next target symbol y_j, conditioned on the previous decoder state $\mathbf{s}_{j-1}$.

2.1.3 Decoder

The decoder generates output symbols one by one, conditioned on the encoder states via the attention mechanism. It uses another LSTM, initialized using the final encoder hidden state: $\mathbf{s}_0 = \mathbf{h}_N$. At decoding step j, we compute

$$\mathbf{s}_j = \text{LSTM}\left(E_{trg}(y_{j-1}), \mathbf{s}_{j-1}\right)$$
$$\tilde{\mathbf{s}}_j = \tanh(W_{hs}[\mathbf{s}_j; \mathbf{c}_j] + \mathbf{b}_{hs})$$

The conditional probability that the j-th target word is generated is:

$$p(y_j \mid \boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(W_{so}\tilde{\mathbf{s}}_j + \mathbf{b}_{so}).$$

Here, E_{trg} is the target embedding lookup table, W_{hs} and W_{so} are weight matrices, and $\mathbf{b}_{hs}$ and $\mathbf{b}_{so}$ are bias vectors.

During decoding, beam search is used to find an output sequence with high conditional probability.

2.1.4 Long Short-Term Memory RNNs

At the heart of both the encoder and decoder are recurrent neural networks such as LSTMs [HS97]. These are computed as follows:

$$\mathbf{f}_i = \sigma\left(W_f\mathbf{x}_i + U_f\mathbf{h}_{i-1} + \mathbf{b}_f\right) \quad \text{forget gate} \tag{2.3}$$
$$\mathbf{i}_i = \sigma\left(W_{in}\mathbf{x}_i + U_{in}\mathbf{h}_{i-1} + \mathbf{b}_{in}\right) \quad \text{input gate} \tag{2.4}$$
$$\mathbf{o}_i = \sigma\left(W_o\mathbf{x}_i + U_o\mathbf{h}_{i-1} + \mathbf{b}_o\right) \quad \text{output gate} \tag{2.5}$$
$$\mathbf{u}_i = \tanh\left(W_u\mathbf{x}_i + U_u\mathbf{h}_{i-1} + \mathbf{b}_u\right) \quad \text{update} \tag{2.6}$$
$$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \mathbf{f}_i \odot \mathbf{c}_{i-1} \quad \text{cell} \tag{2.7}$$
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i) \quad \text{hidden} \tag{2.8}$$

Inputs $\mathbf{x}_i$ are embeddings or hidden states of a lower layer at timestep i, $\mathbf{h}_i$ are the hidden states and $\mathbf{c}_i$ the cell states. $W.$ and $U.$ denote parameter matrices, $\mathbf{b}.$ bias terms. LSTMs mitigate the vanishing gradients problem of vanilla RNNs by using a gating mechanism in which sigmoidal gates explicitly control how much information in the cell state should be kept or overwritten [Hoc98].

2.1.5 RNN-Free Models

RNNs have proven powerful sequence models in many tasks, but also possess a practical drawback: Their slow computation speed. The reason for this is that computations across a sequence cannot be parallelized because the computation at each time step depends on the outcome at the previous time step. This has led researchers to consider alternatives to RNNs, the most popular ones being time-delay/convolutional neural networks (TDNNs/CNNs) and self-attentional neural networks.

TDNNs/CNNs have been proposed in the late 1980s [WHH+87, WHH+89] for acoustic modeling, and have also been extensively used in computer vision [LBD+89]. However, only relatively recently have they been applied to discrete sequence modeling [KGB14, Kim14, SHG+14] and machine translation [GAG+17]. These networks define learnable filters over receptive fields of a fixed and usually small size. By stacking multiple layers the receptive field grows larger in each layer. Parallelization on modern GPU hardware is possible because each time step can be computed independently of the other time steps.

A limitation of TDNNs/CNNs is that the context is still fixed to a window of a certain size, even after stacking multiple layers. This is in contrast to RNNs that condition on infinitely long contexts, at least in theory. Recently, self-attentional models have been introduced [CDL16, PTDU16, LFdS+17] that combine support for arbitrarily large contexts and parallelization. They have been found to yield excellent results for machine translation [VSP+17].

The main idea behind self-attention is to condition the state for every time step on every other time steps by computing pairwise relevance scores. The attention mechanism is used to compute such relevance scores and makes sure that weights for all states sum up to 1. The new state is then computed as the weighted average over all other states. Formally, let $[\mathbf{x}_1 \dots \mathbf{x}_l]$ denote a sequence of state vectors. For each position $\mathbf{x}_i, 1 \leq i \leq l$, we compute the following:

$$e_{ij} = f(\mathbf{x}_i, \mathbf{x}_j) \qquad \forall l : 1 \leq j \leq l \tag{2.9}$$

$$\alpha_i = \text{softmax}(\mathbf{e}_i) \tag{2.10}$$

$$\mathbf{y}_i = \sum_{j=1}^{l} \alpha_{ij} \mathbf{x}_j \tag{2.11}$$

Here, f is a (potentially parametrized) similarity function, usually based on dot

products, bilinear combination, or a multi-layer perceptron. f is used to establish relevance between all pairs of sequence items. These are then run through a softmax and used to compute a weighted average over all states, independently at each position.

For sufficient model expressiveness, multiple layers can be stacked and interleaved with position-wise multi-layer perceptrons [VSP+17].

2.1.6 Modeling Units

Neural sequence models have traditionally used word tokens as modeling units [BDVJ03], based on the intuition that language is usually based on words to convey particular concepts, and on the idea of using word embeddings in continuous space to relate such word-based concepts to one another. This position can be challenged in several ways. First, some languages such as Chinese and Japanese have no clear concept of a word. Second, sparsity issues lead to poorly estimated word embeddings, particularly in languages with rich morphology or compounding. Third, these models cannot account for unseen words, a common problem even in languages without rich morphology.

An obvious alternative is to use characters as basic modeling unit. These form the basis of virtually every language that has a writing system, often do not suffer from sparsity issues, and can account for unknown words. In practice, character units require smaller embedding spaces and are much cheaper to compute at the softmax output projection. Despite this, they usually lead to more expensive models because sequence lengths are much longer, and sometimes larger hidden dimensions are required [BJM16]. Also, they often perform slightly worse than word-based models.

These trade-offs have led researchers to consider sub-word models, exploring the space between words and characters as units [MSD+12]. The currently most popular technique is byte-pair encoding [SHB16] which allows to flexibly choose a tradeoff between vocabulary size and sequence length.

Besides the issues mentioned above, there are two important factors in practice that determine whether characters, words, or sub-words are the most desirable representation. First, the data size: For small training data, sparsity issues can become severe and character-level models or small sub-word units often perform best, while for large data sets longer units are preferable. Second, when using an attention mechanism in an encoder-decoder model, it is desirable to have similar scales at both sides so that similarity scores are computed between comparable entities.

2.2 Automatic Speech Recognition

Hidden Markov model (HMM) have been explored in the 1970s for the recognition of continuous speech at IBM [JBM75] and Carnegie Mellon University [Bak75] and have provided the by far dominant framework for continuous speech recognition until very recently. As HMM-free, all-neural methods are achieving increasingly competitive accuracies in recent years, there is less clarity about what the best approach is compared to the translation scenario. All-neural methods include connectionist temporal classification [GMH13], attentional encoder-decoder models [CJLV16, BCS+16], and the recurrent neural aligner [SSRB17]. In this thesis we make use of both HMM-based and encoder decoder-based speech recognition and survey both below.

2.2.1 Preprocessing

Acoustic signals for speech are represented as discrete sequences of samples of the changing electric currents in a microphone. These samples are captured through an A/D converter at intervals of a certain frequency, often 16kHz, and quantized to e.g. 16-bit numbers. The resulting representation contains a large amount of numbers for every speech utterance, which is challenging for both computational and modeling reasons. The signal must therefore first be mapped to a space of much lower dimension while attempting to remove much of the redundancy present in a speech signal.

Based on the observation that humans can acquire the skill of interpreting speech signals in the frequency domain but are very poor at doing the same in the time domain, many preprocessing methods use the discrete Fourier transform to convert the signal into the frequency domain. This is done with windows of e.g. 16ms or 25ms size, chosen to be shorter than the length of most phonemes that occur in a signal. The windows are shifted by a step size of e.g. 10ms to capture the sequence of acoustic events. The feature space can be reduced by grouping the spectrum into bins that cover a certain range of frequencies each. To this end, the Mel scale can be used to divide into bins that correspond to perception by the human ear. The result is a feature representation called Mel filterbank features that will be the main technique used throughout this thesis. Other common techniques are cepstral coefficients that applies a series of further transformations, or perceptual linear prediction that follows a rather different predictive approach [RJ93].

2.2.2 HMM-based Speech Recognition

HMM-based speech recognition decomposes the task using the intuition of the noisy channel model [Sha48]:

$$\hat{W} = \underset{W}{\operatorname{argmax}} Pr(W|X) = \underset{W}{\operatorname{argmax}} \frac{Pr(X|W) \cdot Pr(W)}{Pr(X)}$$
$$= \underset{W}{\operatorname{argmax}} Pr(X|W) \cdot Pr(W),$$

where W denotes a word sequence, X is an acoustic observation, and $\hat{W}$ is our best guess for the true uttered word sequence. The noisy channel considers the correct word sequence as having been garbled by a noisy channel so that only a noisy signal is observed; the task is then to decode the original signal (the word sequence) based on the noisy signal (the acoustic signal). In the derived formula, $P(W)$ is a linguistically motivated prior, and $P(X|W)$ is the likelihood term, i.e. the probability of observing the acoustic sequence given a particular word sequence. The noisy channel model offers a convenient machinery for combining the prior and the conditional model.

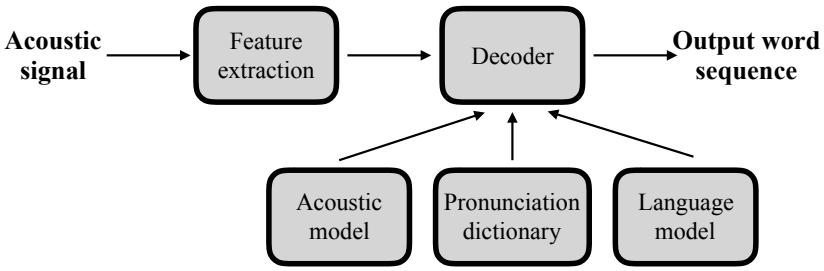


Figure 2.2: Block diagram of HMM-based speech recognition.

Figure 2.2 illustrates how this approach is put to practice. The speech signal is preprocessed as outlined in Section 2.2.1 to obtain a sequence of feature vectors X. $P(W)$ is usually computed by an n-gram language model [Sha48, CG98], and $P(X|W)$ can be modeled through a GMM-HMM or hybrid acoustic model. Furthermore, because acoustic models work over phonemes but language models over words, a pronunciation dictionary is necessary to establish a proper mapping. Importantly, the language model, acoustic model, and dictionary are estimated independently of each other. Decoding is realized by employing a beam search over the hypothesis space. Because this model is often unable to handle punctuation and produces numbers and other special terms in a pronounced format that is hard to read, a post-processing step is necessary to produce legible output.

2.2.3 Encoder-Decoder-Based Speech Recognition

A drawback of the HMM-based approach is the need for creating pronunciation dictionaries, posing a considerable burden when developing ASR for new languages. A recent trend has therefore been to learn dictionaries implicitly via neural methods, often referred to as end-to-end ASR. One common approach is connectionist temporal classification (CTC) [GMH13] which computes a high-level representation of a sequence of acoustic feature vectors using an RNN, and then classifies each frame by assigning it e.g. an alphabet letter in the case of English. Blank labels and mechanisms to deal with repeated characters are used to account for the fact that the input sequences are much longer than the output sequences, and the CTC loss function marginalizes over all equivalent outputs (also called segmentations) via a dynamic programming approach.

CTC eliminates the need to create dictionaries, but is still dependent on external language models for good performance. This is because outputs are conditionally independent of each other, therefore only a weak implicit language model can be learned in the RNN layers. A remedy to this is provided by the listen-attend-spell model [CJLV16], another HMM-free approach to speech recognition that uses an encoder-decoder architecture as described in Section 2.1 with a few modifications.

The learned word embeddings are replaced by acoustic features (see Section 2.2.2), such that the encoder component now serves as an acoustic model. Because acoustic sequences are very long compared to text inputs, the encoder performs downsampling to make memory and computation time manageable. This is achieved through a pyramidal LSTM, a stack of LSTM layers where pairs of consecutive outputs of a layer are concatenated before being fed to the next layer, such that the number of states is halved between layers. As output units, characters rather than words are used and yield much better results. This can be explained by a more direct correspondence between encoded units (downsampled audio frames) and outputs, and by the fact that training data in number of words is usually much smaller for speech recognition than machine translation, causing data sparsity problems when using full word-based vocabularies.

Note several important differences to the HMM-based approach:

- No noisy-channel assumption is made. Instead, $P(W \mid X)$ is modeled directly.
- All parameters are trained jointly. This can be a desirable property and can also simplify implementation and maintenance. On the other hand, exploiting auxiliary data such as monolingual data for language modeling becomes less straight-forward.

- No pronunciation dictionary is required, the creation of which is one of the major burdens in traditional speech recognition.
- This model is more flexible regarding output text normalization, e.g. can be trained to directly produce output that is properly cased, punctuated, and uses properly formatted numbers.

2.3 Considerations for Cascaded Systems

Even in the hypothetical case of a perfect speech recognition system, simply chaining the aforementioned speech recognition and machine translation models as-is would be problematic because the speech recognition output differs from the machine translation training data in significant ways. Such train-test mismatch is a reason of degradation with many machine learning models. The most important problems are listed below. Not all of these are dealt with explicitly in this thesis, but a good understanding is nevertheless important. Section 2.4 will survey the proposed solutions in existing literature.

2.3.1 Text Normalization

Speech recognition output is often unpunctuated, contains numbers as written-out words, and may otherwise differ regarding normalization conventions. Therefore, the speech recognition output must be post-processed to follow conventions of the translation training data as closely as possible in order to minimize the data mismatch and avoid out-of-vocabulary problems.

2.3.2 Segmentation

In scenarios where the input is either a very long audio, or comes as a live stream of unbounded length, speech recognition outputs may be unsegmented or possess segmentation guided by acoustic properties rather than into linguistically meaningful sentences. This is in contrast to the translation training data that usually consists of well-formed individual sentences. In addition to the mismatch, reordering phenomena often make it necessary to be in possession of a full sentence in order to translate it. Therefore, a segmentation step must be performed, often jointly with the punctuation step needed for text normalization. When doing simultaneous translation this is further complicated by the lack of future context, and often one must trade off between high translation accuracy and low latency.

2.3.3 Style

Spoken language is often more casual, less grammatical, and may suffer from disfluencies such as false starts, repetitions and filler words. It uses a limited vocabulary, makes many inexplicit references, and is structured as shorter intonation units that are often not topically coherent [CD87]. None of these phenomena are likely to be appropriately captured by the textual translation training data. It can therefore be beneficial to remove disfluencies and correct incorrect grammar before attempting translation. Note that the style mismatch can be of varying degrees. For example, a well-rehearsed lecture is usually closer in style to written language than a spontaneous dialog between family members.

2.3.4 Paralinguistic Information

Spoken representations of text often do not represent paralinguistic information such as word emphasis. However, such information can be very useful for translation purposes, especially when generating speech outputs, but also for text outputs. Some prior work approaches this issue by including emphasis information into the ASR output and proposing ways for translating such representations [DSN18, KTS+13, AOB12, AAB06, TGN13].

2.4 Speech Translation: Prior Research

2.4.1 Early Work

Early efforts to speech translation around the Janus project [WJM+91, WCE+93, LGG+95] used a simple cascading approach where speech recognizer and translation systems were built separately and the best hypothesis of the former would be used as input to the latter. Another early approach used template matching for the translation stage [WW91]. The possibility of speech-to-speech translation, producing speech output instead of text output, has also been considered, for example by Lavie et al. [LGG+95]. However, whether to output text or speech has usually been treated as a user interface issue that should be evaluated separately and can be achieved by simply applying speech synthesis to the speech translation text outputs. These early efforts were focused on limited domains such as a scheduling dialog scenario, in which results were promising. However, errors propagated from the speech recognizer were especially challenging for interlingua-based machine translation, widespread at this time, because it relied on parsers that in turn often expected well-formed inputs [WCE+93, LGG+95, LGGP03].

Along with the trend in general machine translation, following work turned increasingly to data-driven approaches [WW98, TMS+98, BBF+02, SSN07].

2.4.2 Integration of Components

Research efforts soon turned to the question of how to better integrate the recognition and translation stage. Ney describes a probabilistic framework for fully integrated data-driven approaches [Ney99]. According to this framework, speech translation seeks a solution to

$$\underset{e_1^I}{\operatorname{argmax}} \, Pr\left(e_1^I \mid x_1^T\right),$$

or equivalently to

$$\underset{e_1^I}{\operatorname{argmax}} \, Pr\left(e_1^I\right) Pr\left(x_1^T \mid e_1^I\right),$$

where e_1^I are the translated words in the target sentence of length I, and x_1^T are the frames in the acoustic sequence of length T. While the speech transcript f_1^J is ultimately of no interest to us,[1] we can bring it into the model by marginalizing over all possibilities:

$$\underset{e_1^I}{\operatorname{argmax}} \, Pr\left(e_1^I\right) Pr\left(x_1^T \mid e_1^I\right) = \underset{e_1^I}{\operatorname{argmax}} \, Pr\left(e_1^I\right) \sum_{f_1^J} Pr\left(f_1^J \mid e_1^I\right) Pr\left(x_1^T \mid f_1^J\right)$$

Explicit marginalization is intractable, efforts can therefore be categorized by the approximations they take.

A first simple way of better integrated recognition and translation is moving from 1-best translation to n-best translation. This option was explored by Quan et al. [QFC05] and Lee et al. [LLL07], but has been mentioned in earlier works as well [LGG+95]. The n-best approach corresponds to replacing the sum over all possible transcripts f_1^J by a sum over only the list of the n-best outputs of the speech recognizer. This was found to improve results in the travel domain while being rather expensive.

On the opposite end of the spectrum was a fully integrated approach that used the finite state transducer (FST) formalism to model a decoder that directly produced translations from audio inputs, with corresponding transcript as a side product [Vid97, BR01, CNO+04, PGJ+07]. Conveniently, this approach allowed using general-purpose FST tools to conduct an approximate search over the intractable search

[1] One may argue that this is not always true: Speech translation user interfaces often display both the transcript and the translation to the user, making it necessary to produce not only a translation but also a transcript.

space. Conceptually, this amounted to permitting the full marginalization in the model, but exploring only a limited region of the search space by using pruning and other search heuristics. Results were promising in a limited domain scenario, hotel front desk dialogs, but also somewhat inconsistent and revealed robustness issues.

Follow-up work suggested using word graphs such as lattices [SJVS04, ZKYL05] or confusion networks [BF05] as a more compact and computationally convenient alternative to n-best lists. Word graphs integrated nicely with word-based translation models based on the IBM models [BPPM93]. While slight improvements were observed in some cases, the main advantages turned out to be of computational nature. After phrase-based translation [KOM03] emerged, Matusov et al. developed a method for performing lattice decoding in a phrase-based context and reported good results [MNS05, MHN08]. Evaluated speech translation tasks were now moving from limited travel-related tasks to open-domain tasks such as translation of TED talks [TED12] or speeches from the European parliament.

2.4.3 Advanced Topics

With accuracies of even the cascaded approaches reaching more acceptable levels, researchers turned to tackle the more advanced aspects of speech translation (see Section 2.3). Domain adaptation techniques were used to adapt models to the spoken language domain [LGGP03, Füg08]. It was shown how to automatically optimize recognition outputs that lead to better translation quality [KBT+15, HDA11]. The scenario of translating longer speeches rather than short utterances was explored. This scenario is challenging because machine translation models can usually handle only single sentences, but sentences boundaries are unknown. The works by Matusov et al. [MMN06] and by Fügen [Füg08] thus segment the speech recognition output and insert punctuation to optimize translation performance. As such systems are often deployed in a real-time situation, simultaneous speech translation with low latency is a very useful feature [Füg08, ONS+14, NPH+18]. As spoken language contains disfluencies that hurt readability of the transcript and causes translation errors, disfluency removal [CFH+13] can be beneficial.

2.4.4 Speech Translation Corpora

It is important to realize that all efforts to this point had used separate speech recognition and machine translation translation corpora to train models. Translated speech data was only available in small quantities and was useful only as validation

and test data, but not for training. This often led to domain mismatch between the translation and recognition components, and it was recognized that addressing this issue would also result in a tighter integration of both models. One approach was to generate synthetic speech recognition outputs by imagining recognition errors, which can then be used to train a more robust translation model [TMD14, RGLF15]. Some efforts were devoted to collecting speech translation corpora that could be used for training purposes [SKM+12, PKL+13, GAAD+18, KBK18]. However, initial attempts by Post et a. did not observe gains from training a machine translation model directly on speech recognition output [PKL+13]. Paulik made use of interpreted speech data, i.e. speech utterances paired with the utterances of a human interpreter [Pau10]. Such data can be easier to obtain in some situations, but is a less direct form of supervision for a speech-to-text translation system.

2.4.5 New Chances and Challenges

Machine translation and speech recognition, as well as machine learning in general, underwent several major changes in the past years and decades. Machine translation transitioned from interlingua-based approaches to statistical approach and then to end-to-end trainable neural approaches in recent years. Speech recognition has long been dominated by HMM-based models, but these have recently been challenged (though not replaced) by end-to-end trainable neural approaches. These changes re-raise some of the questions and issues related to speech translation. For example, it has been observed that neural machine translation is less robust to speech recognition errors than statistical machine translation models, although overall accuracies are improved enough to make up for this [RDBF17]. Prior approaches to lattice translation are no longer applicable with neural machine translation because output independence assumptions no longer hold. Finally, the question of how to leverage end-to-end speech translation corpora has not been conclusively answered.

Chapter 3

Databases

This chapter introduces two speech recognition corpora and one speech translation corpus that will be used throughout this thesis. The chapter is then concluded by discussing evaluation metrics that can be used to assess model performance based on the given data.

3.1 WSJ

The Wallstreet Journal (WSJ) continuous speech recognition corpus [PB92] contains English sentences selected from the Wallstreet Journal and read by a variety of speakers. It is a widely used benchmark in speech recognition. The speech being read instead of spontaneous places this corpus is on the easier side of the spectrum. Table 3.1 shows statistics of this corpus, and Table 3.2 lists some examples. We will make use of it to develop our speech recognition baseline in Section 4.1.

Data set	Audio duration	Number of utterances	Number of words
WSJ/Train (*I284*)	81h	37,416	645,876
WSJ/Dev (*dev93*)	1:05h	503	8,334
WSJ/Test (*eval92*)	0:42h	333	5,700

Table 3.1: WSJ corpus statistics.

1	*he chats with her briefly then resumes his political analysis*
2	*quote we have a dirty war exactly the same as in argentina unquote he says*
3	*quote it isn't acceptable but it's understandable unquote noise*
4	*and what should the government do about the murders*
5	*nothing says the businessman*

Table 3.2: WSJ corpus example utterances. A curiosity are the explicitly spoken punctuation marks.

3.2 TEDLIUM

The TEDLIUM corpus [RDE14] is a widely used corpus of recorded TED talks [TED12], containing talks on technology, entertainment, and design, delivered in English by well-prepared, high-profile speakers. The corpus includes the recorded audios and transcripts created by volunteers. In this thesis, we use its second edition. Compared to WSJ this is a more difficult speech recognition task because it contains realistic, spontaneous speech. However, the level of difficulty is still only moderate because the audio quality is excellent and the preparedness of the speakers results in only a limited amount of disfluencies and other characteristics of spontaneous speech. Table 3.3 shows the statistics, and Table 3.4 lists example utterances.

Data set	Audio duration	Number of utterances	Number of words
TEDLIUM/Train	206h	92,968	2,250,412
TEDLIUM/Dev	1:35h	507	18,226
TEDLIUM/Test	2:37h	1,157	28,432

Table 3.3: TEDLIUM corpus statistics.

1	*and now we 're trying to go from that digital code*
2	*into a new phase of biology with designing and synthesizing life so we 've always been trying to ask big questions*
3	*what is life is something that i think many biologists have been trying to understand at various levels we 've tried various approaches*
4	*paring it down to minimal components we 've been digitizing it now for almost twenty years when we sequenced the human genome it was going from the analog world of biology into the digital world of the computer*
5	*now we 're trying to ask can we regenerate life*

Table 3.4: TEDLIUM corpus example utterances.

	Audio duration	Number of sentences	Number of words
Fisher/Train	162h	138,819	1,810,385
Fisher/Dev	4:21h	3,979	50,700
Fisher/Dev2	4:28h	3,961	47,946
Fisher/Test	4:14h	3,641	47,896
Callhome/Train	13h	15,080	181,311
Callhome/Devtest	3:33h	3,966	47,045
Callhome/Evltest	1:47h	1,829	23,626

Table 3.5: Fisher-Callhome corpus statistics.

	1-best WER	Lattice oracle WER
Fisher/Dev	41.3	19.3
Fisher/Dev2	40.0	19.4
Fisher/Test	36.5	16.1
Callhome/Devtest	64.7	36.4
Callhome/Evltest	65.3	37.9

Table 3.6: Accuracy of ASR outputs in Fisher-Callhome corpus.

3.3 Fisher-Callhome

We conduct most of our speech translation experiments on the Fisher and Callhome Spanish–English Speech Translation Corpus [PKL+13]. This is a corpus of Spanish telephone conversations that includes audio recordings, transcripts, and translations into English, as well as automatic transcripts and speech recognition lattices created by a Kaldi system. The Fisher portion consists of telephone conversations between strangers, while the Callhome portion contains telephone conversations between relatives or friends. Data statistics are shown in Table 3.5. For the Fisher development and test sentences, four translation references exist, while for sentences from Fisher/Train and Callhome there is only a single reference translation. ASR word error rates (WER) are relatively high (Table 3.6), due to the spontaneous speaking style and challenging acoustic conditions. From a translation viewpoint, on the other hand, the data can be considered as relatively easy with regards to both the topical domain and particular language pair. Table 3.7 lists some example utterances with their translations. Lattices contain on average 3.4 (Fisher/Train) or 4.5 (Callhome/Train) times more words than their corresponding reference transcripts. The average sentence length is between 11.8 and 13.1.

	Source sentence	Translated sentence
1	*aló*	*hello*
2	*aló*	*hello*
3	*hola*	*hello*
4	*hola*	*hello*
5	*con quién hablo*	*with whom am i speaking*
6	*eh silvia sí cómo se llama*	*eh silvia yes what is your name*
7	*hola silvia eh yo me llamo nicole*	*hello silvia eh my name is nicole*
8	*ah mucho gusto*	*ah nice to meet you*
9	*mucho gusto em y dónde está usted*	*nice to meet you em and where are you from*
10	*n eh yo estoy en filadelfia*	*eh i'm in philadelphia*
11	*ay mira yo estoy en nueva york*	*aye look i'm in new york*
12	*y ust ah no sabía que el estudio incluía gente tan lejos pero que bueno*	*and you ah i didn't know that the study included people so far but how nice*
13	*sí yo participé en un es un estudio así em hace como un año y*	*yes i participated in a study like this em like a year ago and*
14	*ah ah mm mi hijo participó en ese sí*	*ah ah hmm my son participated in this yes*
15	*sí supe que es para todo país o sea para gente de todo país*	*yes i knew that it was in the whole country like people from the whole country*

Table 3.7: Fisher corpus example utterances.

3.4 Evaluation

Throughout this thesis we mainly rely on automatic reference-based evaluation to assess accuracy and compare different models. The main metrics we use are word error rate (WER) for speech recognition and BLEU (short for *bilingual evaluation understudy*) for translation.

3.4.1 WER

WER is based on the edit distance, also referred to as Levenshtein distance. The main idea is to count the minimum number of word-level edits necessary to transform the incorrect output string into the correct reference string. Edits can be substitutions, insertions, and deletions. The WER is then defined as

$$\text{WER} = \frac{\text{substitutions} + \text{insertions} + \text{deletions}}{\text{reference length}} \times 100\%.$$

The minimum number of edits can be computed efficiently through a dynamic programming algorithm. In languages where no clear word boundaries exist, such as

Chinese and Japanese, the character edit rate can be used instead which works exactly the same but operates on character-level instead of word-level.

While WER is meaningful even when computed for individual test sentences, it is usually computed at the corpus level, so that longer sentences in the corpus are given proportionally more weight in the final score than shorter sentences.

3.4.2 BLEU

The BLEU score [PRWZ02] is based on a precision term that computes how many n-grams in an output actually appear in the reference string, and a brevity penalty that prevents gaming the metric by generating short translations that contain only safe or highly confident n-grams:

$$\text{BLEU4} = \text{brevity penalty} \times \prod_{i=1}^{4} \text{precision}_i,$$

where

$$\text{precision}_i = \frac{\sum_{n\text{gram} \in \text{output}} \text{Count}_{\text{clip}}\left(n\text{gram}\right)}{\sum_{n\text{gram}' \in \text{output}} \text{Count}\left(n\text{gram}'\right)}.$$

Commonly, precision scores for n-grams up to order 4 are computed, as shown in the equation. $\text{Count}_{\text{clip}}\left(n\text{gram}\right)$ refers to a clipped count, i.e. the number of output n-grams that appear in the reference where each n-gram in the reference can be used only once.

The brevity penalty is defined as follows:

$$\text{brevity penalty} = \min(1, e^{1-\frac{\text{reference length}}{\text{output length}}}).$$

BLEU scores can also be computed against multiple references, in which case the clipped counts are clipped at the maximum count of n-grams which occurs in a single reference, and the brevity penalty is computed against the length of the reference closest in size to the candidate translation.

Note that the n-gram precision for higher order n-grams, e.g. 4-grams, is not unlikely to be zero for a single sentence, which would result in the BLEU score for a whole sentence becoming zero. For this reason, BLEU-scores are computed at the corpus level, not at the level of individual sentences.

Chapter 4

All-Neural Speech Recognition

As a first step toward advancing speech translation using recent deep learning techniques, we first experiment with a speech *recognition* task. Speech recognition can be seen as an easier version (or subtask) of the speech translation task, with acoustic modeling similarly challenging but without the complexities of having to produce outputs in a different language. This makes speech recognition an ideal test bed for improving models, debugging implementations, and conducting analyses. A good speech recognition model can also directly improve speech translation by helping form a strong cascade. This chapter first describes efforts to produce and analyze a strong baseline based on state-of-the-art modeling techniques from the available literature, and then explores novel improvements. Importantly, this chapter directly models $P(W \mid X)$, the probability of the transcript W given the audio signal X, with no decompositions. This will allow us in Chapter 6 to extend the models introduced here to be capable of *translating* speech.

4.1 Analyzing the State of the Art

We start by describing the development of a strong baseline system by using the initial work by Chan et al. [CJLV16] as a starting point. This work has formed the basis of many subsequent works on attentional speech recognition models and has been reported to yield promising results competitive with other models.

4.1.1 Basic Settings

We choose basic settings based on findings in literature and on our own preliminary experiments. These settings are used throughout the thesis unless otherwise noted.

For audio preprocessing, we extract 40-dimensional Mel filterbank features with per-speaker mean and variance normalization. We exclude utterances longer than 1500 frames from training to keep memory requirements manageable. The encoder-decoder attention is MLP-based, and the decoder uses a single LSTM layer. The number of hidden units is 128 for the encoder-decoder attention MLP, 64 for target character embeddings, and 512 elsewhere unless otherwise noted. The model uses input feeding [LPM15]. For the encoder, we use bidirectional LSTMs with 256 hidden units per direction. During inference, we use beam search and length normalization.

We set the batch size dynamically based on the number of source and target tokens of the longest sentence in the minibatch. We choose the average batch size such that most of the available GPU memory is used, here 24 utterances per minibatch. We use Adam [KB14] with initial learning rate of 0.0003, decayed by 0.5 when validation WER did not improve over 10 epochs initially and 5 epochs after the first decay. We implement our method based on the DyNet neural network toolkit [NDG+17].

4.1.2 Findings

We start our exploration using a vanilla implementation of the listen-attend-spell architecture by Chan et a. [CJLV16] that features a pyramidal LSTM encoder and the settings just described.

Overfitting. Overfitting is a major challenge in attentional speech recognition and proper generalization techniques are therefore an important consideration. While prior work has often used weight noise and weight decay for normalization, we have observed inconsistent gains and a necessity to perform expensive grid search to tune the respective hyperparameters. Instead, we opt for a solution that uses variational dropout combined with target character type dropout, both proposed by Gal et al. [GG16], both because it yielded best results and because it was more stable across the various models and settings.

In particular, recurrent dropout is applied to all LSTMs that are used in the encoder or decoder. Independently for each LSTM layer, direction, and minibatch element, we draw dropout masks $\mathbf{m}_x$ and $\mathbf{m}_h$ from a multivariate Bernoulli distribution: $\mathbf{m}_x \sim \text{Bernoulli}_{d_x}(1-p)$ and $\mathbf{m}_h \sim \text{Bernoulli}_{d_h}(1-p)$ where p it the dropout probability. We then replace $\mathbf{h}_{i-1}$ by $\frac{1}{1-p}\mathbf{m}_h \odot \mathbf{h}_{i-1}$ and $\mathbf{x}_i$ by $\frac{1}{1-p}\mathbf{m}_x \odot \mathbf{x}_i$ in Equations 2.3 through 2.6 at training time and obtain the following:

$$\mathbf{f}_i = \sigma\left(W_f \frac{1}{1-p}\mathbf{m}_x \odot \mathbf{x}_i + U_f \frac{1}{1-p}\mathbf{m}_h \odot \mathbf{h}_{i-1} + \mathbf{b}_f\right) \quad \text{forget gate} \quad (4.1)$$

$$\mathbf{i}_i = \sigma\left(W_{in} \frac{1}{1-p}\mathbf{m}_x \odot \mathbf{x}_i + U_{in} \frac{1}{1-p}\mathbf{m}_h \odot \mathbf{h}_{i-1} + \mathbf{b}_{in}\right) \quad \text{input gate} \quad (4.2)$$

$$\mathbf{o}_i = \sigma\left(W_o \frac{1}{1-p}\mathbf{m}_x \odot \mathbf{x}_i + U_o \frac{1}{1-p}\mathbf{m}_h \odot \mathbf{h}_{i-1} + \mathbf{b}_o\right) \quad \text{output gate} \quad (4.3)$$

$$\mathbf{u}_i = \tanh\left(W_u \frac{1}{1-p}\mathbf{m}_x \odot \mathbf{x}_i + U_u \frac{1}{1-p}\mathbf{m}_h \odot \mathbf{h}_{i-1} + \mathbf{b}_u\right) \quad \text{update} \quad (4.4)$$

$$(4.5)$$

Crucially, the same mask is applied at every time step, and only at training time but not at test time. For word type dropout, we randomly choose a fraction p of the word types from the vocabulary and replace all corresponding word embeddings by a zero-vector during training. More details are given in [GG16].

Encoder architecture. We implemented and tested several of the advanced encoder architectures proposed by Zhang et al. [ZCJ17]. While some of their models, in particular those that feature convolutional networks or convolutional LSTMs, did not yield improvements over our baseline, we did obtain good results using an LSTM/network-in-network architecture that stacks blocks consisting of a bidirectional LSTM, a network-in-network (NiN) projection, batch normalization [IS15], and a rectified linear unit (Figure 4.1). The final LSTM/NiN block is topped off by another bidirectional LSTM layer. NiN denotes a simple linear projection applied at every time step, possibly performing downsampling by concatenating pairs of adjacent projection inputs.

Label smoothing. Label smoothing has been found to prevent overconfidence and to improve decoding accuracy by Chorowski et al. [CJ17]. While the authors suggest a variant that smoothes across labels of adjacent time steps, we have found the simpler approach of uniform label smoothing [SVI+16] to yield best results. Specifically, instead of assigning all probability mass to the correct label at a particular decoding time step, we only assign it a probability of β, and spread the remaining probability mass $1 - \beta$ uniformly over all remaining labels.

Fixed embedding norm. In the context of neural machine translation, Nguyen et al. suggest fixing the norm of target embeddings to a fixed value in order to prevent bias toward common tokens [NC18]. We confirm consistent improvements in the

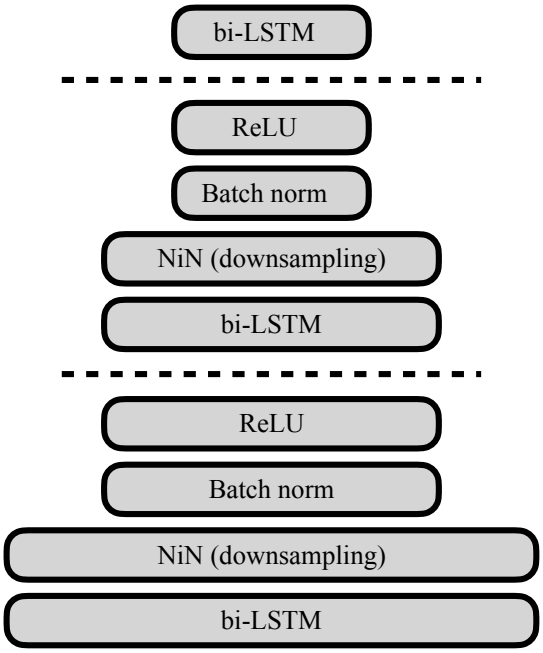


Figure 4.1: Block diagram of the LSTM/NiN encoder model. The encoder contains three LSTM/NiN layers, followed by a final LSTM layer. Downsampling operations are illustrated through reduced layers widths.

speech recognition task when tuning the fixed value appropriately. Specifically, we replace character embeddings $\mathbf{e}$ by $r\frac{\mathbf{e}}{\|e\|_2}$ for a fixed scalar hyper parameter r.

Stopping criterion. Despite having received little attention in the literature, we have found a well-calibrated early stopping criterion to be of high importance. In particular, we observe improvements when performing a full beam search during validation checkpoints and writing out tentative best models according to word error rate, compared to not conducting beam search or measuring only perplexity.

Speed and memory optimization. When modeling audio, the corresponding input sequences can get very long and consume a lot of memory. This in turn hurts training speed significantly because we can only use small mini batch sizes, and therefore miss out on the main advantage of using modern GPU hardware, namely highly optimized batch operations. Modern dynamic deep learning frameworks allow defining computation graphs in a very natural and readable way, such that the LSTM equations (2.3-2.8) can be directly translated into code. While greatly facilitating prototyping and model extension, this sometimes comes at the cost of increasing memory requirements. In particular, in case of our employed deep

learning framework DyNet [NDG+17], this led to memory being allocated for each of the elementary operations. To make memory manageable in the case of speech inputs, we implement a solution that performs most intermediate steps in temporary memory, but retains the flexibility to be used in a natural `for` loop which is important for decoding. Namely, we proceed in three steps, the first computing Equations 2.3-2.6 (or 4.1-4.4, respectively), the second step computing 2.7, and the third step computing 2.8. Each of these requires memory for activations and gradients, but no memory is needed for intermediate operations and memory usage is reduced about threefold compared to DyNet's original implementation.

4.1.3 Ablation Study

We conduct an ablation study to provide empirical evidence for the effectiveness of the methods discussed in Section 4.1.2. In particular, we train and test models on the respective splits of the WSJ dataset 3.1, starting with the basic settings described in Section 4.1.1 and testing the effectiveness of several features individually through a leave-one-out ablation experiment. We can see in Table 4.1 that the combined improvement amounted to 2.89% WER absolute, or about 20% in relative terms. The use of label smoothing was particularly effective, while all methods except for the fixed norm contributed strongly to the final results.

Model	WER
Baseline model	16.91
Full − dropout	14.91
Full − label smoothing	16.18
Full − fixed norm	14.16
Full − NiN encoder	15.11
Full model	14.02

Table 4.1: Baseline ASR ablation results. The table shows the model with all experimental features, leave-one-out variants, and leave-all-out model.

4.1.4 Comparison to Other Approaches

We compare the speech recognition model developed above on the TEDLIUM benchmark to several other models and toolkits, including ESPnet [WHK+18], a CTC system similar to the model described by Zenkel et al. [ZSN+18], the numbers published

in the original TEDLIUM paper [RDE14] that are based on a hybrid DNN/HMM model by employing Kaldi [PGB+11], and an in-house Janus system [SMFW01] with a 5-layer BiLSTM acoustic model and a large language model. The results in Table 4.2 show that our model is competitive with other end-to-end models. It also shows that these do not quite achieve the accuracy of a highly tuned HMM/DNN system, although it must be noted that these make use of a language model trained on large data which is not available to our model. Removing the extra language model data can lead to a degradation of up to 25% relative, according to the numbers by Rousseau et al. [RDE14].

Model	Dev WER	Test WER
ESPnet [WHK+18]	19.8	18.6
CTC	16.0	16.4
HMM/DNN [RDE14]	10.4	11.3
Janus	13.1	12.1
Ours	15.4	16.0

Table 4.2: ASR toolkit comparison on TEDLIUM dev and test sets.

4.1.5 Analysis of Character-Level Behavior

To obtain a better understanding of our model's behavior, we conduct a detailed analysis. First, we plot its generated attention matrix for an example sentence in Figure 4.2. Such visualizations can be a helpful debugging tool and we can see that the attention is producing an intuitively reasonable alignment between input audio and output characters.

We also examine the open-vocabulary nature of the model. Note that because we generate outputs character by character, we are able to generate any possible word, even if this word has never been seen during training.[1] Thus, it is insightful to see whether new words were actually generated in practice, and whether such new word creations are performed correctly.

Looking at unique word types, we observe 13724 word types in the WSJ training corpus. The generated output for the test set contains 1897 word types, similar to but slightly higher than the 1835 word types appearing in the reference. Relative to the training text, the generated text contains 5.98% unknown words (newly created words

[1] Of course, words that contain unknown characters cannot be generated, but for European languages as regarded in this thesis such cases occur rarely if at all.

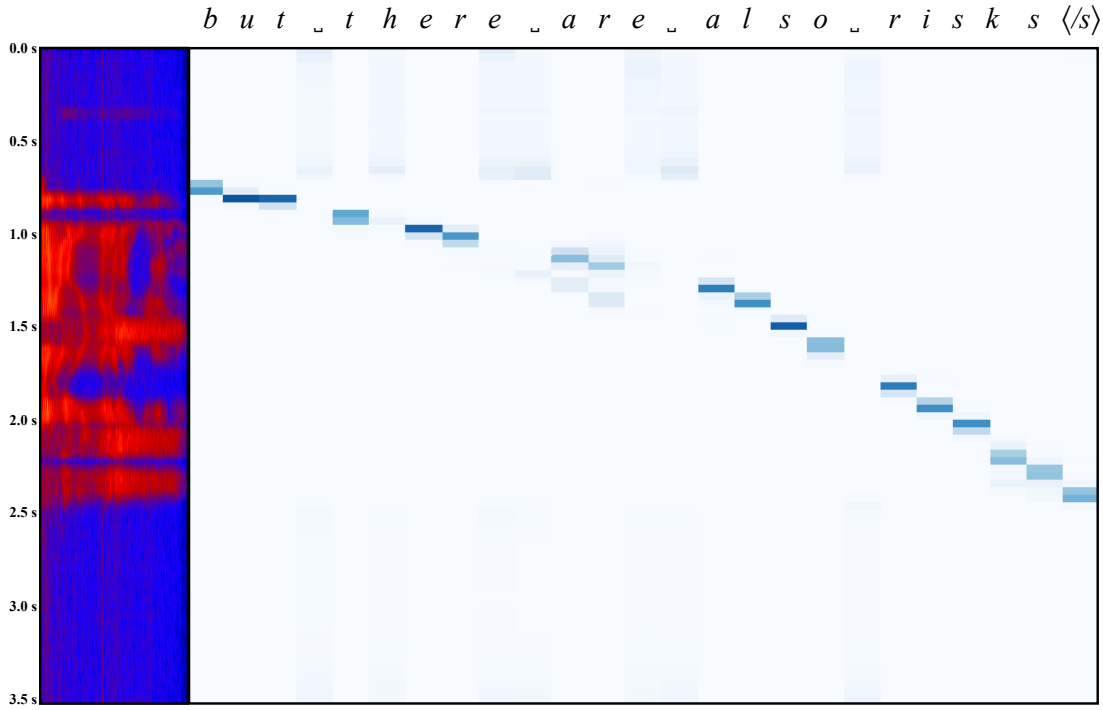


Figure 4.2: Attention matrix for a short, correctly recognized sentence.

that do not appear in the training data), the reference slightly less at 4.89%. Out of the newly generated words, 21.99% appear in the corresponding reference sentence (precision), likewise out of the unknown words from the reference 26.88% appear in the generated output (recall). To summarize, this open vocabulary model was able to recover unknown words in about a quarter of cases, but also produced a significant amount of errors due to incorrectly generated new words. To examine how severe these errors are, Table 4.3 shows the top 5 correct or incorrect newly created words and the top 5 missed unknown words. We see that some but not all of the successfully recovered unknown words correspond to morphological variants of words that were probably seen during training (*populists, amends*). Moreover, some of the unrecovered OOVs corresponded to rare words or proper nouns that were transcribed using intuitive but wrong alternative spelling (*ciba* vs. *seeba*, *multiplicity* vs. *multiplecity*), while some are plain errors (*strongers* vs. *astronomers*).

4.1.6 Related Work

A range of largely contemporaneous related work has proposed further extensions, even to the point where the outperforming of HMM-based systems has been reported

Recovered unknown words	Incorrect created words	Unrecovered unknown words
barry (3)	seeba (6)	ciba (6)
corning (2)	strongers (4)	aircoa (4)
populists (2)	theses (4)	astronomers (4)
amends (2)	multiplecity (3)	wang (4)
convoy (2)	wholestructure (3)	multiplicity (3)

Table 4.3: Top 5 examples for three kinds of unknown words. Namely, we show the top successfully recovered unknown words, incorrectly created new words, and unknown words that were not recovered.

[CSW+18]. We may think of them as reducing the gap between the large available "bag of tricks" of conventional models and the still smaller bag of tricks available for attentional speech recognition. Exploring these is beyond the scope of this thesis, but would likely have a positive impact on both our baseline and proposed method results in the upcoming chapters.

Improvements include monotonic attention [TSN17], integrating multiple tasks [TTLL17, TSW+17], improved decoding [CJ17], word-piece models, multi-head attention, and optimization improvements [CSW+18].

4.2 Self-Attentional Acoustic Models

Having established a strong baseline, we move on to advancing this model by improving upon its encoder component which takes on the function of an acoustic model. In order to transform an acoustic signal into a useful abstract representation, acoustic models must generally take into account the complex interplay of local and global dependencies in an acoustic signal. At a local, temporally constrained level, we observe concrete linguistic events (phonemes), while at a global level the signal is influenced by factors such as channel and voice properties. Traditional acoustic models reflect this intuition about global and local dependencies by first applying a normalization phase, a global operation that aims at producing invariance with respect to channel and speaker characteristics. After this, traditionally a hidden Markov model is applied over polyphones, modeling only local dependencies (beads-on-a-string view [Ost99]). This restriction has in part been motivated by the intuition that global effects should no longer affect the signal at this stage.

However, the empirical success of RNNs for acoustic modeling [SSB15] has challenged this intuition and indicated that consideration of the global context is still

beneficial at this stage. Unfortunately, RNNs suffer from slow computation speed and may not be able to optimally exploit long-range context. Self-attentional architectures [CDL16, PTDU16, LFdS+17] have recently shown promising results as an alternative to RNNs for modeling discrete sequences [VSP+17]. These models relate different positions in a sequence by computing pairwise similarities, in order to compute a higher level representation of the sequence. Self-attention is attractive (1) computationally because it can be efficiently implemented through batched tensor multiplication, and (2) from a modeling perspective because it allows direct conditioning on both short range-context and long-range context, without the need to pass information through many intermediate states as is the case with RNNs.

In the following, we explore self-attentional architectures for acoustic modeling, by replacing the LSTM-based encoder component of our baseline in Section 4.1 with self-attention.

4.2.1 Challenges and Benefits

In order to make self-attentional architectures work for acoustic modeling, several challenges must be addressed. First, self-attention computes the similarity of each pair of inputs, so that the amount of memory grows quadratically with respect to the sequence length. This is problematic for modeling acoustic sequences, because these can get very long, e.g. our training utterances contain up to 2026 frames (800 on average). To address this issue we apply downsampling through tensor reshaping before feeding a sequence into the self-attentional layers.

The second challenge is incorporating positional information into the model. Unlike an RNN, self-attention has no inherent mechanism of modeling sequence position. Vaswani et al. propose an additive trigonometric position encoding, which is problematic in the case of acoustic modeling because our inputs are fixed speech features rather than flexibly learned word embeddings [VSP+17]. While concatenating positional embeddings instead provides some remedy, we find it necessary to design a hybrid self-attention/RNN architecture to obtain good results.

The third challenge is effective modeling of context relevance. Speech frames contain much less information than words and it is therefore more difficult to estimate the importance of pairs of frames with respect to each other. Based on the intuition that locality of context plays a special role in acoustic modeling, we propose to apply diagonal Gaussian masks with a trainable shape parameter to attention heads. This gives attention heads more control over context relevance and improves word error

rates consistently, by up to 1.59%. We observe that while bottom layer attention heads converge toward diversity in context range, higher layers use long-range context.

Attention mechanisms improve upon the often criticized poor interpretability of neural end-to-end models because they enforce an explicit expression of dependencies. Our self-attentional model brings this interpretability inside the encoder, making it possible to examine how speech is encoded before making the final recognition decisions. An analysis (Section 4.2.6) reveals that different attention heads measure similarity along different linguistically plausible dimensions such as phoneme clusters, indicating that they function in part to reduce acoustic variability by establishing more representative averaged forms of matching acoustic events across the utterance.

4.2.2 Basic Self-Attentional Acoustic Model

Self-attention is applied to a sequence of state vectors and transforms each state into a weighted average over all the states in the sequence, with more relevant states being given more influence. The underlying intuition is that states at each time step should be conditioned on the most relevant states across the whole sequence. Our basic form of self-attention follows Vaswani et al. [VSP+17], where relevance is measured by computing dot product similarity after applying a linear projection to both vectors. For acoustic sequences, neighboring frames are naturally similar if they represent parts of the same acoustic event. When an event with similar acoustic characteristics appears at different places in an utterance, those occurrences would be deemed relevant, as well. Following [VSP+17] we use 8 attention heads where each head can compute this similarity independently.

Our model is specifically computed as follows (Figure 4.3):

$$Q_i = XW_i^{\mathrm{Q}} \tag{4.6}$$

$$K_i = XW_i^{\mathrm{K}} \tag{4.7}$$

$$V_i = XW_i^{\mathrm{V}} \tag{4.8}$$

$$\mathrm{head}_i = \mathrm{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d}})V_i \quad \forall i \tag{4.9}$$

$$\mathrm{MultiHeadAtt} = \mathrm{concat}(\mathrm{head}_1, \mathrm{head}_2, \ldots) \tag{4.10}$$

$$\mathrm{MidLayer} = \mathrm{LayerNorm}\left[\mathrm{MultiHeadAtt} + X\right] \tag{4.11}$$

$$\mathrm{SAL} = \mathrm{LayerNorm}\left[\mathrm{FF}\left(\mathrm{MidLayer}\right) + \mathrm{MidLayer}\right] \tag{4.12}$$

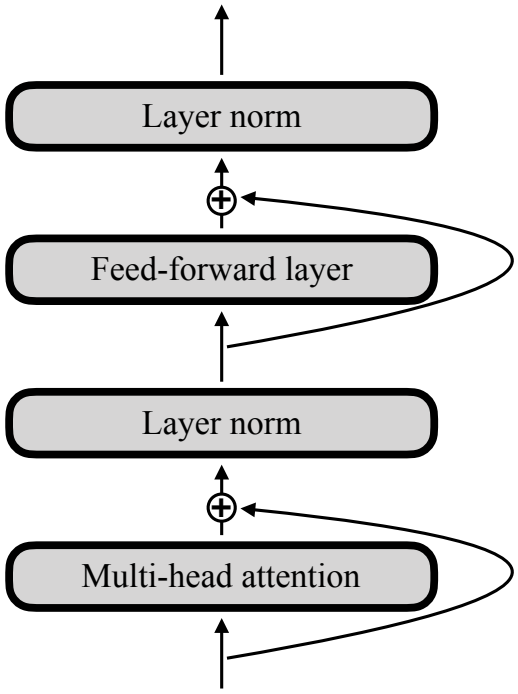


Figure 4.3: Block diagram of the core self-attentional encoder model. Note that a typical model will stack several of such blocks.

Here, $X \in \mathbb{R}^{l \times d}, Q_i, K_i, V_i \in \mathbb{R}^{l \times d/n}$ denote inputs and their query/key/value transformations for attention heads indexed by $i \in \{1, \cdots, 8\}$, sequence length l, and hidden dimension d. SAL denotes the final output of the self attention layer. $W_i^{\mathrm{Q}}, W_i^{\mathrm{K}}, W_i^{\mathrm{V}} \in \mathbb{R}^{d \times d/n}$ are parameter matrices. LayerNorm is according to [BKH16]. FF is a position-wise feed-forward network intended to introduce additional depth and nonlinearities, defined as

$$\mathrm{FF}(x) = \max\left(0, xW_1 + b_1\right) W_2 + b_2.$$

4.2.3 Tailoring Self-Attention to Speech

4.2.3.1 Downsampling

To introduce downsampling so that the model described in Section 4.2.2 fits in memory, we apply a tensor reshaping operation before every self-attention block. This reduces the sequence length by a factor of a and increases the vector state dimension by the same factor:

$$X \in \mathbb{R}^{l \times d} \rightarrow_{\text{reshape}} \hat{X} \in \mathbb{R}^{\frac{l}{a} \times ad}$$

We then compute (4.6) through (4.12) as before, with the shape of weight matrices adjusted to $W_i^{\mathrm{Q}}, W_i^{\mathrm{K}}, W_i^{\mathrm{V}} \in \mathbb{R}^{da \times d/n}$. This reduces the memory consumption of the attention matrix by factor a^2. It is crucial to apply reshapes also before the bottom layer so that the large bottom attention matrix is scaled down. Note that this approach is very similar to downsampling as in the pyramidal LSTM, except that it is applied to a sequence feature matrix instead of per-timestep, and also applied before the bottom layer.

4.2.3.2 Position Modeling

Position information is crucial in sequence-to-sequence models, but self-attention is completely agnostic to sequence positions. Prior works added trigonometric position encodings [VSP+17] or learned position embeddings [GAG+17] to input vectors, but we found that this approach does not work well for acoustic sequences. This is intuitive, as the inputs are fixed feature vectors rather than trainable word embeddings, making it difficult for the model to separate position and content for each state.

Concatenated Position Representation A straight-forward solution to enable separation of position and content for fixed inputs is to concatenate position representation instead of using a sum. We explore three variants: First, concatenating trigonometric encodings [VSP+17] to the input feature vectors. Second, concatenating learned embeddings [GAG+17] to inputs. Third, concatenating separately learned position embeddings to the queries and keys (Q,K in Equations 4.6 and 4.7) so that the key and query position can be taken into account when computing relevance at each layer.

Hybrid Models RNNs are effective at keeping track of positional information. We can exploit this by introducing recurrent layers into our encoder. We explore two alternatives:

Stacked hybrid model. Here, we stack 2 LSTM/NiN blocks (Fig. 4.1) without downsampling, followed by a final LSTM, on top of our self-attention layers. This approach does not make the self-attention layers themselves position-aware, but the final encoder states are position-aware. Reversing the order of self-attention and LSTM/NiN is also conceivable but would compromise speed because slow recurrent computations are applied before downsampling.

Interleaved hybrid model. Another option is to replace the feed-forward operation (FF in Equation 4.12) by an LSTM. Note that this introduces LSTMs before the sequence is fully downsampled and therefore compromises some of the speed gains. On the other hand, it allows the higher self-attention layers to take advantage of position information encoded by lower interleaved LSTMs.

4.2.3.3 Attention Biasing

Self-attention allows direct conditioning on the whole sequence, but it is unclear to what extent this is beneficial for our acoustic model. While context required to model polyphones may span only a relatively small temporal window, remaining channel and speaker properties may require long-range context. To account for the special role of context locality in acoustic modeling, we introduce an explicit way of controlling the context range by using a bias matrix $M \in \mathbb{R}^{l \times l}$ and computing

$$\text{head}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d}} + M)V_i.$$

By setting values around the diagonal of this mask to a higher value, we can bias the self-attention toward attending in a local range around each frame.

Local Masking We can apply hard masking by setting M as an inversely banded matrix of bandwith $b \in \mathbb{N}_{\text{odd}}$ with

$$M_{jk} = \begin{cases} 0 & |j-k| < \frac{b}{2} \\ \text{-}\infty & \text{else} \end{cases}.$$

As a result, all attention weights outside the band are set to 0, so that the self-attention is restricted to a local region of size b. The hyperparameter b can be set prior to training such that the model effectively attends to a range similar to polyphone context in hidden Markov models.

Gaussian Bias For more flexibility, we use a soft Gaussian mask by defining

$$M_{jk} = \frac{-(j-k)^2}{2\sigma^2}.$$

σ is a trainable standard deviation parameter. It is learned separately for each attention head, so that the context range can differ between attention heads. Note that compared to the standard formulation of a Gaussian curve the exponential

function and the normalizer term are dropped, as these are already accounted for by the surrounding softmax operator that this term appears inside. Besides more modeling expressiveness, the learned variances can also be inspected and may help us to understand and interpret the model.[1]

4.2.4 Experimental Setup

We focus our experiments on the TEDLIUM corpus (Section 3.2), with the development split used as validation data. In addition to the basic settings described in Section 4.1.1, the model uses variational recurrent dropout with probability 0.2 and target character dropout with probability 0.1 [GG16]. We apply label smoothing [SVI+16] and fix the target embedding norm to 1 [NC18]. For inference, we use a beam size of 20 and length normalization with exponent 1.5. Self-attention layers use a hidden dimension of 256 and feed-forward dimension of 256, and attention dropout with probability 0.2. When LSTMs are part of the encoder, we use bidirectional LSTMs with 256 hidden units per direction. Concatenated position representation vectors are of size 40. We set the batch size dynamically, with the average set to 24 (18 for LSTM-free models).

The vocabulary consists of the 26 English characters, apostrophe, whitespace, and special start-of-sequence and unknown-character tokens.

4.2.5 Quantitative Results

4.2.5.1 Comparison to Baselines

The first set of experiments compares the proposed hybrid models to the baselines. The results are summarized in Table 4.4. We observe similar word error rates, with the interleaved model outperforming the stacked model and outperforming the pyramidal LSTM baseline on the development data but not the test data. The LSTM/NiN baseline was strongest. In terms of training speed, the stacked model is fastest by a large margin, followed by the interleaved model and the LSTM/NiN model. To confirm that the attention mechanism is actually contributing to the hybrid model and not just passing on activations, we performed a sanity check by training a stacked hybrid model with attention scores off the diagonal set to $-\infty$, and observed a drop of 1.25% absolute WER.

[1] To overcome trainability issues and encourage the optimizer to adjust the variance parameter, we found it necessary to re-parametrize it using $\tau^2 = \sigma$ and optimize τ via back-propagation.

Model	Dev WER	Test WER	Training speed (char/sec)
Pyramidal	15.83	16.16	1.1k
LSTM/NiN	14.57	14.70	1.1k
Stacked hybrid	16.38	17.48	2.4k
Interleaved hybrid	15.29	16.71	1.5k

Table 4.4: Accuracy and speed of conventional and self-attentional acoustic encoders. Training speed was measured on a GTX 1080 Ti GPU.

4.2.5.2 Position Modeling

Next, we evaluate the different approaches to position modeling (Section 4.2.3.2). The results are summarized in Table 4.5. When using additive positional encodings the model diverged, while concatenating embeddings converged, albeit to rather poor optima. The key/query positional embeddings in isolation diverged, and combination with concatenated input embeddings did not improve results. Only the hybrid models were able to obtain results comparable to the baselines. We also tried combining hybrid models with positional embeddings, but did not see improvements over the model without positional embeddings.

Model	Dev WER	Test WER
Additive (trigonometric)	diverged	
Concatenative (trigonometric)	30.27	38.60
Concatenative (embedding)	29.81	31.74
Stacked hybrid	16.38	17.48
Interleaved hybrid	15.29	16.71

Table 4.5: Results on position modeling for self-attentional acoustic encoder.

4.2.5.3 Attention Biasing

This set of experiments tests the effect of introducing explicit attention biases that enable the model to control its context range (Section 4.2.3.3). The local diagonal mask was set to constrain the context to a window of 5 time steps, and Gaussian biasing variances were initialized to 9 (small setting) or 100 (large setting). Results are summarized in Table 4.6. For the stacked model, it can be seen that the biasing helps in general. The strongest model variant was the learnable Gaussian mask. Interestingly, it was important to initialize the Gaussian to possess a large variance. We hypothesize that this improves gradient flow early on in the model training, similar

to how initializing LSTM forget gate biases to 1 (no forgetting) improves results [JZS15]. The interleaved hybrid model shows similar trends. Note that the sometimes inconsistent ordering between development and test set results can be explained by the fact that the TEDLIUM development set is relatively small with only five hundred utterances.

Model	Dev WER	Test WER
Stacked hybrid	16.38	17.48
+ Local masking	15.42	16.17
+ Gauss mask (initially small)	16.05	16.96
+ Gauss mask (initially large)	14.90	15.89
Interleaved hybrid	15.29	16.71
+ Local masking	15.44	16.19
+ Gauss mask (initially small)	16.43	16.89
+ Gauss mask (initially large)	15.00	15.82

Table 4.6: Results on self-attentional acoustic encoder with attention biasing.

The Gaussian mask allows inspecting its trainable variance parameter. Fig. 4.4 shows how the parameter evolves when initialized to a large value. It can be seen that in the first layer, diversity seems to be desirable, with some attention heads focusing on a small local context, and others on larger contexts. In contrast, the second layer does not appear to benefit from limiting its context. This partly confirms the idea of hierarchical modeling, where the modeling granularity increases across layers, but also shows that even at the bottom layer a controlled amount of long-range context is desirable.

4.2.6 Interpretability of Attention Heads

We hypothesize that certain attention heads respond to certain types of acoustic events. To test this hypothesis, we correlate the average attention that each attention head places on frames with the corresponding phoneme labels obtained via forced decoding. We re-train the stacked hybrid model with phonemes instead of characters as targets, and use encoder-decoder attention scores, summed over phoneme types, to obtain a soft alignment of phoneme labels for each frame. This gives us a measure for how much each frame in the sequence corresponds to the phoneme type under inspection.

We now correlate these phoneme activations to each of the first layer's 8 attention heads. We average the matrices across rows to obtain the overall attention that each

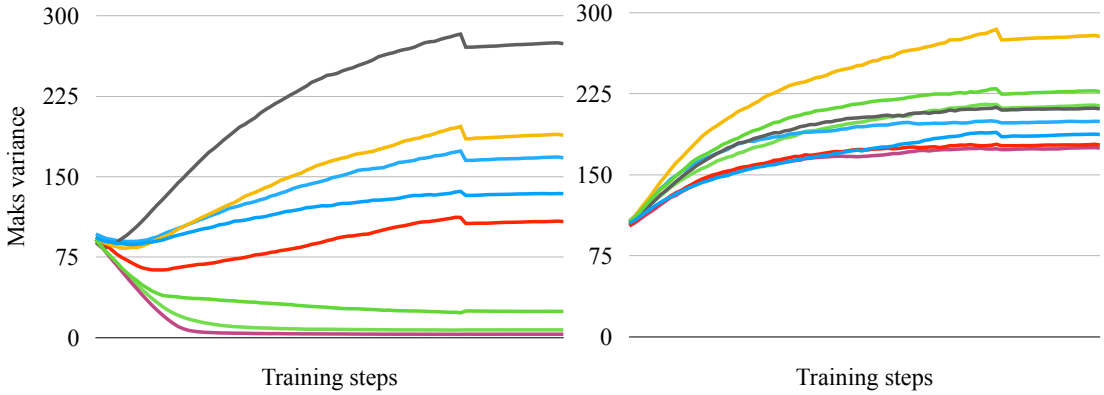


Figure 4.4: Evolution of Gaussian mask in self-attentional encoder. Shown are the variance parameters for each of the 8 attention heads over course of training (left: first layer, right: second layer). As a result of downsampling, each time step corresponds to 2 frames in the first layer and 4 frames in the second layer. Note the somewhat peculiar bump across all heads toward the end of the training, caused by learning rate decay in combination with reverting of the model to the best checkpoint.

frame receives. We then compute the Pearson correlation coefficient of the summarized self-attention and encoder-decoder attention sequences, concatenated over utterances.

Table 4.7 shows the most highly correlated phonemes for each attention head, along with an attempt to classify these manually according to linguistic categories. This works remarkably well and we can clearly see a linguistically plausible division of labor, even though categories are neither exhaustive nor disjunct. Notice that head 2 seems to always focus on the utterance end where we usually expect silence, and head 8 is mostly unfocused, which we may interpret as these heads establishing channel and speaker context.

4.2.7 Related Work

The attention mechanism was popularized by work on visual attention [MHGK14] and enabled the break-through of sequence-to-sequence models [BCB15]. Self-attention has been explored in subsequent work all in the NLP domain [CDL16, PTDU16, LFdS$^+$17, VSP$^+$17].

Binary masks have been employed for various purposes in prior work, such as preventing a self-attentional decoder to attend to the future during training [VSP$^+$17] and to implement directional attention [SZL$^+$18]. Our proposed Gaussian masking approach bears some resemblance to prior work on self-attention [IC17] that uses a *linear* distance map instead of a Gaussian and does not include trainable parameters,

i	Top phonemes	Entropy	Comments
1	S, TH, Z	3.7	sibilants
2	</s>	1.9	silence
3	UW, Y, IY, IX B, G, D M, NG, N	3.6	”you” diphthong voiced plosives nasals
4	XM, AW, AA, AY, L, AO, AH	3.2	A, schwa
5	ZH, AXR, R	3.5	R, ZH
6	ZH, Z, S IY, IH, Y, UW	3.2	sibilants ”you” diphthong
7	S, </s>, TH, CH, SH, F	3.4	fricative, noise
8	mixed	3.7	unfocused

Table 4.7: Linguistic function of self-attention heads in acoustic encoder. Note that we conducted a small amount of cherry picking by removing 4 outliers that did not seem to fit categories (OY from head 1, ZH from head 3, EH and ER from head 7). Entropy is computed over the correlation scores, truncated below 0.

so that their model is less flexible and interpretable than our approach. Both binary masked regions and Gaussian masking has also been employed in the context of tailoring the encoder-decoder cross attention to the ASR task, the former constraining an attention mechanism to attend to a local region around the previous prediction [BCS+16], the latter to define smooth attention regions for conducting monotonic attention [TSN17]. Both treat the attention width as a fixed hyperparameter. Learnable Gaussian windows have further been explored in the context of time-delay neural networks [BW91].

Some contemporaneous work investigated improvements to the positional encoding by integrating relative positions into self-attention [SUV18]. Exploring a similar idea for self-attentional acoustic models would be an interesting venue for future work, although relative position information increases the memory requirements further and may therefore not be straight-forwardly applicable to this task.

Beyond the NLP domain, there has been contemporaneous work on self-attention for acoustic modeling [PHG+18] which also employs a hybrid RNN/self-attention architecture and employs a masking technique similar to our more basic local masking variant (Section 4.2.3.3). Other contemporaneous work also moves beyond the NLP domain by exploring self-attention for graph-models [VCC+18].

4.3 Chapter Summary

In this chapter we identified the development of a strong all-neural speech recognition system as a necessary first step for advancing the state-of-the-art in speech translation. We first reported our steps taken to produce and analyze such a system using available recent modeling techniques. We then explored a novel approach of using self-attention, which has been found very powerful for text translation tasks, for acoustic modeling. Adopting self-attention to acoustic modeling is challenging for computational and modeling reasons. We investigate ways to address these challenges and obtain our best results when using a hybrid model architecture and Gaussian biases that allow controlling context range. This model is almost as good as a strong LSTM-based baseline at much faster computation speed. We highlight interpretability as an advantage over conventional models. Future work may investigate self-attentional models for other sequences of low-information states such as characters, and attempt transferring results on controlling context range and interpretability to text modeling.

Chapter 5

Tight Coupling in Cascaded Systems

On our quest toward end-to-end speech translation models, we now examine two strategies in which encoder-decoder models can be extended to achieve tight integration in a cascaded speech translation scenario. The cascaded scenario can be formally described as a decomposition

$$Pr(T \mid X) = \sum_{S \in \mathcal{H}} Pr(T \mid S, X) Pr(S \mid X) \approx \sum_{S \in \mathcal{H}} p_{\text{mt}}(T \mid S) p_{\text{asr}}(S \mid X).$$

As before, T is the desired translation, X the audio signal, S the source side transcript, and $\mathcal{H}$ denotes the speech recognizer's hypothesis space. According to the cascading scenario, the task is decomposed into separately trained recognition and translation models, p_{asr} and p_{mt}. Note also that we made a reasonable but not always correct independence assumption between the translation and the acoustic signal given the source-side transcript. Exhausting the whole transcript hypothesis space, as indicated by the sum operator, is computationally prohibitive and leads to the question of how to find suitable approximations.

In this chapter, we assume a cascaded scenario in which the output of a speech recognizer is given and needs to be translated as accurately as possible. Our first proposed innovation aims at making the translation component in a cascade more robust toward errors in the speech recognizer output. In this case, $\mathcal{H}$ is approximated in its simplest form, considering only the first-best ASR output S'. The aim is a model $p_{\text{robust}}(T \mid S')$ that minimizes the compounding effect of recognition errors. The result will be a very simple method that can be easily applied in many situations. The second

innovation aims at a tight coupling between ASR and MT by using word lattices as a powerful interface that allows the translation model to peek into the hypothesis space of the ASR system. In other words, we approximate $\mathcal{H}$ by pruned speech recognition lattices $\mathcal{L}$, and then train a modified encoder-decoder model $p_{\text{lat2seq}}(T \mid \mathcal{L})$ to directly translate these rich lattice inputs.

5.1 Robust Neural Machine Translation

A speech translation cascade requires applying a machine translation model on erroneous outputs from an automatic speech recognizer. Ideally, we would like the translation process to ignore or even to correct the corrupted inputs. Translation models are usually trained on well-formed parallel sentences that do not exhibit such noise. This results in a harmful mismatch between training and test data, and further aggravates the difficulty of transforming malformed inputs in the first place. The now prevalent encoder-decoder models [KB13, SVL14, BCB15] have been identified to be especially sensitive to noisy data [CKF+16, HNvG17, BB18], and more specifically to corrupted inputs due to erroneous ASR [RDBF17].

Robustness at test-time may be improved by inducing suitable forms of noise during the training process. The spectrum of suitable approaches ranges from general-purpose regularizers,[1] such as dropout [SHK+14], to task-specific approaches that alter the training data to resemble the corrupted inputs at test-time. Task-specific approaches can make stronger assumptions about the data distribution and are potentially more effective or provide additive gains when combined with general-purpose methods. As a disadvantage, they are also more complex and may require task-specific knowledge or resources. Another tradeoff to consider concerns trainability. Neural sequence-to-sequence models are known to suffer from explaining-away effects, where models may learn to generate outputs by relying on the target-side context while ignoring the source-side context [YBD+17, Koe17], especially when the source side provides only a weak or noisy signal. As a result, the careful calibration of type and amount of the induced noise may be necessary.

One way of inducing speech-translation-specific noise is to train on actual ASR outputs paired with the correct translations. Unfortunately, such data is scarce, and exploiting it may not be straightforward (see [PKL+13] and Section 5.1.2.1; but

[1] In this context, we use the notions of good generalization (avoiding overfitting, e.g. via regularization) and robustness (stability w.r.t. noisy data) loosely interchangeably. In fact, both are strongly linked in the sense that in general, good generalization implies robustness [CMX11].

[CHHL17]). Alternatively, it has been proposed to synthesize realistic ASR error patterns and suitable translations thereof, and augment the training data accordingly [TMD14, RGLF15]. However, this approach has not yet been shown to transfer to neural machine translation (see discussion in Section 5.1.2.4), and is relatively complex, requiring the availability of resources such as pronunciation dictionaries and suitable language models.

In this section, we seek to improve the robustness of a neural machine translation model applied to speech recognition input by exploring tradeoffs between general-purpose and task-specific methods. For this purpose, we introduce a simple noise model that is inspired by the word error rate which categorizes the common ASR error types into substitutions, insertions, and deletions. Accordingly, our noise model artificially corrupts the source side of a parallel training corpus by randomly introducing substitutions, insertions, or deletions. Our noise model is simpler than the prior approaches [TMD14, RGLF15], but nonetheless effective, and provides a flexible test bed that allows exploring the middle ground between task specificity and generality in the context of neural sequence-to-sequence models. In addition, we discuss preliminary efforts toward refining the noise model to capture more task-specific intuitions similar to these prior approaches.

Our approach is methodologically inspired by reward-augmented maximum likelihood (RAML) [NBC+16]. We use a similar sampling procedure on the source side, instead of the target side as in RAML. However, RAML is very differently motivated, aiming at fixing exposure bias whereas we are concerned with noise from upstream components. Moreover, sampling according to RAML is biased toward producing fewer deletions than substitutions and insertions, which our noise model purposefully avoids.

5.1.1 Noised Sequence-to-Sequence Training

This section proposes our noise model that will be applied to every input sentence of the training data. The general idea follows the intuitions behind the WER, according to which ASR errors can be categorized into substitutions, insertions, and deletions. Design goals are flexibility to capture various levels of refinement, and convenient control of the amount of noise and other properties. We first describe the vanilla model, and then present several refinements.

5.1.1.1 Vanilla Noise Model

The vanilla noise model, outlined in Algorithm 1, can be summarized as follows. For each sentence, we first decide on the number of edits, while considering the desired amount of overall noise. The edits are then randomly divided into substitutions, insertions and deletions. Finally, for each edit a position is randomly chosen along with a new word for substitutions and deletions.

More formally, let hyperparameter $\tau \in [0, 1]$ denote the amount of noise to be induced, let V be a sampling vocabulary, and assume a sentence of length n as $\langle w_0{=}\text{sos}, w_1, \cdots, w_n, w_{n+1}{=}\text{eos}\rangle$. We first draw the number of edits e (line 1). The Poisson distribution is a suitable choice because it is defined over non-negative integers and has probability mass centered around its mean. For simplicity, we allow a maximum of n edits for a sentence of length n. Thus, we sample according to a n-truncated Poisson distribution [Moo52], defined as

$$p_\lambda(k) \propto \exp(-\lambda)\frac{\lambda^k}{k!}$$

with support $k \in \{0, \cdots, n\}$, where we set $\lambda \coloneqq \tau \cdot n$. The mean of this distribution is approximately λ. Because of the finite support, this distribution reduces to a categorical distribution and is thus trivial to sample from.

Next, we draw the number of substitutions n_s, number of insertions n_i, and number of deletions n_d such that $n_\text{s}{+}n_\text{i}{+}n_\text{d} = e$ and $n_\text{s}, n_\text{i}, n_\text{d} \in \mathbb{N}^0$ (line 2). This defines a space over $\langle n_\text{s}, n_\text{i}, n_\text{d}\rangle$, known as the discrete 3-simplex [Cos71]. We sample from a uniform distribution over this space (Section 5.1.1.1).

We then draw without replacement a position for each substitution, insertion, and deletion (lines 3, 4, 5). Finally, we corrupt the original sentence accordingly (lines 6 through 16), sampling new words for substitutions and insertions uniformly from the sampling vocabulary (lines 7 and 14).

Sampling from the Discrete Simplex In order to determine the number of edit operations $n_1, \cdots, n_d$ for each operation type (here: $n_\text{s}, n_\text{i}, n_\text{d}$, corresponding to substitutions, insertions, and deletions), we uniformly sample

$$\langle n_1, \cdots, n_d\rangle \sim \text{DiscrSimplex}\,(d, e)$$

such that $\sum_{i=1}^{d} n_d = e$ and $n_i \in \mathbb{N}^0$. This can be accomplished by slightly adjusting the sampling approach for the continuous simplex [ST04] to the discrete

Algorithm 1 Vanilla Noise Model.
– given magnitude of noise: $\tau \in [0, 1]$
– given sentence $\langle w_0{=}\text{sos}, w_1, \cdots, w_n, w_{n+1}{=}\text{eos}\rangle$
– given vocabulary V

sample distance $e \sim \text{TruncPoisson}\,(\tau \cdot n, n)$
sample $\langle n_\text{s}, n_\text{i}, n_\text{d}\rangle \sim \text{DiscrSimplex}\,(3, e)$
sample substitution positions $s_1, \cdots, s_{n_\text{s}}$ uniformly without replacement from $\{1, \cdots, n\}$
sample insertion positions $i_1, \cdots, i_{n_\text{i}}$ uniformly without replacement from $\{0, \cdots, n\}$
sample deletion positions $d_1, \cdots, d_{n_\text{d}}$ uniformly without replacement from $\{1, \cdots, n\} \setminus \{s_1, \cdots, s_{n_s}\}$
for $i \leftarrow 1 \cdots n_\text{s}$ **do**
    uniformly sample $\tilde{w} \sim V$
    replace $w_i \leftarrow \tilde{w}$ ▷ substitution
end for
for $i \leftarrow 1 \cdots n_\text{d}$ **do**
    replace $w_i \leftarrow \epsilon$ ▷ deletion
end for
for $i \leftarrow n_\text{i} \cdots 1$ **do**
    uniformly sample $\tilde{w} \sim V$
    insert $\tilde{w}$ between w_i and w_{i+1} ▷ insertion
end for

simplex as follows. Sample auxiliary random variables $x_1, \cdots, x_{d-1}$ uniformly without replacement from $\{1, 2, \cdots, e+d-1\}$. Let $x_0{=}0, x_d{=}e{+}d$. Finally, let $n_i = x_i{-}x_{i-1}{-}1, \forall i{\in}\{1, 2, \ldots, d\}$. Proof of correctness directly follows argumentation in [ST04].

5.1.1.2 Refinements

The following discusses several simple steps, all aiming at making the sampled noise more similar to the ASR outputs. For more elaborate refinements, we refer to prior work [TMD14, RGLF15].

Sampling Vocabulary: Linguistic Conditioning The vanilla model draws substitutions and insertions uniformly from the vocabulary (lines 7 and 14), causing a large portion of induced noise to be drawn from the long tail of rarely occurring words. As a more linguistically informed strategy, we can draw from a unigram instead of a uniform distribution over the vocabulary, replacing lines 7 and 14 accordingly.

Sampling Vocabulary: Acoustic Conditioning Preferably, substitutions would be chosen based on acoustic similarity to the original input. Here, we use negative character edit distance as an approximation for acoustic similarity, and sample according to exponentiated distances $p(\tilde{w} \mid w) \propto \exp(-\text{dist}(w, \tilde{w}))$, replacing lines 7 and 14.

Sampling Positions ASR tends to err more often for certain types of words than others. For example, shorter tend to be confused more often because these words can suffer from linguistic and acoustic ambiguity. We can model this by substituting or deleting short words more often, again working with an exponentiated distribution $p(\text{pos} = j) \propto \exp(-|w_j|)$ (lines 3 and 5).

Proportion of Error Types ASR usually produces more substitutions than insertions and deletions. We may wish to reflect this in our noise distribution, for example by drawing edit operations from a 7-simplex and assigning 1 bucket to insertions, 1 bucket to deletions, and 5 buckets to substitutions[1] (lines 1 and 2).

[1]This particular choice of distribution is motivated by our experimental data containing about 5 times as many substitutions as insertions or deletions.

5.1.2 Experiments

We conduct experiments on Fisher-Callhome (Section 3.3). We use only the 138,819 training sentences of the Fisher/train part of the corpus and omit the much smaller Callhome/Train. We use Fisher/Dev as held-out testing data for most of our experiments, which has a WER of 41.3%.

For preprocessing, we tokenized and lowercased source and target sides. We removed punctuation from the reference transcripts on the source side for consistency with the automatic transcripts which also do not contain punctuation. Although punctuation is removed, we use the manual segmentation as given in the corpus, and leave dealing with noisy segmentation boundaries to future work. Our source-side vocabulary contains all words from the automatic transcripts for Fisher/Train, replacing singletons by an unknown word token, totaling 14,648 words. Similarly, on the target side we used all words from the reference translations of Fisher/Train, replacing singletons by the unknown word, yielding 10,800 words in total.

We use a standard attentional encoder-decoder architecture with one encoder and decoder layer. Basic training settings are given in Section 4.1.1. We used 128-dimensional word embeddings. We use variational dropout [GG16] in encoder and decoder LSTMs (p=0.5). To obtain a more noise-robust baseline, we also apply word type dropout [GG16] to the source word embeddings (p=0.1).

For all experiments, we first pretrained a model using reference transcripts only, starting from an initial learning rate of 0.0003, restarting Adam and halving learning rates when perplexities did not improve for 2 consecutive epochs [DN17]. We then fine-tuned the model weights by training on noisy data according to the proposed noise model. Fine-tuning used an initial learning rate of 0.00001 and the same learning rate decay and restarting strategy as during pretraining. The pretraining-finetuning scheme was used in part to make experimental effort manageable, and in part because we observed better BLEU scores in preliminary experiments.

5.1.2.1 Main Results

Figure 5.1 compares our baseline model against several models trained using our noise model. `VanillaNoise` induces varying amounts of noise using the basic model and yields substantial improvements over the `Baseline`, which is trained only on clean data. `UnigramNoise` replaces the uniform sampling distribution with a unigram distribution and yields similar gains. Perhaps surprisingly, `DeletionNoise`, a simplified model that induces only deletions, produces strong results as well. We present a possible

explanation later. Note that improvements are achieved only for small to moderate amounts of noise. For $\tau = 0.4$, which is close to the WER of the test data, results are rather poor. This indicates that we are facing a trade-off between better trainability for small values for τ, and better distributional similarity with the test data for higher values for τ. We also trained a model by fine-tuning on actual 1-best transcripts rather than using the proposed noise model. Results are rather poor at 32.55 BLEU points, which may be explained by the amount of noise being so high that trainability is compromised, and possibly by some proneness to overfitting because the same noise is used in every epoch.

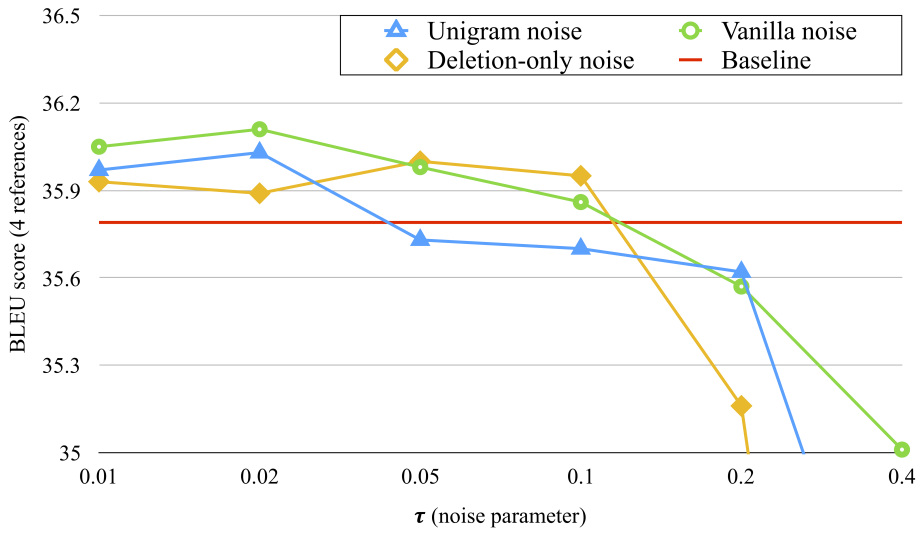


Figure 5.1: BLEU scores for noised training with inputs from ASR. We evaluate on Fisher/Dev with 4 reference translations and vary the amount of the induced training noise.

Figure 5.2 shows the performance of the same models when using clean reference transcripts as inputs. Translation of clean inputs is improved for one configuration of inducing noise, in which case the induced noise can be understood to act as a general-purpose regularizer.[1] However, note that the performance drops quickly when increasing the noise parameter τ, again highlighting both the importance of distributional similarity between training and test data, and potential trainability issues.

Figure 5.3 evaluates models in terms of n-gram precision, which we compute identically to the BLEU score but drop the brevity penalty. Comparing results to

[1] This explanation is supported by prior work relating data noising to traditional smoothing methods [XWL+17].

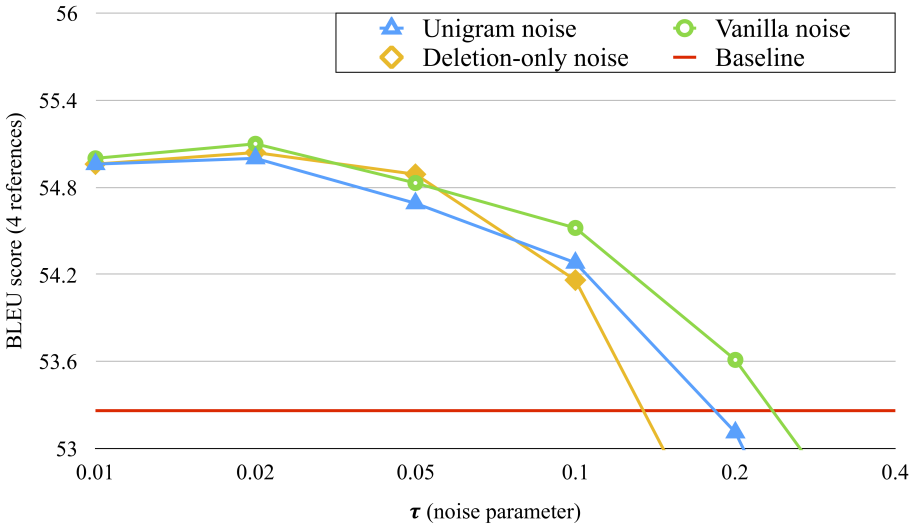


Figure 5.2: BLEU scores for noised training with clean inputs. We evaluate on Fisher/Dev with 4 reference translations and vary the amount of the induced training noise.

Figure 5.1, we can clearly observe some interactions that lead to trading off precision for recall. Most notably, `DeletionNoise` performs substantially worse than `VanillaNoise` and `UnigramNoise` when measuring only precision. Closer analysis showed that models generally tend to produce shorter outputs the more noise is contained in the inputs. The BLEU metric's brevity penalty is known to punish such short outputs quite severely. `DeletionNoise`, on the other hand, is trained on inputs where words are deleted. In other words, the training-time inputs are shorter than the test-time inputs, counteracting the tendency to produce shorter outputs and thereby avoiding a severe brevity penalty. While this helps BLEU score, arguably producing shorter outputs for noisier inputs is a desirable behavior that we would also expect from a human translator, and BLEU may thus not be a sufficient ground for model selection in our task.

5.1.2.2 Impact of ASR Quality

For this experiment, we combined all available test data (Fisher/Dev, Fisher/Dev2, Fisher/Test, Callhome/Devtest, Callhome/Evltest), and divided it into bins according to ASR WER. Figure 5.4 shows the length ratio of translations produced for these inputs for two different models. It can be seen that both `Baseline` and `UnigramNoise` produce length ratios close to 1.0 for clean inputs. However, when inputs contain even moderate amounts of noise, uncertainty in `Baseline` seems to become problematic and outputs quickly become rather short. `UnigramNoise` on the other hand appears to

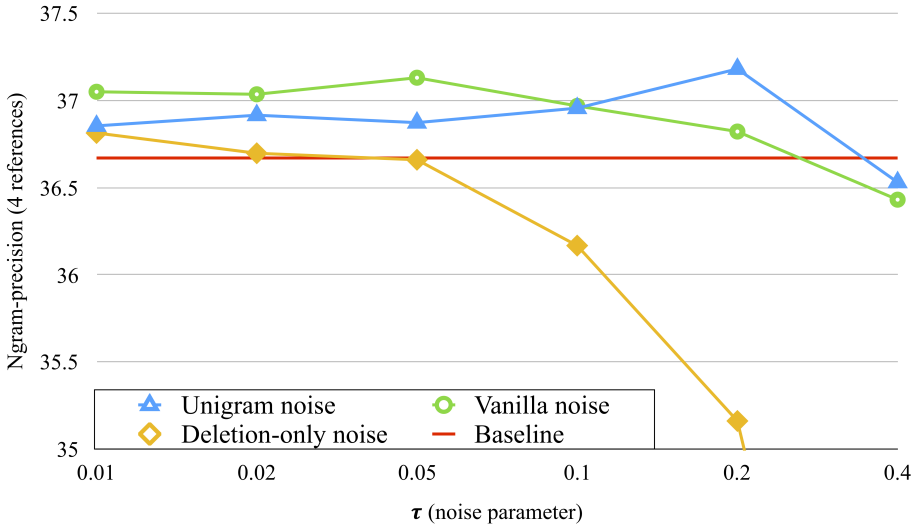


Figure 5.3: *n*-gram precision for noisy inputs over various training noise settings. We evaluate on Fisher/Dev and vary the amount of the induced training noise. *n*-gram precision is computed as BLEU score with removed brevity penalty.

handle noisier inputs much more gracefully, while also exhibiting a tendency for shorter outputs when inputs are noisy.

While this demonstrates a greater robustness of the noise-induced model, it also raises the question as to what extent shorter outputs for noisy inputs are desirable. Consider the example of a simultaneous interpreter who's job is to produce a translation of some on-going speech in real-time. Interpreters often face cognitive limits where they are either too slow to deliver a complete translation, or fail to understand the speech correctly. In such cases, trained interpreters usually resort to compensatory strategies. According to a study by Al-Khanji et al. [AKESH00], human interpreters resort to strategies that reduce preciseness or omit information about ten times as often as they substitute information with misleading content. We may view a speech translation system as similarly facing "cognitive constraints", which may arguably better be compensated for by strategies of omission rather than substitution.[1] This claim would also need to be investigated from a usability perspective, however.

[1]Consider a typical example we found in an English ASR transcript, *Boesch as ever his son decides to have a feast.* While the first 3 or 4 words are clearly recognition mistakes (caused by a rare name in the audio), the rest makes sense and a human might choose to only translate the latter part. Another example is *buildings and boundaries around the location very part*, where the last 2 words are easily recognizable as mistakes and could be dropped before translating. However, an experienced translator might guess correctly that *very part* should be replaced by *where to park*.

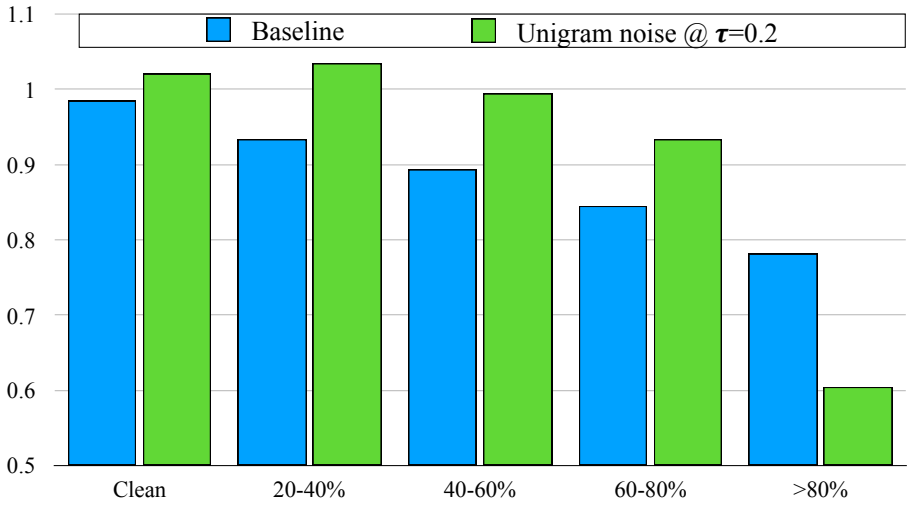


Figure 5.4: Translation length ratio for noised training against ASR accuracy. We bin the test inputs according to their WER.

5.1.2.3 Large System Results

We have used the proposed approach in two evaluation systems, a multilingual translation model submitted to IWSLT 2017 and a deep transformer model submitted to IWSLT 2018. These are large-scale systems with respect to both model size and training data size. Details can be found in the system description papers [PSS+17, SPN+18]. For simplicity, we have constrained ourselves to the deletion-only setting with $\tau = 0.01$. The results in Table 5.1 show improvements between 0.2 and 0.5 BLEU points and demonstrate that our approach scales to larger systems.

System	Original data	Noised data
tst2013 (English to German) [PSS+17]	17.9	**18.4**
tst2013 (German to English) [PSS+17]	15.7	**16.0**
tst2014 (German to English) [SPN+18]	19.1	**19.3**

Table 5.1: BLEU scores for noised training in evaluation systems.

5.1.2.4 Negative Results for Model Refinements

Our analysis so far only considered the unigram-sampling refinement of the vanilla noise model. We also tested acoustic conditioning, better sampling positions, and more realistic proportion of error types (Section 5.1.1.2), but did not observe noticeable improvements and do not explore the details here. Future work may attempt using even

more realistic error patterns along the lines of prior work [TMD14, RGLF15]. However, a possible difficulty when trying this may be that, unlike phrase-based machine translation, neural machine translation has been known to be ineffective at learning from rare training examples [Koe17]. Permutations of error patterns potentially consist mainly of such hard-to-learn rarely occurring patterns. Counteracting this by increasing the amount of noise may lead to trainability issues as observed in our experiments as well. Instead, it may be necessary to represent knowledge about confusability more explicitly and efficiently in the model.

5.1.3 Related Work

The most closely related works aim at synthesizing ASR error patterns in order to train an SMT model to translate erroneous ASR outputs. [TMD14, RGLF15]. Unlike neural models, SMT models are effective at translating even rare phrases by simply looking up their translation in the phrase table. This makes them suitable for learning many possible permutations of error patterns. We argue that this is not the case for sequence-to-sequence models, which provides an explanation for why we did not observe gains from model refinements that make our error model more similar to these prior approaches (Section 5.1.2.4).

Inducing noise in the training inputs can be seen as a form of data augmentation, which has been used in several applications such as acoustic modeling [JH13], computer vision [ZBH+17], language modeling [XWL+17], and statistical machine translation where data can be augmented by paraphrases [CBKO06]. It has been described as more powerful than general-purpose regularization in the context of deep learning [ZBH+17]. Note that these approaches aim at inducing *label-preserving* noise, in contrast to our noise model which may alter or destroy the meaning of an input despite keeping targets unchanged. Data augmentation has also been used to improve robustness to noisy inputs as in our work, such as research on speech recognition under noisy conditions [IKS+13].

Several works have identified noisy or mismatched text inputs as a challenge for neural models: [KK17] mention domain mismatch as a challenge for neural machine translation, [CKF+16] show that NMT suffers from noisy training data, Li et al. show that recurrent neural networks can be sensitive to corrupted input sequences [LMJ17].

A growing body of work is dedicated to robustness when dealing with social media content, which may include phenomena such as spelling mistakes, abbreviations, and internet slang. Prior to our work, Heigold et al. [HNvG17] analyze this issue for neural

machine translation of such data, and following our contribution several researchers have proposed additional solutions [CTM+18, BB18]. The latter work demonstrates the importance of using natural (as opposed to synthetic) noise to make models robust to realistic noisy test-time conditions. Michel et al. [MN18] have created a suitable data set and benchmark to facilitate comparability of future work.

5.2 Neural Lattice-to-Sequence Models

Previous research on traditional phrase-based or tree-based statistical machine translation has used word lattices (e.g. Figure 5.5) as an effective tool to pass on uncertainties from up-stream components [Ney99, CNO+04]. This prevents the system to commit to early decisions and allows passing alternative hypotheses between models. Several works have shown quality improvements by translating lattices, compared to translating only the single best upstream output. Examples include translating lattice representations of ASR output [SJVS04, ZKYL05, MHN08], multiple word segmentations, and morphological alternatives [DMR08].

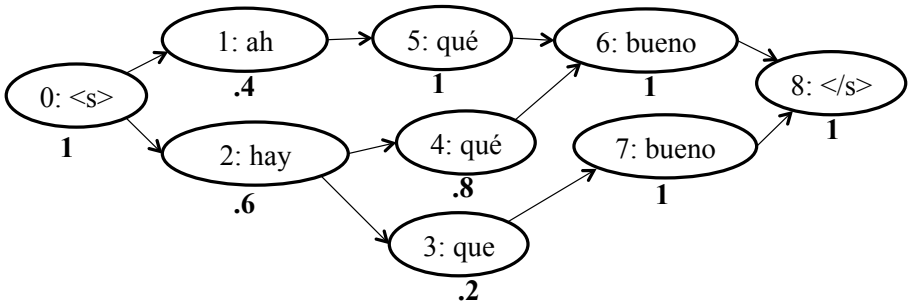


Figure 5.5: An example lattice with posterior scores. The lattice contains 3 possible paths. Translating the whole lattice potentially allows for recovering from errors in the corresponding 1-best hypothesis.

The recent neural encoder-decoder models [KB13, SVL14, BCB15] force us to rethink approaches to handling lattices, because their recurrent design no longer allows for efficient lattice decoding using dynamic programming as was used in the earlier works.

As a remedy, prior work [STX+17] proposed replacing the sequential encoder by a lattice encoder to obtain a lattice-to-sequence (`lat2seq`) model. This was achieved by extending the encoder's Gated Recurrent Units (GRUs) [CMG+14] to be conditioned on multiple predecessor paths. The authors demonstrate improvements in Chinese-to-English translation by translating lattices that combine the output of

multiple word segmenters, rather than a single segmented sequence.

However, this model does not address one key aspect of lattices that we argue is critical to obtaining good translation results: their ability to encode the certainty or uncertainty of the paths through the use of posterior scores. Specifically, we postulate that these scores are essential for tasks that require handling lattices with a considerable amount of erroneous content, such as those produced by ASR systems. In this chapter, we propose a lattice-to-sequence model that accounts for this uncertainty. Specifically, our contributions are as follows:

- We employ the popular child-sum TreeLSTM [TSM15] to derive a lattice encoder that replaces the sequential encoder in an attentional encoder-decoder model. We show empirically that this approach yields only minor improvements compared to a baseline that is fine-tuned on sequential recognition outputs. This finding stands in contrast to the positive results by Su et al. [STX+17], as well as Ladhak et al. on a lattice classification task [LGD+16], and suggests higher learning complexity of our speech translation task.

- We hypothesize that lattice scores are crucial in aiding training and inference, and propose several techniques for integrating lattice scores into the model: (1) We compute *weighted child-sums*,[1] where hidden units in the lattice encoder are conditioned on their predecessor hidden units such that predecessors with low probability are less influential on the current hidden state. (2) We bias the TreeLSTM's *forget gates* for each incoming connection toward being more forgetful for predecessors with low probability, such that their cell states become relatively less influential. (3) We *bias the attention mechanism* to put more focus on source embeddings belonging to nodes with high lattice scores. We demonstrate empirically that the third proposed technique is particularly effective and produces strong gains over the baseline. According to our knowledge, this is the first attempt of integrating lattice scores at the *training stage* of a machine translation model.

- We exploit the fact that our lattice encoder is a strict generalization of a sequential encoder by *pretraining* on sequential data, obtaining faster and better training convergence on large corpora of parallel sequential data.

[1] This is reminiscent of the weighted pooling strategy [LGD+16] for spoken utterance classification.

5.2.1 Attentional Lattice-to-Sequence Model

The encoder-decoder model described in Section 2.1 assumes sequential inputs and is therefore limited to taking a single output of an up-stream model as input. Instead, we wish to consume lattices to carry over uncertainties from an up-stream model.

5.2.1.1 Lattices

Lattices (e.g. Figure 5.5) represent multiple ambiguous or competing sequences in a compact form. They are a more efficient alternative to enumerating all hypotheses as an n-best list, as they allow for avoiding redundant computation over subsequences shared between multiple hypotheses. Lattices can either be produced directly, e.g. by an ASR dumping its pruned search space [PKL+13], or can be obtained by merging several n-best sequences [DMR08, STX+17].

A word lattice $\mathcal{G} = \langle V, E \rangle$ is a directed, connected, and acyclic graph with nodes V and edges E. $V \subset \mathbb{N}$ is a node set, and $(k, i) \in E$ denotes an edge connecting node k to node i. $C(i)$ denotes the set of predecessor nodes for node i. We assume that all nodes follow a topological ordering, such that $k < i \; \forall \; k \in C(i)$. Each node i is assigned a word label $w(i)$.[1] Every lattice contains exactly one start-of-sequence node with only outgoing edges, and exactly one end-of-sequence node with only incoming edges.

5.2.1.2 Lattices and the TreeLSTM

One thing to notice here is that lattice nodes can have multiple predecessor states. In contrast, hidden states in LSTMs and other sequential RNNs are conditioned on only one predecessor state ($\tilde{h}_j$ in left column of Table 5.2), rendering standard RNNs unsuitable for the modeling of lattices. Luckily the TreeLSTM [TSM15], which was designed to compose encodings in trees, is also straightforward to apply to lattices; the TreeLSTM composes multiple child states into a parent state, which can also be applied to lattices to compose multiple predecessor states into a successor state. Table 5.2, middle column, shows the TreeLSTM in its child-sum variant that supports an arbitrary number of predecessors. Conditioning on multiple predecessor hidden states is achieved by simply taking their sum as $\tilde{\mathbf{h}}_i$. Cell states from multiple predecessor are each passed through their own forget gates $\mathbf{f}_{jk}$ and then summed.

Encoding a lattice results in one hidden state for each lattice node. Our `lat2seq`

[1] It is perhaps more common to think of each edge representing a word, but we will motivate why we instead assign word labels to nodes in Section 5.2.1.3.

	Sequential LSTM	TreeLSTM	Proposed LatticeLSTM
recurr.	$\tilde{\mathbf{h}}_i = \mathbf{h}_{i-1}$	$\tilde{\mathbf{h}}_i = \sum_{k \in C(i)} \mathbf{h}_k$	$\tilde{\mathbf{h}}_i = \sum_{k \in C(i)} \frac{w_{b/f,k}^{\mathbf{S}_h}}{\mathbf{Z}_{h,k}} \mathbf{h}_k$ (5.2)
forget gate	$\mathbf{f}_i = \sigma\left(W_f \mathbf{x}_i + U_f \tilde{\mathbf{h}}_i + \mathbf{b}_f\right)$	$\mathbf{f}_{ik} = \sigma(W_f \mathbf{x}_i + U_f \mathbf{h}_k + \mathbf{b}_f)$	$\mathbf{f}_{ik} = \sigma(W_f \mathbf{x}_i + U_f \mathbf{h}_k + \left[\ln w_{b/f,k} \mathbf{S}_f - \mathbf{Z}_{f,k}\right] + \mathbf{b}_f)$ (5.3)
input gate	$\mathbf{i}_i = \sigma\left(W_{in} \mathbf{x}_i + U_{in} \tilde{\mathbf{h}}_i + \mathbf{b}_{in}\right)$	as sequential	as sequential
output gate	$\mathbf{o}_i = \sigma\left(W_o \mathbf{x}_i + U_o \tilde{\mathbf{h}}_i + \mathbf{b}_o\right)$	as sequential	as sequential
update	$\mathbf{u}_i = \tanh\left(W_u \mathbf{x}_i + U_u \tilde{\mathbf{h}}_i + \mathbf{b}_u\right)$	as sequential	as sequential
new cell	$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \mathbf{f}_i \odot \mathbf{c}_{i-1}$	$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \sum_{k \in C(i)} \mathbf{f}_{ik} \odot \mathbf{c}_k$	as TreeLSTM
new hidden	$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$	as sequential	as sequential
attent.	$\alpha_{ij} \propto \exp\left(s\left(\cdot\right)\right)$		$\alpha_{ij} \propto \exp\left[s\left(\cdot\right) + S_a \ln w_{m,i}\right]$ (5.4)

Table 5.2: Formulas for sequential, tree, and proposed lattice LSTMs. TreeLSTM are according to Tai et al. [TSM15]. Also shown is the conventional vs. proposed integration into the attention mechanism (bottom row). Inputs $\mathbf{x}_j$ are word embeddings or hidden states of a lower layer. *W.* and *U.* denote parameter matrices, **b**. bias terms.

framework uses this network as encoder, computing the attention over all lattice nodes.[1] In other words we replace Equation 2.1 by the following:

$$\overrightarrow{\mathbf{h}}_i = \text{LatticeLSTM}\left(\mathbf{x}_i, \{\overrightarrow{\mathbf{h}}_k \mid k \in C(i)\}\right) \tag{5.1}$$

Similarly, we encode the lattice in backward direction and replace Equation 2.2 accordingly. Figure 5.6 illustrates the result. The computational complexity of the encoder is $\mathcal{O}(|V| + |E|)$, i.e. linear in the number of nodes plus number of edges in the graph. The complexity of the attention mechanism is $\mathcal{O}(|V|M)$, where M is the output sequence length. $|V|$ depends on both the expected input sentence length and the lattice density.

5.2.1.3 Node-labeled Lattices

At this point, we take a step back to motivate our choice of assigning word labels to lattice nodes, which is in contrast to the prior work [LGD+16, STX+17] who assign

[1] This is similar in spirit to prior work [EHT16] that used the TreeLSTM in an attentional tree-to-sequence model to translate sentences given in form of a syntax tree.

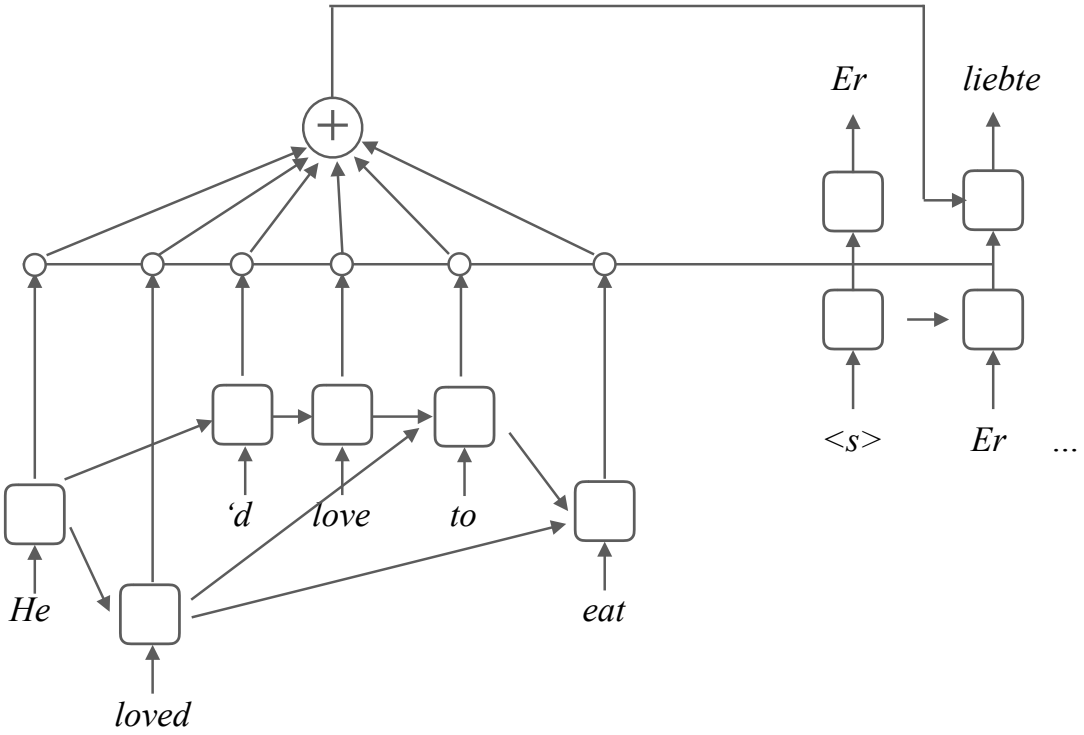


Figure 5.6: Network structure for the `lat2seq` model. For simplicity, only a single, unidirectional encoder is shown, while in practice we use bidirectional encoders with multiple layers.

word labels to edges. Recurrent states in edge-labeled lattice encoders are conditioned not only on multiple predecessor states, but must also aggregate words from multiple incoming edges. This implies that hidden units may represent more than one word in the lattice. Moreover, in the edge-labeled case the hidden units that are in the same position in the forward and backward encoders represent different words, but are nevertheless concatenated and attended to jointly. For these reasons we find our approach of encoding word-labeled lattices more intuitively appealing when used as input to an attentional decoder, although empirical justification is beyond the scope of this work. We also note that it is easy to convert an edge-labeled lattice into a node-labeled lattice using the line-graph algorithm [HB78], which we utilize in this work.

5.2.2 Integration of Lattice Scores

This section describes a key ingredient of our lattice-to-sequence model: integration of lattice scores encoding input uncertainty. These lattice scores assign different probabilities to competing paths, and are often provided by up-stream statistical models. For example, an ASR may attach posterior probabilities that capture acoustic evidence and linguistic plausibility of words in the lattice. In this section, we describe our method, first explaining how we normalize scores to a format that is easily usable in our method, then presenting our methods for incorporating these scores into our encoder calculations.

5.2.2.1 Lattice Score Normalization

Lattice scores that are obtained from upstream systems (such as ASR) are typically given in forward-normalized fashion, interpreted as the probability of a node given its predecessor. Here, outgoing edges sum up to one, as illustrated by the left-most scores in Figure 5.7. However, in some of our methods it will be necessary that scores be normalized in the backward direction, so that the weights from incoming connections sum up to one, or globally normalized, so that the probability of the node is the marginal probability of all the paths containing that node.

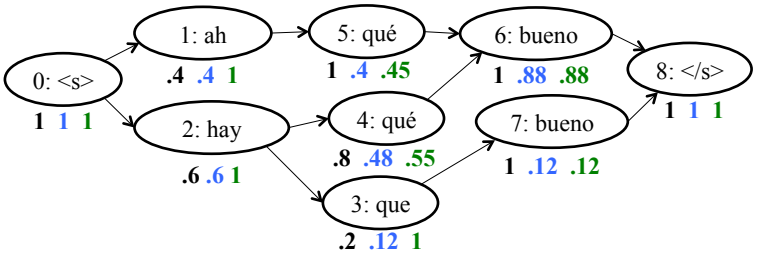


Figure 5.7: Lattice with normalized scores. The black scores are forward-normalized, blue scores are marginal scores, and green scores are backward-normalized.

Let $w_{f,i}$, $w_{m,i}$, $w_{b,i}$ denote forward-normalized, marginal, and backward-normalized scores for node i respectively, illustrated in Figure 5.7. Given $w_{f,i}$, we can compute marginal probabilities recursively by multiplying a predecessor's marginal by the forward probability to the current node and summing over all predecessors:

$$w_{m,i} = \sum_{k \in C(i)} w_{m,k} \cdot w_{f,i}.$$

This corresponds to the the forward algorithm [Rab89]. Then, we can normalize

backward using

$$w_{b,i} = \frac{w_{m,i}}{\sum_{k \in C'(i)} w_{m,k}},$$

where $C'(i)$ denotes the successors of node i. All 3 forms are employed in the sections below.

Furthermore, when integrating these scores into the `lat2seq` framework, it is desirable to maintain flexibility over how strongly they should impact the model. For this purpose, we introduce a peakiness coefficient S. Given a lattice score $w_{b,i}$ in backward direction, we compute $w_{b,i}^S / Z_i$. $Z_i = \sum_{k \in C(i)} w_k$ is a re-normalization term to ensure that incoming connections still sum up to one. In the forward direction, we compute $w_{f,i}^S / Z_i$ and normalize analogously over outgoing connections. Setting $S{=}0$ amounts to ignoring the scores by flattening their distribution, while letting $S{\to}\infty$ puts emphasis solely on the strongest nodes. S can be optimized jointly with the other model parameters via back-propagation during model training.

5.2.2.2 Integration Approaches

We suggest three methods to integrate these scores into our `lat2seq` model, with equations shown in the right column of Table 5.2. These methods can optionally be combined, and we conduct an ablation study to assess the effectivity of each method in isolation (Section 5.2.4.3).

The first method consists of computing a weighted child-sum (`WCS`), using lattice scores as weights when composing the hidden state $\tilde{\mathbf{h}}_i$. This is based on the intuition that predecessor hidden states with high lattice weights should have a higher influence on their successor than states with low weights. The precise formulas for `WCS` are shown in (5.2).

The second method biases the forget gate $\mathbf{f}_{ik}$ for each predecessor cell state such that predecessors with high lattice score are more likely to pass through the forget gate (`BFG`). The intuition for this is similar to `WCS`; the composed cell state is more highly influenced by cell states from predecessors with high lattice score. `BFG` is implemented by introducing a bias term inside the sigmoid as in (5.3).

In the cases of both `WCS` and `BFG`, all hidden units have their own independent peakiness. Thus $\mathbf{S}_h$ and $\mathbf{S}_f$ are vectors, applied element-wise after broadcasting the lattice score. The re-normalization terms $\mathbf{Z}_{h,k}$ and $\mathbf{Z}_{f,k}$ are also vectors and are applied element-wise. We use backward-normalized scores $w_{b,i}$ for the forward-directed encoder, and forward-normalized scores $w_{f,i}$ for the backward-directed encoder.

In the third and final method, we bias the attentional weights (`BATT`) to put more focus on lattice nodes with high lattice scores. This can potentially mitigate the problem of having multiple contradicting lattice nodes that may confuse the attentional decoder. `BATT` is computed by introducing a bias term to the attention as in (5.4). Attentional weights are scalars, so here the peakiness S_a is also a scalar. We drop the normalization term, relying instead on the softmax normalization. Both `BFG` and `BATT` use the logarithm of lattice scores so that values will still be in the probability domain after the softmax or sigmoid is computed.

5.2.3 Pretraining

We use a two-step training process where the model is first *pretrained* on sequential data, then *fine-tuned* on lattice data. We found this important for achieving good results, because this allowed initializing the weights based on clean translation data before fine-tuning on noisy lattices which can be difficult from a trainability perspective (Section 5.1). Moreover, the pretraining allows for efficient training using batched computation as is commonly used with sequence-to-sequence models.

The fine-tuning on parallel data with lattices on the source side can be much slower because batch computations are not trivial to exploit here, due to the changing network structure from sentence to sentence. However, auto-batching [NGD17] can be used as a remedy to make lattice training almost as fast as sequential training. In either case, we found it important to use (potentially simulated) minibatch training when fine-tuning in order to stabilize the training.

At test time, the model is able to translate both sequential and lattice inputs and can therefore be used even in cases where no lattices are available, at potentially diminished accuracy.

5.2.4 Experiments

5.2.4.1 Setting

We conduct experiments on Fisher-Callhome (Section 3.3) which includes automatic transcripts and lattices. The lattices contain on average 3.4 (Fisher/Train) or 4.5 (Callhome/Train) times more words than their corresponding reference transcripts. For preprocessing, we tokenized and lowercased source and target sides. We removed punctuation from the reference transcripts on the source side for consistency with the automatic transcripts and lattices. All models are pretrained and fine-tuned on Fisher/Train unless otherwise noted. Our source-side vocabulary contains all words

from the automatic transcripts for Fisher/Train, replacing singletons by an unknown word token, totaling 14,648 words. Similarly, on the target side we used all words from the reference translations of Fisher/Train, replacing singletons by the unknown word, yielding 10,800 words in total.

Our implementation is based on DyNet [NDG+17]. We used a standard attentional model with default parameters: a layer size of 256 per encoder direction and 512 for the decoder, word embedding size of 512. We used 2 encoder and decoder layers for better baseline performance. For the sequential baselines, the LSTM variant in the left column of Table 5.2 was employed. Forget gates were initialized to 1, following [JZS15].

We used Adam [KB14] for training, with an empirically determined initial learning rate of 0.001 for pretraining and 0.0001 for fine-tuning. We halved the learning rate when the dev perplexity (on Fisher/Dev) gets worse. Pretraining and fine-tuning on 1-best sequences were performed until convergence, and training on lattices was performed for 2 epochs.[1] Minibatch size was 1000 words for pretraining, and 20 sentences for lattice training. Unless otherwise noted, peakiness coefficients were jointly optimized during training. We repeated training 3 times with different random seeds for parameter initialization and data shuffling, and report averaged results. The decoding beam size was 5.

5.2.4.2 Main Results

We compare 4 systems: Performing pretraining on the sequential reference transcripts only (R), fine-tuning on 1-best transcripts (R+1), fine-tuning on lattices without scores (R+L), and fine-tuning on lattices including lattice scores (R+L+S). At test time, we try references, lattice oracles,[2] 1-best transcripts, and lattices as inputs to all 4 systems. The former 2 experiments give upper bounds on achievable translation accuracy, while the latter 2 correspond to a realistic setting. Table 5.3 shows the results on Fisher/Dev2 and Fisher/Test.

Before even considering lattices, we can see that 1-best fine-tuning boosted BLEU scores quite impressively (1-best/R vs. 1-best/R+1), with gains of 1.3 and 0.7 BLEU points. This stands in contrast to [PKL+13] who find the 1-best transcripts not to be helpful for training a hierarchical machine translation system. Possible explanations are learning from repeating error patterns, and improved robustness to erroneous inputs. On top of these gains, our proposed set-up (lattice/R+L+S) improves BLEU scores

[1] For comparison, we tried training on lattices from scratch, which converged at a dev perplexity that was 10% worse than with the pretraining plus fine-tuning strategy.

[2] The path through the lattice with the best WER.

Test-time	Trained on				Trained on			
inputs	R	R+1	R+L	R+L+S	R	R+1	R+L	R+L+S
Reference	53.9	53.8	53.7	54.0	52.2	51.8	52.2	52.7
Oracle	44.9	45.6	45.2	45.2	44.4	44.6	44.6	44.8
1-best	35.8	37.1	36.2	36.2	35.9	36.6	36.2	36.4
Lattice	25.9	25.8	36.9	**38.5**	26.2	25.8	36.1	**38.0**
			Fisher/Dev2				Fisher/Test	

Table 5.3: BLEU scores for lattice-to-sequence models and baselines. BLEU scores are computed using 4 references. Models are pretrained only (R), fine-tuned on either 1-best outputs (R+1), lattices without scores (R+L), or lattices with scores (R+L+S). Statistically significant improvement (paired bootstrap resampling, $p < 0.05$) over 1-best/R+1 is in bold.

Test-time	Trained on			
inputs	R	R+1	R+L	R+L+S
Reference	7.1	6.5	6.8	6.7
Oracle	13.4	9.5	10.6	10.6
1-best	24.7	13.7	16.4	16.3
Lattice	23.4	15.7	13.0	**12.6**

Table 5.4: Perplexity results for lattice-to-sequence models and baselines. Perplexities are computed on Fisher/Dev2. Models are pretrained only (R), fine-tuned on either 1-best outputs (R+1), lattices without scores (R+L), or lattices with scores (R+L+S).

by another 1.4. Removing the lattice scores (lattice/R+L) diminishes the results and performs worse than the 1-best baseline (1-best/R+1), indicating that the proposed lattice score integration is crucial for good performance. This demonstrates a clear advantage of our proposed method over that of [STX+17].[1]

As can be seen in the table, models fine-tuned on lattices show reasonable performance for both lattice and sequential inputs (1-best/R+L, lattice/R+L, 1-best/R+L+S, lattice/R+L+S). This is not surprising, given that the lattice training data includes lattices of varying density, including lattices with very few paths or even only one path. On the other hand, without fine-tuning on lattices, using lattices as input

[1] Our model makes it also possible to integrate posterior scores into sequential 1-best translation, but this did not yield any gains compared to not using the scores and is not explored further here.

performs poorly (lattice/R and lattice/R+1). A closer look revealed that translations were often too long, potentially because implicitly learned mechanisms for length control were not ready to handle lattice inputs.

Table 5.4 reports perplexities for Fisher/Dev2. Unlike the corresponding BLEU scores, the lattice encoder appears stronger than the 1-best baseline in terms of perplexity even without lattice scores (lattice/R+L vs. 1-best/R+1). To understand this better, we computed the average entropy of the decoder softmax, a measure of how much confusion there is in the decoder predictions, independent of whether it selects the correct answer or not. Over the first 100 sentences, this value was 2.24 for 1-best/R+1, 2.39 for lattice/R+L, and 2.15 for lattice/R+L+S. This indicates that the decoder is more confused for lattices without scores, while integrating lattice scores removes this problem. These numbers also suggest that it may be possible to obtain further gains using methods that stabilize the decoder.

5.2.4.3 Ablation Experiments

Next, we conduct an ablation study to assess the impact of the three proposed extension for integrating lattice scores (Section 5.2.2.2). We train models with different peakiness coefficients S, either ignoring lattices scores by fixing S=0, using lattice scores as-is by fixing S=1, or optimizing S during training. Table 5.5 shows the results. Overall, joint training of S gives similar results as fixing S=1, but both clearly outperform fixing S=0. Removing confidences (setting S=0) in one place at a time reveals that the attention mechanism is clearly the most important point of integration, while gains from the integration into child-sum and forget gate are smaller and not always consistent.

We also analyzed what peakiness values were actually learned. We found that all 3 models that we trained for the averaging purposes converged to S_a=0.62. $\mathbf{S}_h$ and $\mathbf{S}_f$ had per-vector means between 0.92 and 1.0, at standard deviations between 0.02 and 0.04. We conclude that while the peakiness coefficients were not particularly helpful in our experiments, stable convergence behavior makes them safe to use, and they might be helpful on other data sets that may contain lattice scores of higher or lower reliability.

5.2.4.4 Callhome Experiments

In this experiment, we test a situation in which we have a reasonable amount of sequential data available for pretraining, but only a limited amount of lattice training data for the fine-tuning step. This may be a more realistic situation, because speech translation corpora are scarce. To investigate in this scenario, we again pretrain our

BATT S_a	BCS $\mathbf{S}_h$	BFG $\mathbf{S}_f$	Fisher /Dev2	Fisher /Test
0	**0**	**0**	36.9	36.1
1	**1**	**1**	**38.2**	**37.4**
*	*	*	**38.5**	**38.0**
0	**1**	**1**	37.2	36.2
1	**0**	**1**	**37.9**	**37.5**
1	**1**	**0**	**38.2**	**37.8**
0	*	*	37.0	36.3
*	**0**	*	**38.3**	**37.9**
*	*	**0**	**38.1**	**37.5**
1-best/R+1			37.2	36.6

Table 5.5: Lattice-to-sequence ablation experiments. BLEU scores (4 references) are evaluated for different configurations of the peakiness coefficients $S_a, \mathbf{S}_h, \mathbf{S}_f$. 0/1 means fixing to that value, * indicates optimization during training. Statistically significant improvement over 1-best/R+1 is in bold.

Test-time inputs	Trained on R	R+1	R+L	R+L+S
Reference	24.7	24.3	24.8	24.4
Oracle	15.8	16.8	16.3	15.9
1-best	11.8	13.3	12.4	12.0
Lattice	9.3	7.1	**13.7**	**14.1**

Table 5.6: Lattice-to-sequence experiments on Callhome. BLEU scores are on Callhome/Evltest using1 reference. All models are pretrained on Fisher/Train references (R), and potentially fine-tuned on Callhome/Train. The best result using 1-best or lattice inputs is in bold. Statistically significant improvement over 1-best/R+1 is in bold.

models on Fisher/Train, but then fine-tune them on the 9 times smaller Callhome/Train portion of the corpus. We fine-tune for 10 epochs, all other settings are as before. We use Callhome/Evltest for testing. Table 5.6 shows the results. The trends are consistent to Section 5.2.4.2: The proposed model (lattice/R+L+S) outperforms the 1-best baseline (1-best/R+1) by 0.8 BLEU points, which in turn beats the pretrained system (1-best/R) by 1.5 BLEU points. Including the lattice scores is clearly beneficial, although lattices without scores also improve over 1-best inputs in this experiment.

5.2.4.5 Impact of Lattice Quality

Next, we analyze the impact of using lattices and lattice scores as the ASR WER changes. For this purpose, we concatenate all development and test data sets from Table 3.5 and divide the result into bins according to the 1-best WER. We sample 1000 sentences from each bin, and compare BLEU scores between several models.

The results are shown in Figure 5.8. For very good WERs, lattices do not improve over 1-best inputs, which is unsurprising. In all other cases, lattices are helpful. Lattice scores are most beneficial for moderate WERs, and not beneficial for very high WERs. We speculate that for high WERs, the lattice scores tend to be less reliable than for lower WERs.

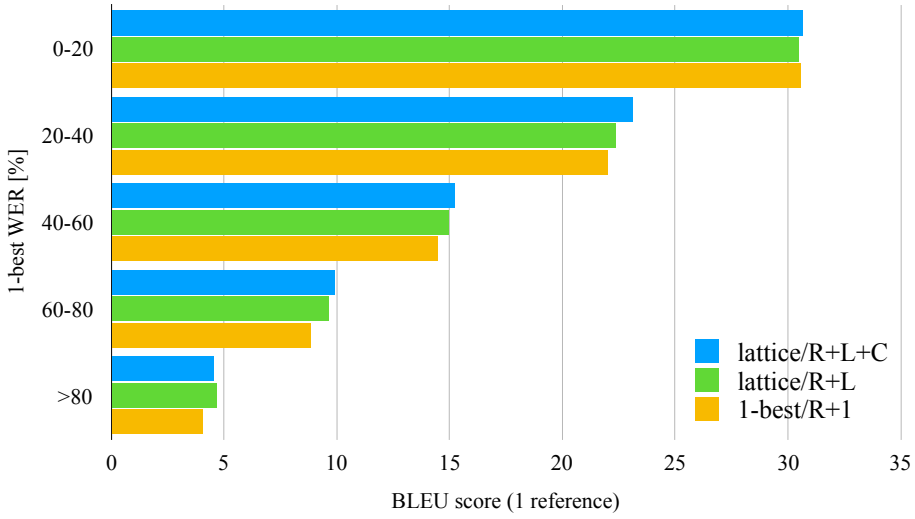


Figure 5.8: Lattice-to-sequence results over varying 1-best WER.

5.2.4.6 Example Translations

We list several cherry-picked examples for cases where using lattices improved the translation, as well as where lattices hurt accuracy.

Example 1: Erroneous 1-best

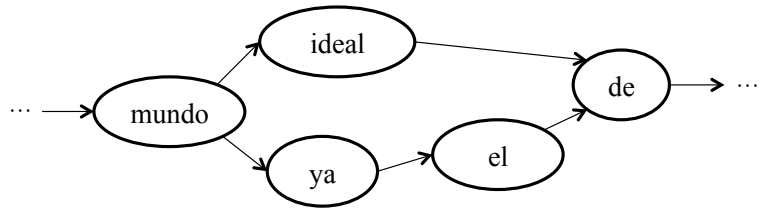
In this example, *ideal* is contained in lattice but not 1-best transcript, and consequently correctly translated only in the lattice/R+L+S setting.

1-best/R+1 input: *y y eso es algo que a mi me parece contraproducente verdad porque uno piensa y cuando ya a todos uno quisiera tal vez* ***un mundo*** *ya el*

de que una vez que cadena cuerpos trabajarán por el bienestar de de todos

1-best/R+1 output: *and , and that 's something that seems to me , right ? because one thinks , and when you think , and when everyone would like perhaps* ***a world*** *already , the one time that the chain changes for the*

lattice/R+L+S partial input:



lattice/R+L+S output: *and , and that 's something that seems to me , right ? because one thinks , and when you see , when you go to* ***a ideal world*** *, you see that they are illegals for the , well , they are all foreigners*

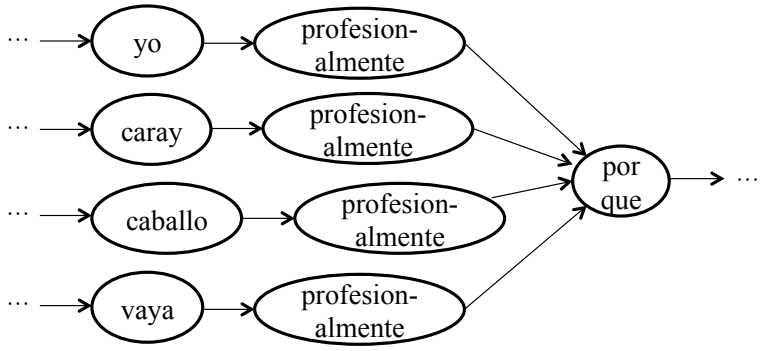
Example 2: Redundant Lattice Content

Another frequent pattern was a word appearing (once) in the 1-best transcript, but multiple times in the lattice, and only the lattice/R+L+S model translating this word.

1-best/R+1 input: *los que van porque que es un día los que van porque no tiene alicia derrita jugar y los que sí caray* ***profesionalmente*** *porque hay ciertos counselor bueno creo que soy josé playa que*

1-best/R+1 output: *the ones that go , because it 's a day that they go , because they don 't have alicia , play and the ones that are italian , because there are some* ***<unk>*** *, well , i think i 'm jose*

lattice/R+L+S partial input:



lattice/R+L+S output: *the ones that go , because it 's a day that they go because you don 't want to play and play , and the ones that influenced* ***professionally*** *, because there are certain things , well , i think that i 'm jose*

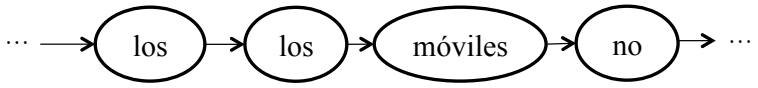
Example 3: Lattice Context

In this example, the lattice/R+L+S system correctly produced `<unk>` as translation to the unknown input word *móviles*, while the 1-best/R+1 system produced a seemingly random word instead. A possible explanation is the added context helping the `lat2seq` system to know when to be unsure.

1-best/R+1 input: *sí sí bueno contar otro que usaban los teléfonos satélite los los* ***móviles*** *no funcionaban bien porque pero a veces si funcionaban sí*

1-best/R+1 output: *yes , well , tell me that i used to use satellite phones , the* ***kids*** *didn 't work well , because sometimes it worked , yes*

lattice/R+L+S partial input:



lattice/R+L+S output: *yes , yes , well , with the other that i used to use the satellite phones , the , the* ⟨***unk***⟩ *didn 't work well because , but sometimes it worked , yes*

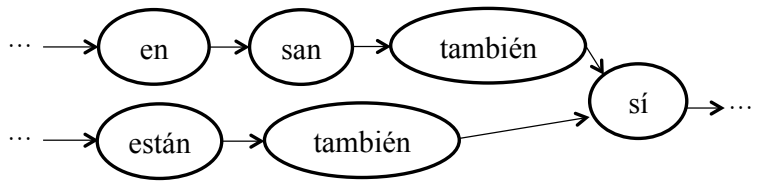
Counter Example

In this example, the lattice (but not the 1-best transcript) contained then word *san*, which tricked the `lat2seq` decoder to produce *san francisco*.

1-best/R+1 input: *pero los demás aquí están también sí está bien está tranquilo para acá*

1-best/R+1 output: *but the rest here are also , yes , it 's ok , it 's quiet for here*

lattice/R+L+S partial input:



lattice/R+L+S output: *but the rest here in* ***san francisco*** *is very quiet here*

5.2.5 Related Work

Ney [Ney99] and Casacuberta et al. [CNO+04] showed that tight integration of ASR and MT components are beneficial for the speech translation task. However, the employed FST formalism proved somewhat limiting, and follow-up work proposed improvements upon this in the context of traditional statistical MT models [SJVS04, ZKYL05, MHN08, BF05].

Lattices have also been found useful for other tasks, including translation of multiple word segmentations, and morphological alternatives [DMR08] and speech disfluency removal [CNW14].

Prior to our work, two papers have used lattices in the context of neural models. Ladhak et al. [LGD+16] proposed a lattice classification model by extending vanilla RNNs accordingly, while Su et a. [STX+17] have extended gated recurrent units (GRUs) in order to obtain a lattice-to-sequence model for the purpose of passing Chinese word segmentation lattices to a translation model. Both of these fail to incorporate lattice scores in an appropriate way into the model, which we have shown to be critical for exploiting lattices for speech translation purposes. Our work is the first to extend LSTMs, the most widespread RNN variant, to support lattice inputs. It is also the first work to integrate lattices into the training phase of a speech translation system.

5.3 Chapter Summary

In this chapter, we explored effective ways of coupling a neural machine translation model tightly to a preceding speech recognition stage in a cascaded speech translation scenario.

We first identified robustness to noisy inputs as a challenge for neural sequence-to-sequence models, and proposed to introduce randomized noise into the training by using a simple generative noise model. We found that this improves robustness when properly calibrating type and amount of noise, and that type and amount of noise at training and test time affect the length of the outputs. We highlighted the trade-off between trainability and distributional data similarity, and found that the amount of induced noise must be much smaller than the expected noise at test time for good results. Future work may investigate appropriate trade-offs between precision and recall when translating noisy inputs from a usability perspective, use our method for different tasks such as translating user-generated content, and experiment with more refined types of noise or other ways of modeling acoustic similarity in the

context of neural machine translation of ASR outputs.

Next, we investigated translating uncertain inputs from an error-prone up-stream component using a neural lattice-to-sequence model. Our proposed model takes word lattices as input and is able to take advantage of lattice scores. It is the first time that lattices have been incorporated already at the training stage of a speech translation system. In our experiments, we have found consistent improvements over translating 1-best transcriptions and that consideration of the lattice scores, especially in the attention mechanism, is crucial for obtaining these improvements.

Chapter 6

End-to-End Models

In the previous chapters, we introduced several ways to make use of the flexibility of encoder-decoder models to tighten the coupling between ASR and MT models and consequently to improve speech translation. We will now turn to the one-model approach in which we tackle speech translation by training a single model in an end-to-end fashion, i.e. by optimizing all parameters jointly. Mathematically, rather than decomposing as before, we model $P(T \mid X)$ directly.

Why might this be a good idea? There are a number of reasons:

- The joint optimization of parameter has generally been observed to be helpful in machine learning.
- Direct modeling allows exploiting end-to-end training data, which may be available in some situations but cannot be used for cascaded model training.
- This approach offers a way to circumvent error propagation. Error propagation can lead to compounding follow-up errors caused by a lack of robustness in the translation component or, relatedly, by the resulting mismatch between training and testing conditions.
- The assumption that $P(T \mid X, S) = P(T \mid S)$, i.e. the independence assumption between the translation and the audio signal given the source-language transcript, is not always correct. For example, some languages do not require a question mark when writing but do indicate interrogative statements through prosody, in which access to prosodic information can contain important information for the correct translation of a sentence. The one-model approach ensures that such information is accessible by the translation model.

- From a psychological point of view, it can be argued that the one-model approach is closer to how the human brain would perform a similar task. Inspiration from human cognitive processing can be helpful, given how effective humans are at handling spoken language. For instance, it has been noted that for many spoken texts, there is no such thing as the one correct interpretation, only a number of different reasonable interpretations [Buc01]. Lexical ambiguity is very common [RDJ05] and humans are able to maintain multiple interpretations and to revise earlier decisions while listening. This is also described as top-down or interactive processing [Buc01]. Humans are able to hold multiple syntactic interpretations valid until they can be resolved. This is, however, a highly demanding cognitive action [JC92]. The human ability to hold multiple interpretations implies that a hidden representation in continuous space may be of advantage for a computational model to represent a space of alternative possible interpretations. Also note that the high cognitive demands for humans implies a reasonable chance for computational models to outperform humans. This is especially true in highly demanding situations such as simultaneous interpretation. The above observations are underpinned by Seleskovitch, a professional interpreter and instructor for interpreters, who advocates retaining the *meaning* of the source language utterance, rather than the lexical items. Seleskovitch argues that concepts (semantic storage) are far easier to remember than words (lexical storage). Semantic storage also allows the interpreter to access concepts already stored in the brain, which allows the interpreter to naturally employ the brain's natural language-generation ability, by which humans convert concepts to words [Sel78].

6.1 Chapter Overview

Recent work showed that it is feasible to employ a single sequence-to-sequence model for speech translation [DAC+16, WCJ+17, BBKP18, AC18], offering appealing properties such as prevention of error propagation and joint model training. Despite these promises, two problems persist: First, reports on whether direct models (Fig. 6.1) outperform cascaded models are inconclusive, with some work in favor of direct models [WCJ+17], some in favor of cascaded models [KSN17, BBKP18], and one work in favor of direct models for two out of three tested language pairs [AC18]. Second, cascaded and direct models have been compared under identical data situations, but this is an unrealistic assumption: In practice, cascaded models can be trained on

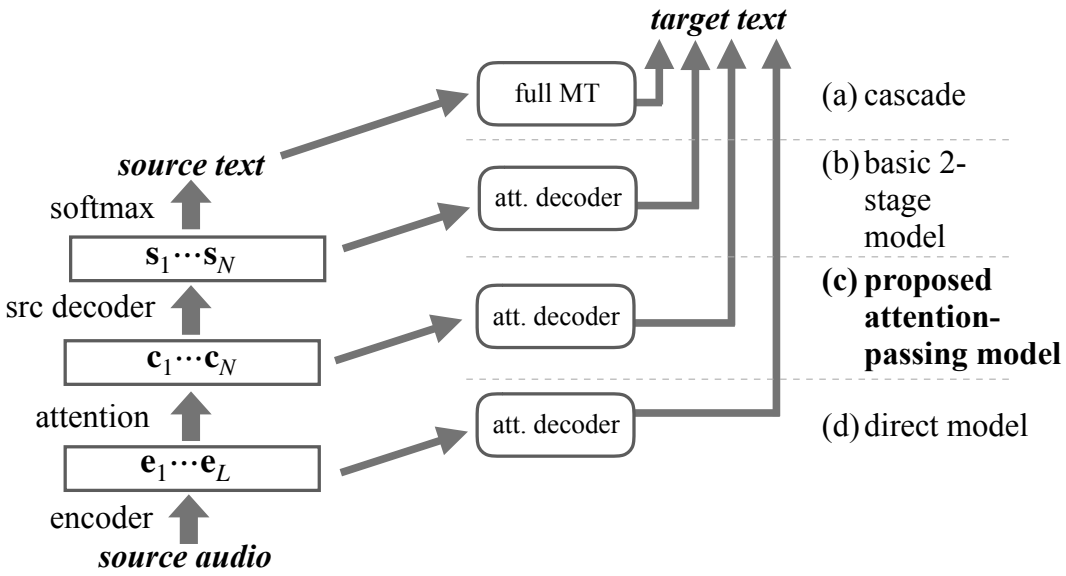


Figure 6.1: Conceptual diagrams for various speech translation approaches. *Cascade* is a traditional approach that applies machine translation to the output of a separately trained speech recognizer. All other models are end-to-end trainable: The *direct* model is a standard attentional encoder-decoder model. The basic 2-stage model uses two attention stages and passes the decoder states of a source-text decoder to a translation component. Our proposed *attention-passing* model operates in between, applying two attention stages but passing the more robust context vectors to the translation component, instead of passing the decoder states.

much more abundant independent ASR and MT corpora, while end-to-end models require hard-to-acquire end-to-end corpora of speech utterances paired with textual translations.

Our first contribution is a closer investigation of these two issues. Regarding the question of whether direct models or cascaded models are generally stronger, we hypothesize that direct models require more data to work well, due to the more complex mapping between inputs (source speech) and outputs (target text). This would imply that direct models outperform cascades when enough data is available, but under-perform in low-data scenarios. We conduct experiments and present empirical evidence in favor of this hypothesis. Next, for a more realistic comparison with regards to data conditions, we train a direct speech translation model using more auxiliary ASR and MT training data than end-to-end data. This can be implemented through multi-task training [WCJ+17, BBKP18]. Our results indicate that the auxiliary data is beneficial only to a limited extent, and that direct multi-task models are still heavily dependent on the end-to-end data.

As our second contribution, we apply a *two-stage* model [TLS+17, KSN17] as an alternative model to our problem, hoping that such models may overcome the data-efficiency shortcoming of the direct model. Basic two-stage models (Fig. 6.1) consist of a first-stage attentional sequence-to-sequence model that performs speech recognition and then passes the decoder states as input to a second attentional model that performs translation. This architecture is closer to cascaded translation while maintaining end-to-end trainability. Introducing supervision from the source-side transcripts midway through the model creates inductive bias that guides the complex transformation between source speech and target text through a reasonable intermediate representation closely tied to the source text. The architecture has been proposed by Tu et al. [TLS+17] to realize a reconstruction objective, and a similar model was also applied to speech translation [KSN17] to ease trainability, although no experiments under varying data conditions have been conducted. We hypothesize that such a model may help to address the identified data-efficiency issue: Unlike multi-task training for the direct model that trains auxiliary models on additional data but then discards many of the additionally learned parameters, the two-stage model uses all parameters of sub-models in the final end-to-end model. Empirical results confirm that the two-stage model is indeed successful at improving data-efficiency, but suffers from some degradation in translation accuracy under high data conditions compared to the direct model. One reason for this degradation is that this model re-introduces the problem of error-propagation, because the second stage of the model depends on the decoder states of the first model stage which often contain errors.

Our third contribution, therefore, is an *attention-passing* extension of the two-stage model (Fig. 6.1) that, instead of passing on possibly erroneous decoder states from the first to the second stage, passes on only the computed attention context vectors. We can view this approach as replacing the early decision on a source-side transcript by an early decision only on the *attention scores* needed to compute the same transcript, where the attention scores are expectedly more robust to errors in transcript decoding. We explore several variants of this model and show that it outperforms both the direct model and the vanilla two-stage model, while maintaining the improved data-efficiency of the latter. Through an analysis, we further observe that there is a trade-off between sensitivity to error propagation and data-efficiency.

6.2 Baseline Models

This section introduces a direct and a two-stage model for performing end-to-end speech translation. Both are based on the attentional encoder-decoder architecture [BCB15] with character-level outputs, and use the encoder architecture described in Section 4.1. Note that we do not use the self-attentional acoustic model of Section 4.2 here because some of our more elaborate models do not fit in GPU memory with these. These models are limited[1] by the fact that they can *only* be trained on end-to-end data which is much harder to obtain than ASR or MT data used to train traditional cascades.[2] Section 6.3 will introduce multi-task training as a way to overcome this limitation.

6.2.1 Direct Model

The sequence-to-sequence model with audio inputs described in Section 4.1 can be trained as a direct speech translation model by using speech data as input and the corresponding translations as outputs. Such a model does not rely on intermediate ASR output and is therefore not subject to error propagation. However, the transformation from source speech inputs to target text outputs is much more complex than that of an ASR or MT system taken individually, which may cause the model to require more data to perform well.

To make matters precise, given L audio encoder states $\mathbf{e}_{1:L}$ using the audio encoder as described previously, the direct model is computed as

$$\mathbf{s}_i = \mathrm{LSTM}\left(\left[W_{\mathrm{e}} y_{i-1}; \mathbf{c}_{i-1}\right], \mathbf{s}_{i-1}; \theta_{\mathrm{lstm}}\right) \tag{6.1}$$

$$\mathbf{c}_i = \mathrm{Attention}\left(\mathbf{s}_i, \mathbf{e}_{1:L}; \theta_{\mathrm{att}}\right) \tag{6.2}$$

$$\tilde{\mathbf{s}}_i = \tanh\left(W_{\mathrm{s}}\left[\mathbf{s}_i; \mathbf{c}_i\right] + \mathbf{b}_{\mathrm{s}}\right) \tag{6.3}$$

$$p\left(y_i \mid y_{<i}, \mathbf{e}_{1:L}\right) = \mathrm{SoftmaxOut}\left(\tilde{\mathbf{s}}_i; \theta_{\mathrm{out}}\right). \tag{6.4}$$

Here, W_*, $\mathbf{b}_*$, and θ_* are the trainable parameters, y_i are output characters, and SoftmaxOut denotes an affine projection followed by a softmax operation. $\mathbf{s}_i$ are decoder states with $\mathbf{s}_0$ initialized to the last encoder state, and $\mathbf{c}_i$ are attentional context vectors

[1] Prior work noted that in severe low-resource situations it may actually be easier to collect speech paired with translations than transcriptions [DAC+16]. However, we focus on well-resourced languages for which ASR and MT corpora exist and for which it is more realistic to obtain good speech translation accuracy.

[2] As a case in point, the largest available speech translation corpora [PKL+13, KBK18] are an order of magnitude smaller than the largest speech recognition corpora [CMW04, PCPK15] ($\sim$ 200 hours vs 2000 hours) and several orders of magnitude smaller than the largest machine translation corpora, e.g. those provided by the Conference on Machine Translation (WMT).

with $\mathbf{c}_0$=$\mathbf{0}$. In equation 6.2, we compute Attention $(\cdot) = \sum_{j=1}^{L} \alpha_{ij}\mathbf{e}_j$ with weights α_{ij} conditioned on $\mathbf{e}_j$ and $\mathbf{s}_i$, parametrized by θ_{att}, and normalized via a softmax operation.

6.2.2 Basic Two-Stage Model

As an alternative to the direct model, the two-stage model uses a cascaded information flow while maintaining end-to-end trainability. Our main motivation for using this model is the potentially improved data-efficiency when adding auxiliary ASR and MT training data (Section 6.3). This model is similar to the architecture first described by Tu et al. [TLS+17]. It combines two encoder-decoder models in a cascade-like fashion, with the decoder of the first stage and the encoder of the second stage being shared (Fig. 6.2). In other words, while a cascade would use the source-text outputs of the first stage as inputs into the second stage, in this model the second stage directly computes attentional context vectors over the decoder states of the first stage. The inputs of the two-stage model are speech frames, the outputs of the first stage are transcribed characters in the source language, and the outputs of the second stage are translated characters in the target language.

Again assuming L audio encoder states $\mathbf{e}_{1:L}$, the first stage outputs of length N are computed identically to equations 6.1–6.4, except that input feeding (conditioning the decoding step on the previous context vector) is not used in the first stage decoder to keep components compatible for multi-task training (Section 6.3.2).:

$$\mathbf{s}_i^{\text{src}} = \text{LSTM}\left(W_{\text{e}}^{\text{src}} y_{i-1}^{\text{src}}, \mathbf{s}_{i-1}^{\text{src}}; \theta_{\text{lstm}}^{\text{src}}\right) \tag{6.5}$$

$$\mathbf{c}_i^{\text{src}} = \text{Attention}\left(\mathbf{s}_i^{\text{src}}, \mathbf{e}_{1:L}; \theta_{\text{att}}^{\text{src}}\right) \tag{6.6}$$

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}}\left[\mathbf{s}_i^{\text{src}}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right) \tag{6.7}$$

$$p\left(y_i^{\text{src}} \mid y_{<i}, \mathbf{e}_{1:L}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_i^{\text{src}}; \theta_{\text{out}}^{\text{src}}\right) \tag{6.8}$$

Next, the second stage proceeds similarly but using the stage 1 decoder states as input:

$$\mathbf{s}_j^{\text{trg}} = \text{LSTM}\left(\left[W_{\text{e}}^{\text{trg}} y_{i-1}^{\text{trg}}; \mathbf{c}_{j-1}^{\text{trg}}\right], \mathbf{s}_{j-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{trg}}\right) \tag{6.9}$$

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{s}_{1:N}^{\text{src}}; \theta_{\text{att}}^{\text{trg}}\right) \tag{6.10}$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s}}^{\text{trg}}\left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{trg}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \tag{6.11}$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \tag{6.12}$$

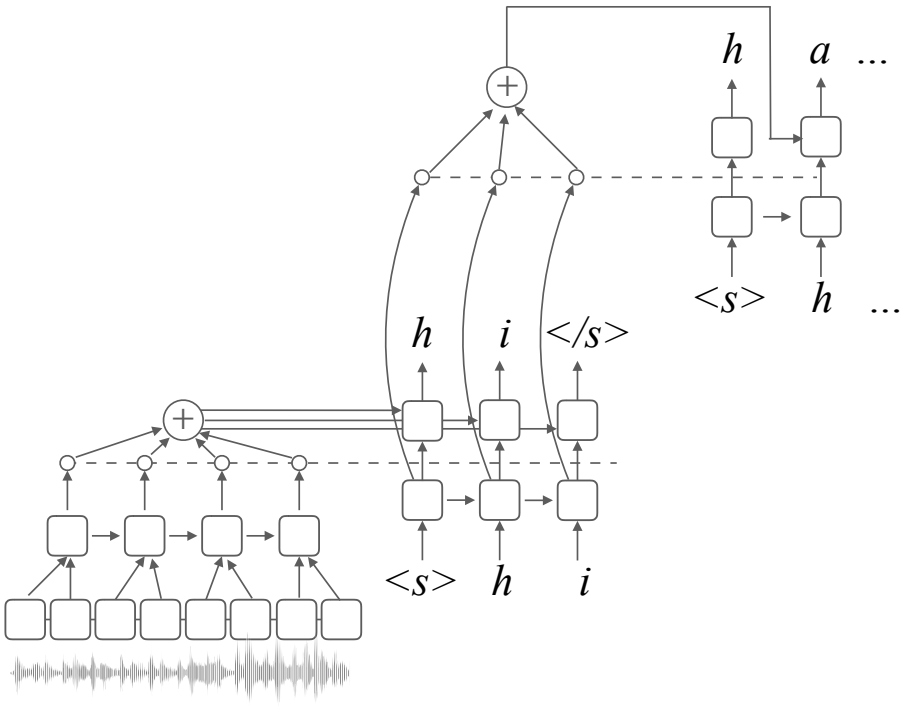


Figure 6.2: Basic two-stage model. The decoder states of the first stage double as encoder states for the second stage.

6.2.3 Cascaded Model

We finally employ a traditional cascaded model as a baseline, whose architecture is kept as similar to the above models as possible in order to facilitate meaningful comparisons. The cascade consist of an ASR component and an MT component, which are both attentional sequence-to-sequence models according to equations 6.1–6.4, trained on the appropriate data. The ASR component uses the acoustic encoder of Section 4.1, while the MT model uses a bidirectional LSTM with 2 layers as encoder.

6.3 Incorporating Auxiliary Data

The models described in Section 6.2 are trained only on speech-to-translation pairs (and transcripts in the case of Section 6.2.2), which is a severe limitation. To incorporate auxiliary ASR and MT data into the training we make use of a multi-task training strategy. Such a strategy trains auxiliary ASR and MT models that share certain parameters with the main speech translation model. We implement multi-task training by drawing several minibatches, one minibatch for each task, and performing an update

based on the accumulated gradients across tasks. Note that this results in a balanced contribution of each task.[1]

6.3.1 Multi-Task Training for the Direct Model

Multi-task training for direct speech translation models has previously been used [WCJ+17, BBKP18], although not for the purpose of adding additional training utterances that are not shown to the main speech translation task.[2] We distinguish five model components: a source speech encoder, a source text encoder (a two-layer bidirectional LSTM working on character level), a source text decoder, a target text decoder, and a general-purpose attention mechanism. The latter is a standard MLP-based attention mechanism whose trainable parameters we opt to share across all tasks, i.e. the attention mechanism learns to relate both source-text and target-text decoder states to both source-text and source-speech encoder states. There are four ways in which these components can be combined into a complete sequence-to-sequence model (see Figure 6.3), corresponding to the following four tasks:

ASR: Combines source speech encoder, general-purpose attention, source text decoder. This is similar to the auxiliary ASR task used by Weiss et al. [WCJ+17] and can be trained on common ASR data.

MT: Combines source text encoder, general-purpose-attention, target text decoder. The addition of an MT task has been mentioned by Bérard et al. [BBKP18] and allows training on common MT data.

ST: Combines source speech encoder, general-purpose-attention, target text decoder. This is our main task and requires end-to-end data for training.

Auto-encoder (AE): Combines source text encoder, general-purpose attention, source text decoder. The AE task can be trained on monolingual corpora in the source language and may serve to tighten the coupling between components and potentially improves the parameters of the general-purpose attention model. We have observed slight improvements by adding the AE task in preliminary experiments and will therefore use it throughout this chapter.

[1] We also experimented with a final fine-tuning phase on only the main task [NC17], but discarded this strategy for lack of consistent gains.

[2] Note that [BKL+19] do experiment with additional speech recognition data, although, differently from our work, for purposes of cross-lingual transfer learning.

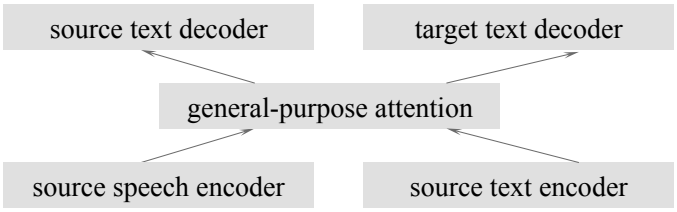


Figure 6.3: Direct multi-task model with shared model components.

6.3.2 Multi-Task Training for the Two-Stage Model

Including an auxiliary ASR task is straight-forward with the two-stage model by simply computing the cross-entropy loss with respect to the softmax output of the first stage, and dropping the second stage.

The auxiliary MT task computes only the second stage, replacing the inputs $\mathbf{s}_{1:N}^{\text{src}}$ by states $\mathbf{e}_{1:N}^{\text{asr}}$ computed as:

$$\mathbf{e}_i^{\text{asr}} = \text{LSTM}\left(W_{\text{e}}^{\text{src}} y_i^{\text{transcr}}, \mathbf{e}_{i-1}^{\text{src}}; \theta_{\text{lstm}}^{\text{src}}\right). \tag{6.13}$$

That is, instead of computing the second-stage inputs using the first stage, we compute these inputs through a conventional encoder that encodes the reference transcript $y_{1:N}^{\text{transcr}}$ and uses the same embeddings matrix and unidirectional LSTM as the first stage decoder.

Note that there is no equivalent to the auto-encoder task of the direct multi-task model here.

Why might this architecture help to make better use of auxiliary ASR and MT data? Note that in the direct model only roughly half of the model parameters are shared between the main task and the ASR task, and likewise for main and MT tasks (Section 6.3.1). Additional data would therefore only have a rather indirect impact on the main task. In contrast, in the two-stage model all parameters of the auxiliary tasks are shared with the main task and therefore have a more direct impact, potentially leading to better data efficiency.

Note that relatedly to our multi-task strategy, Kano et al. [KSN17] have decomposed their two-stage model in a similar way to perform pretraining for the individual stages. This model can thus incorporate additional auxiliary data in a similar fashion, although no experiments in this direction are presented in the paper.

6.4 Attention-Passing Model

We have so far described a direct model that has the appealing property of avoiding error propagation in a principled way but may not be particularly data efficient, and have described a two-stage model that addresses the latter disadvantage. Unfortunately, the two-stage model re-introduces the error propagation problem back into end-to-end modeling, because the second stage heavily depends on the potentially erroneous decoder states of the first stage. We therefore propose an improved *attention-passing* model in this section that is less impacted by error propagation issues.

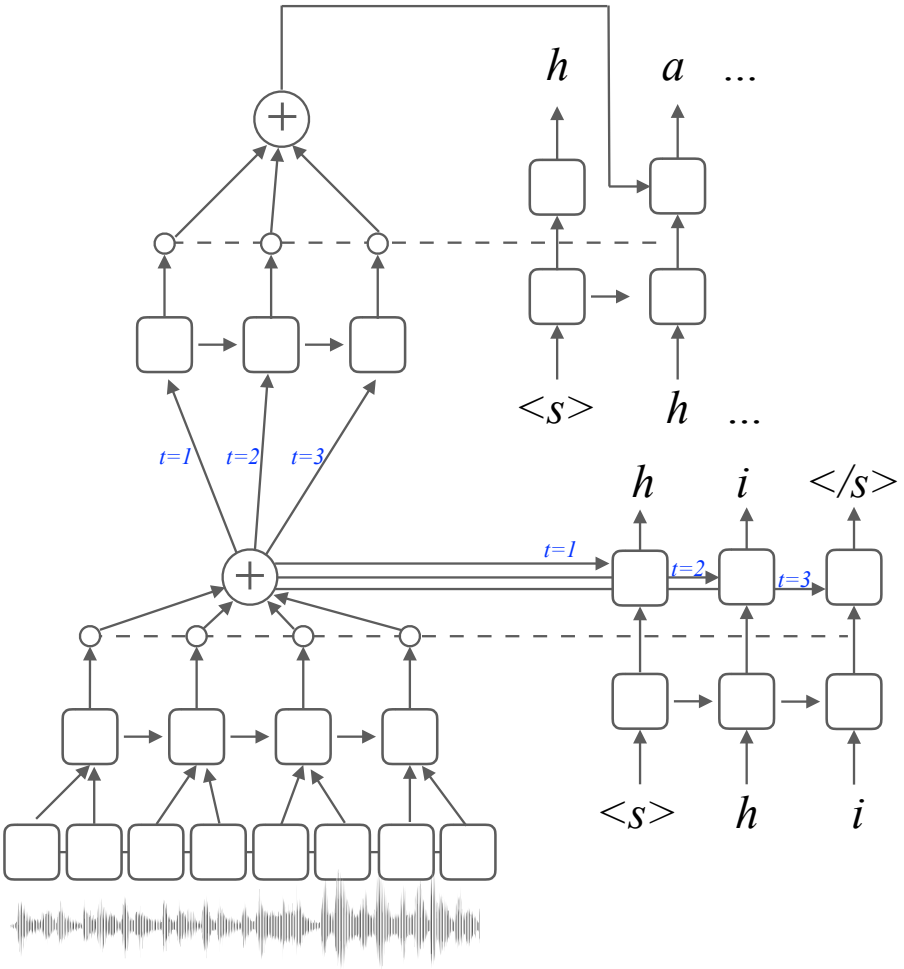


Figure 6.4: Attention-passing model architecture.

6.4.1 Preventing Error Propagation

The main idea behind the attention-passing model is to not feed the erroneous first-stage decoder states to the second stage, but instead pass on only the *context vectors* that summarize the relevant encoded audio at each decoding step. The first stage decoder is unfolded as usual by employing discrete source-text representations, but the only information exposed to the translation stage are the per-timestep context vectors created as a by-product of the decoding. Figure 6.4 illustrates this idea. Intuitively, we expect this to help because we no longer make an early decision on the identity of the source-language text, but only on the corresponding attentions. This is motivated by our observation that recognition attentions are sufficiently robust against recognition errors (Section 6.5.7).

Formally, the first stage remains unchanged from equations 6.5–6.8. The context vectors $\mathbf{c}_i^{\text{src}}$ then form the input to the second stage:

$$\mathbf{x}_i^{\text{trg}} = \text{LSTM}\left(\mathbf{c}_i^{\text{src}}, \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right) \tag{6.14}$$

$$\mathbf{s}_j^{\text{trg}} = \text{LSTM}\left(\left[W_{\text{e}}^{\text{trg}} y_{i-1}^{\text{trg}}; \mathbf{c}_{j-1}^{\text{trg}}\right], \mathbf{s}_{j-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{trg}}\right) \tag{6.15}$$

$$\mathbf{c}_j^{\text{trg}} = \text{Attention}\left(\mathbf{s}_j^{\text{trg}}, \mathbf{x}_{1:N}^{\text{trg}}; \theta_{\text{att}}^{\text{trg}}\right) \tag{6.16}$$

$$\tilde{\mathbf{s}}_j^{\text{trg}} = \tanh\left(W_{\text{s,trg}}\left[\mathbf{s}_j^{\text{trg}}; \mathbf{c}_j^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{trg}}\right) \tag{6.17}$$

$$p\left(y_j^{\text{trg}} \mid y_{<j}, \mathbf{s}_{1:N}^{\text{src}}\right) = \text{SoftmaxOut}\left(\tilde{\mathbf{s}}_j^{\text{trg}}; \theta_{\text{out}}^{\text{trg}}\right) \tag{6.18}$$

6.4.2 Decoder State Drop-Out

In addition to the modifications described in Section 6.4.1, we introduce a block drop-out operation [AMB+16] on the decoder states (see Figure 6.5), replacing equation 6.7 by

$$\tilde{\mathbf{s}}_i^{\text{src}} = \tanh\left(W_{\text{s}}^{\text{src}}\left[\text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}; \mathbf{c}_i^{\text{src}}\right] + \mathbf{b}_{\text{s}}^{\text{src}}\right).$$

The block drop-out operation, denoted as BDrop, replaces the whole vector by zero with a certain probability (here: 0.5). This results in the context vectors $\mathbf{c}_i^{\text{src}}$ becoming the only information available to the output layer whenever the decoder states are dropped out. The motivation for this is to force the model to maximize the informativeness of the context vectors, which are later relied upon as sole inputs to the second stage. Our experiments show that this strategy improves the model accuracy (see Table 6.2).

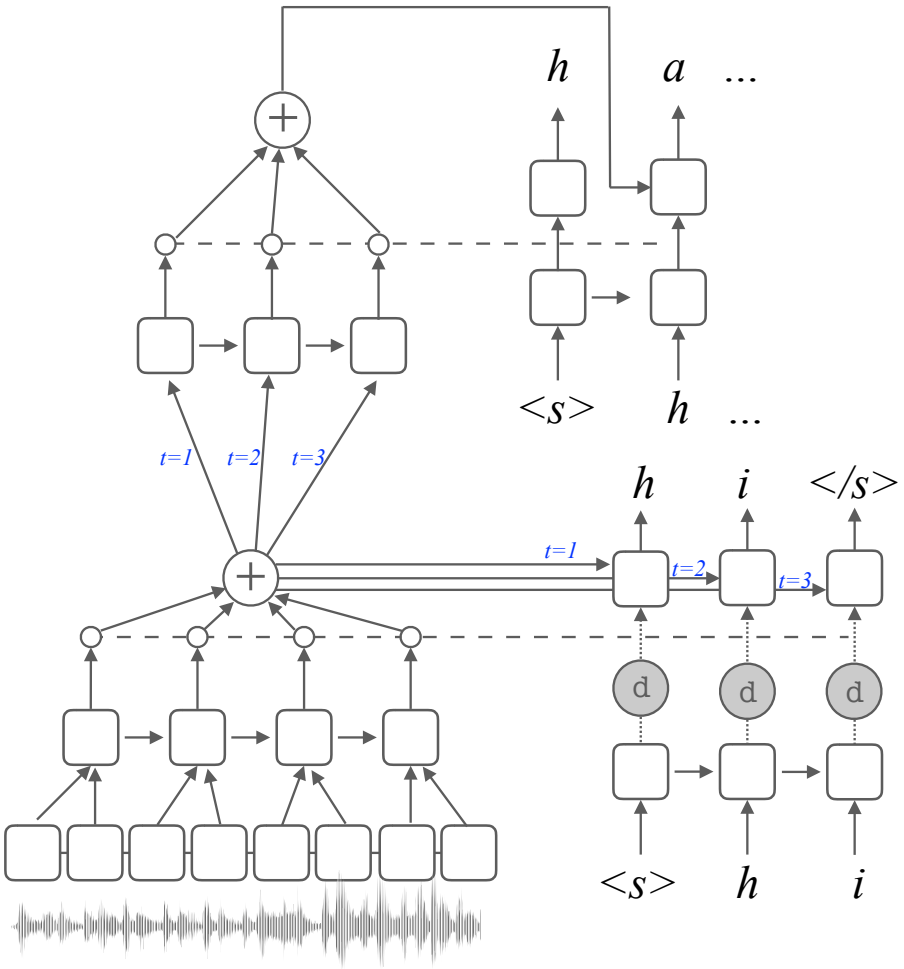


Figure 6.5: Attention-passing model with block drop-out.

6.4.3 Multi-Task Training

Similar to the basic two-stage model, the attention-passing model as a whole is trained on speech-transcript-translation triplets, but can be decomposed into two sub-models that correspond to ASR and MT tasks. In fact, the ASR task is unchanged with the exception of the new block dropout operation. The MT task is obtained by replacing equation 6.14 by

$$\mathbf{x}_i^{\text{trg}} = \text{LSTM}\left(W_e \mathrm{y}_i^{\text{src}}, \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right),$$

i.e. by using the transcript character embeddings as inputs instead of the context vectors used when training the main task. Note that the LSTMs in equations 6.5 and 6.14 are shared in order to have a match between stage 1 decoder and stage 2 encoder as with

the basic model.

6.4.4 Cross Connections

As a further extension to the attention-passing model of Section 6.4.1, we can introduce cross connections that concatenate the dropped-out first stage hidden decoder states to the second-stage encoder inputs (see Figure 6.6). This causes equation 6.14 to be replaced by

$$\mathbf{x}_i^{\text{trg}} = \text{LSTM}\left(\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right], \mathbf{x}_{i-1}^{\text{trg}}; \theta_{\text{lstm}}^{\text{src}}\right) \tag{6.19}$$

This extension moves the model closer to the basic two-stage model, while the inclusion of the context vectors and the block drop-out operation on the hidden decoder states ensure that the second stage decoder does not rely too strongly on the first stage outputs.

6.4.5 Additional Loss

We further experiment with introducing an additional loss aimed at making the LSTM inputs between first stage decoder and second stage encoder RNN more similar. Recall that in our attention-passing model, both RNNs share parameters (equations 6.5 and 6.14), so that similar inputs at both times are desirable. The loss is defined as follows:

$$\mathcal{L}_{\text{add}} = ||\mathbf{c}_i^{\text{src}} - W_e \text{y}_i^{\text{src}}||_2.$$

If combined with the cross connections (Section 6.4.4), the formula is adjusted to $\mathcal{L}_{\text{add}} = ||\,\text{Affine}\left[\mathbf{c}_i^{\text{src}}; \text{BDrop}\left\{\mathbf{s}_i^{\text{src}}\right\}\right] - W_e \text{y}_i^{\text{src}}||_2$. We did not find it beneficial to apply a scaling factor when adding this loss to the main cross-entropy loss in our experiments.

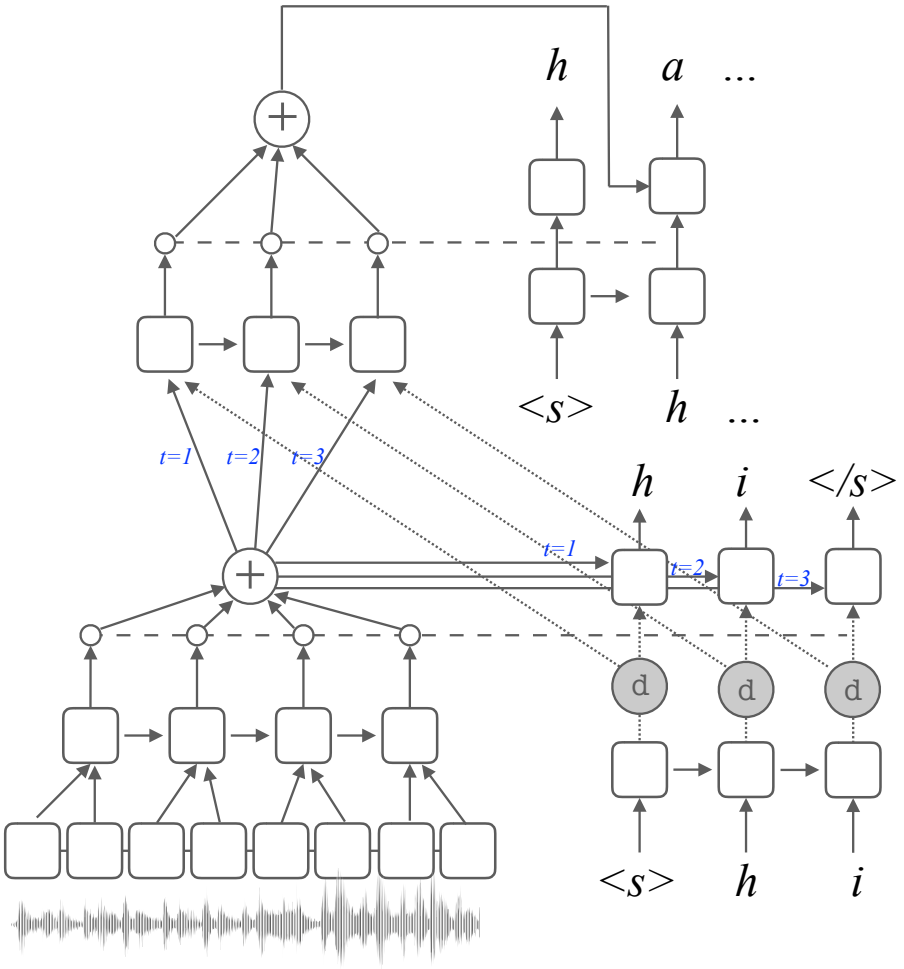


Figure 6.6: Attention-passing model with cross connections.

6.5 Experiments

We conduct experiments on the Fisher and Callhome Spanish–English Speech Translation Corpus [PKL+13], a corpus of Spanish telephone conversations that includes audio, transcriptions, and translations into English. We use the Fisher portion that consists of telephone conversations between strangers. The training data size is 138,819 sentences, corresponding to 162 hours of speech. ASR word error rates (WER) on this dataset are usually relatively high due to the spontaneous speaking style and challenging acoustics. From a translation viewpoint, the data can be considered as relatively easy with regards to both the topical domain and particular language pair.

The vocabulary consists of the common characters appearing in English and

Spanish, apostrophe, whitespace, and special start-of-sequence and unknown-character tokens. The same vocabulary is used on both encoder (for the MT auxiliary task) and decoder sides. We set the batch size dynamically depending on the input sequence size such that the average batch size is 24 sentences. We use Adam [KB14] with initial learning rate of 0.0005, decayed by 0.5 when the validation BLEU score did not improve over 10 check points initially and 5 check points after the first decay. We initialize attention-passing models using weights from a basic two-stage model trained on the same data.

Following Weiss et al. [WCJ+17], we lowercase texts and remove punctuation. As speech features, we use 40-dimensional Mel filter bank features with per-speaker mean and variance normalization. We exclude a small number of utterances longer than 1500 frames from the training data to avoid running out of memory. The encoder-decoder attention is MLP-based, and the decoder uses a single LSTM layer.[1] The number of hidden units is 128 for the encoder-decoder attention MLP, 64 for target character embeddings, 256 for the encoder LSTMs in each direction, and 512 elsewhere. The model uses variational recurrent dropout with probability 0.3 and target character dropout with probability 0.1 [GG16]. We apply label smoothing [SVI+16] and fix the target embedding norm to 1 [NC18]. For inference, we use beam search with beam size 15 and polynomial length normalization with exponent 1.5.[2]

All BLEU scores are computed on Fisher/Test and scored against 4 references.

6.5.1 Cascaded vs. Direct Models

We first wish to shed light on the question of whether cascaded or direct models can be expected to perform better. This question has been investigated previously [WCJ+17, KSN17, BBKP18, AC18], but with contradictory findings. We hypothesize that the increased complexity of the direct mapping from speech to translation increases the data requirements of such models. Table 6.1 compares the direct multi-task model (Section 6.3.1) against a cascaded model with identical architecture to the respective ASR- and MT sub-models of the multi-task model. The direct model is trained with multi-task training on the auxiliary ASR-, MT-, and AE tasks on the same data which outperformed single-task training considerably in preliminary experiments. As can be seen, the direct model outperforms the traditional cascaded set-up only when both

[1] Weiss et al. [WCJ+17] report improvements from deeper decoders, but we encountered stability issues and therefore restricted the decoder to a single layer.

[2] For two-stage models, we apply beam search only for the second stage decoder. We do not employ the two-phase beam search of Tu et al. [TLS+17] because of its prohibitive memory requirements.

are trained on the full data, but not when using only parts of the training data. This provides evidence in favor of our hypothesis and indicates that direct end-to-end models should be expected to perform strongly only in a case where enough training data is available.

Training sentences	Cascade	Direct model
139k	32.45	**35.30**
69k	**26.52**	24.68
35k	**16.84**	14.91
14k	**6.59**	6.08

Table 6.1: BLEU scores (4 references) on Fisher/Test for various amounts of training data. The direct model performs best in the full data condition, but the traditional cascaded model is best in all reduced data conditions.

6.5.2 Two-Stage Models

Next, we investigate performance of the two-stage models, for both the basic variant (Section 6.3.2) and our proposed attention-passing models (Section 6.4). Again, all models are trained in a multi-task fashion by including auxiliary ASR and MT tasks based on the same data. Table 6.2 shows the results. The basic two-stage model performs in between the direct and cascaded models from Section 6.5.1. The attention-passing model (Section 6.4.1), which is designed to circumvent the negative effects of error propagation, outperforms the basic variant and performs similarly to the direct model. The model extensions (Sections 6.4.4 and 6.4.5) further improved the results, with the best model outperforming the direct model by 1.40 BLEU points and the basic two-stage model by 2.34 BLEU points absolute. The last row in the table confirms that the block dropout operation contributed to the gains: removing it led to a drop by 0.66 BLEU points.

6.5.3 Data Efficiency: Direct Model

Having established results in favor of our proposed model on the full data, we now examine the data efficiency of the different models. Our experimental strategy is to compare model performance (1) when trained on the full data, (2) when trained on partial data, and (3) when trained on partial speech-to-translation data but full auxiliary (ASR+MT) data.[1]

[1] An alternative experimental strategy is to train on the full data and then add auxiliary data from other domains to the training. We pursue this strategy in Section 6.5.5 as a more realistic scenario,

Model	BLEU
Cascade	32.45
Direct	35.30
Basic two-stage	34.36
Attention-passing	35.31
+ cross connections	36.51
+ cross connections + additional loss	**36.70**
Best w/o block dropout	36.04

Table 6.2: Results for cascaded and multi-task models under full data conditions.

Figure 6.7 shows the results, comparing the cascaded model against the direct model trained under conditions (1), (2), and (3).[1] Unsurprisingly, the performance of the direct model trained on partial data declines sharply as the amount of data is reduced. Adding auxiliary data through multi-task training improves performance in all cases. For instance, in the case of 69k available speech-to-translation instances, adding the full auxiliary data helps to reach the accuracy of the cascaded model. However, note that this is already somewhat disappointing because the end-to-end data, which is not available to the cascaded model, no longer yields an advantage. Moreover, reducing the end-to-end data further reveals that multi-task training is not able to close the gap to the cascade. In the scenario with 35k end-to-end instances and full auxiliary data, the direct model underperforms the cascade by 9.14 BLEU points (32.50 vs. 23.36), despite being trained on *more* data. The unsatisfactory data efficiency in this controlled ablation study is also a strong indicator that the direct model can be expected to fall behind a cascade that is trained on large amounts of external data. This claim is verified in Section 6.5.5.

6.5.4 Data Efficiency: Two-Stage Models

We showed that the direct model is poor at integrating auxiliary data and heavily depends on sufficient amounts of end-to-end training data. How do two-stage models behave with regards to this data efficiency issue? Figure 6.8 shows that both the basic and best proposed two-stage models perform reasonably well even when having seen

but point out several problems that lead us to not use this as our main approach: Adding external auxiliary data (1) leads to side-effects due to domain mismatch and (2) severely limits the number of experiments that we can conduct due to the considerably increased training time.

[1] Note that the above hyper-parameters were selected for best full-data performance and are not re-tuned here.

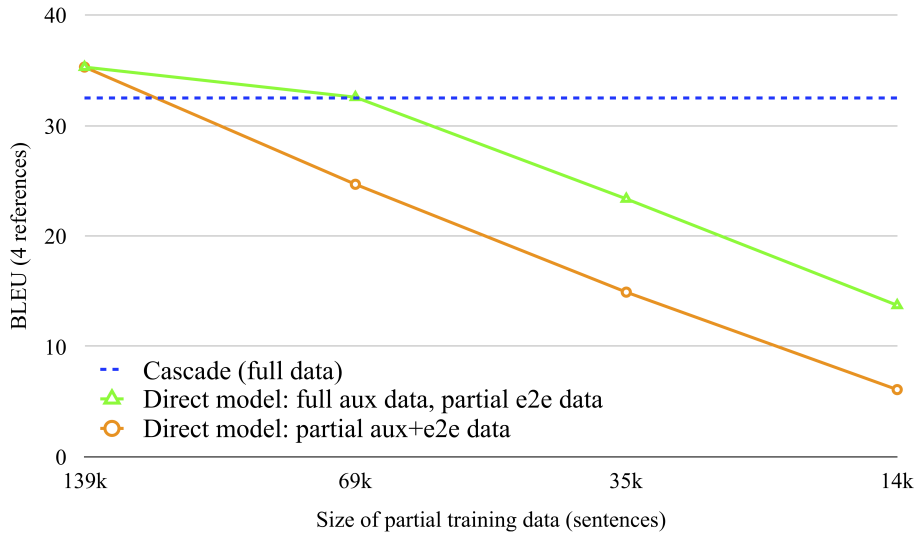


Figure 6.7: Data efficiency for direct (multi-task) model, compared against cascade on full auxiliary data.

much less end-to-end data. We can explain this by noticing that the two-stage model can be naturally decomposed into an ASR sub-model and an MT sub-model, while the direct model needs to add auxiliary sub-models to support multi-task training. Interestingly, the attention-passing model without cross-connections does better than the direct model with regards to data efficiency, but falls behind the basic and best proposed two-stage models. This indicates that access to ASR labels in some form contributes to favorable data-efficiency of speech translation models.

6.5.5 Adding External Data

Our approach for evaluating data efficiency so far has been to assume that end-to-end data is available for only a subset of the available auxiliary data. In practice, we can often train ASR and MT tasks on abundant external data. We therefore run experiments in which we use the full Fisher training data for all tasks as before, and add OpenSubtitle[1] data for the auxiliary MT task. We clean and normalize the Spanish-English OpenSubtitle 2018 data [Tie09] to be consistent with the employed Fisher training data by lowercasing and removing punctuation. We apply a basic length filter and obtain 61M sentences. During training, we include the same number of sentences from in-domain and out-of-domain MT tasks in each minibatch in order to prevent degradation due to domain mismatch. Our models converged before a full

[1] `http://www.opensubtitles.org/`

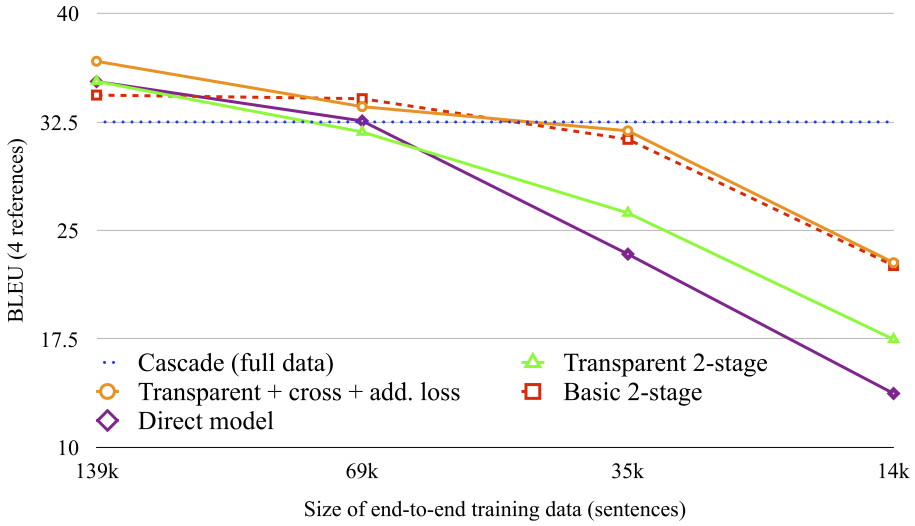


Figure 6.8: Data efficiency across model types. All models use full auxiliary training data through multi-task training.

pass over the OpenSubtitle data, but needed between two and three times more steps than the in-domain model to converge.

Table 6.3 shows that all models were able to benefit from the added data. However, when examining the relative gains we can see that both the cascaded model and the two-stage models benefitted about twice as much from the external data as the direct model. In fact the basic two-stage model now slightly surpasses the direct model, and our best proposed two-stage model is ahead of the basic two-stage model by almost the same absolute difference as before (2.36 BLEU points). These relative gains show that our findings from Sections 6.5.3 and 6.5.4, namely that two-stage models are much more efficient at exploiting auxiliary training data, generalizes to the setting in which large amounts of out-of-domain data are added to the MT task. Out-of-domain data is often much easier to obtain, and we can therefore conclude that the proposed two-stage approach are preferable in many practically relevant situations. Because these experiments are very expensive to conduct, we leave experiments with external ASR data for future work.

6.5.6 Error Propagation

To better understand the impact of error propagation, we analyze how improved or degraded ASR outputs impact the translation results. This experiment is applicable to the attention-passing model, the two-stage model, and the cascade, but not to the

Model	Fisher	Fisher+OpenSub	Relative Improvement
Cascade	32.45	34.58	+6.2%
Direct model	35.30	36.45	+3.2%
Basic two-stage	34.36	36.91	+6.9%
Best proposed	36.70	38.81	+5.4%

Table 6.3: BLEU scores when adding auxiliary OpenSubtitles MT training data. The two-stage models benefit much more strongly than the direct model, with our proposed model yielding the strongest overall results.

direct model which does not compute ASR outputs during inference. We analyze three different settings: using the standard decoded ASR labels, replacing these labels with the gold labels, or artificially degrading the decoded labels by randomly introducing 10% of substitution, insertion, and deletion noise (Section 5.1.1.1). Intuitively, models that suffer from error propagation issues are expected to rely most heavily on these intermediate labels and would therefore be most impacted by both degraded and improved labels.

Table 6.4 shows the results. Unsurprisingly, the cascade responds most strongly to improved or degraded noise, confirming that it is severely impacted by error propagation. Our attention-passing approach that does not directly expose the labels to the translation sub-model is much less impacted. However, the impact is still more significant than perhaps expected, indicating that improved attention models that are more robust to decoding errors [CBS$^+$15, TSN17] may serve to further improve our model in the future. Note that the attention-passing model benefits poorly from gold ASR labels, which is expected because gold labels only improve the ASR alignments and by extension the passed context vectors, but these are quite robust against decoding errors in the first place.

The basic two-stage model is impacted significantly but less strongly than the cascade, in line with our claim that such models are subject to error propagation despite being end-to-end trainable. Note that it falls behind the cascade for gold labels, despite both models being seemingly identical under this condition. This can be explained by the cascaded model's use of beam search and greater number of encoder layers.

Somewhat contrary to our expectations, the model with dropped out cross connections appears equally subject to error propagation despite the block dropout on these connections, displaying the same accuracy gains across the three different settings. This suggests future explorations may be able to determine model variants with an even better trade-off between overall accuracy, data-efficiency, and amount of degradation due to error propagation.

ASR labels	Gold	Decoded	Perturbed
Cascade	58.15 (+44%)	32.45	25.67 (-26%)
Basic two-stage	56.60 (+39%)	34.36	28.81 (-19%)
Attention-passing	40.70 (+13%)	35.31	31.96 (-10%)
+ cross connections	58.29 (+37%)	36.70	30.48 (-20%)

Table 6.4: Robustness against error propagation. The table shows the effect of altering the ASR labels for different models as a measure for robustness. Percentages are relative to the results for unaltered (decoded) ASR labels.

6.5.7 Robustness of ASR Attentions

Our attention-passing model was motivated by the assumption that attention scores are relatively robust against recognition errors. We perform a qualitative analysis to validate this assumption. Figure 6.9 shows the first-stage attention matrix when force-decoding the reference transcript, while Figure 6.10 shows the same for regular decoding, which for this utterance produced significant errors. Despite the errors, we can see that the attention matrices are very similar. We manually inspected the first 100 test attention matrices and confirm that this behavior occurs very consistently. Further quantitative evidence is given in Section 6.5.6 which showed that the attention-passing model is much more resistent to error propagation than the other models.

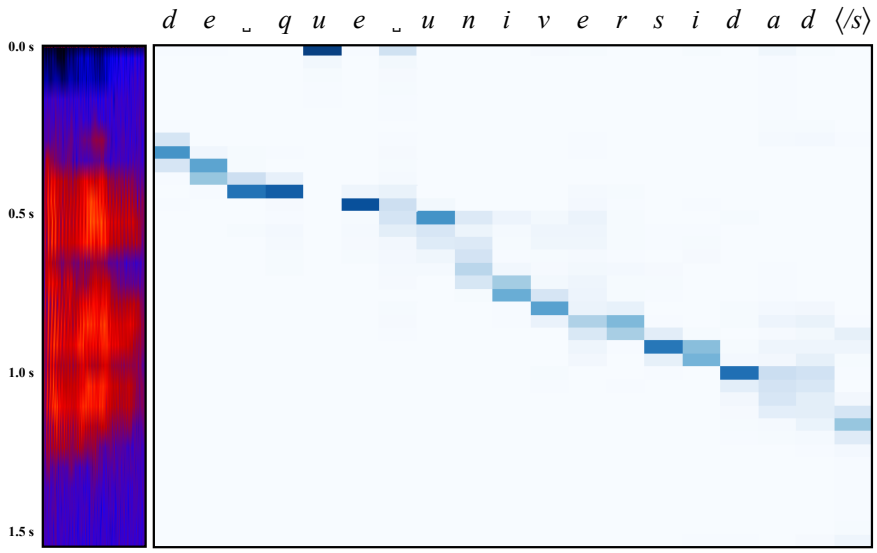


Figure 6.9: ASR attentions when force-decoding the oracle transcripts.

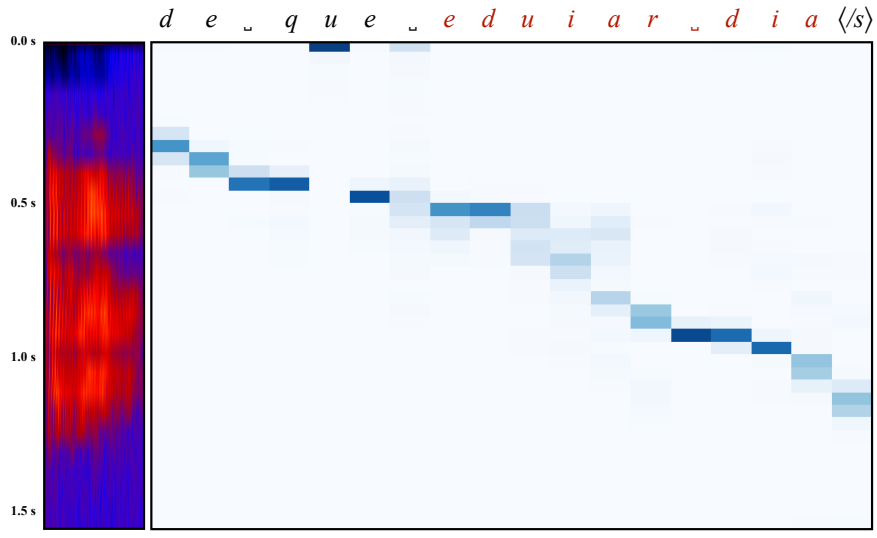


Figure 6.10: ASR attentions after regular decoding.

6.6 Related Work

Model architectures similar to what we have here referred to as the basic two-stage model have first been used by Tu et al. [TLS+17] for a reconstruction task, where the first stage performs translation and the second stage attempts to reconstruct the original inputs based on the outputs of the first stage. A second variant of a similar architecture are Xia et al. [XTW+17]'s deliberation networks, where the second stage refines or polishes the outputs of the first stage. For our purposes, the first stage performs speech recognition, a natural intermediate representation for the speech translation task, corresponding to the second stage output. Toshniwal et al. [TTLL17] explore a different way of lower-level supervision during training of an attentional speech recognizer by jointly training an auxiliary phoneme recognizer based on a lower layer in the acoustic encoder. Similarly to the discussed multi-task direct model, this approach discards many of the learned parameters when used on the main task and consequently may also suffer from data efficiency issues.

Direct end-to-end speech translation models were first used by [DAC+16], although the authors did not actually evaluate translation performance. Weiss et al. [WCJ+17] extended this model into a multi-task model and report excellent translation results. Our baselines do not match their results, despite considerable efforts. We note that other research groups have encountered similar replicability issues [BKL+18], explanations include the lack of a large GPU cluster to perform ASGD training, as well as to explore an ideal number of training schedules and other hyper-parameter settings. Bérard et al. [BBKP18] explored the translation of audio books with direct models and report reasonable results, but do not outperform a cascaded baseline. Kano et al. [KSN17] have first used a basic two-stage model for speech translation. They use a pretraining strategy for the individual sub-models, related to our multi-task approach, but do not attempt to integrate auxiliary data. Moreover, the authors only evaluated the translation of synthesized speech, which greatly simplifies training and may not lead to generalizable conclusions. Anastasopoulos et al. [AC18] conduct experiments on low-resource speech translation and employ a triangle model that can be seen as a combination of a direct model and a two-stage model, but is not easily trainable in a multi-task fashion. It is therefore not a suitable choice for exploiting auxiliary data in order to compete with cascaded models under well-resourced data conditions. Finally, contemporaneous work explores transferring knowledge from high-resource to low-resource languages [BKL+19].

6.7 Chapter Summary

This chapter explored *direct* and *two-stage* models for speech translation with the aim of obtaining models that are strong not only under favorable data conditions, but are also efficient at exploiting auxiliary data. We started by demonstrating that direct models do outperform cascaded models, but only when enough data is available, shedding light on inconclusive results from prior work. We further showed that these models are poor at exploiting auxiliary data, making them a poor choice in realistic situations. We motivated the use of two-stage models by their ability to overcome this shortcoming of the direct models, and found that two-stage models are in fact more data-efficient, but suffer from error propagation issues. We addressed this by introducing a novel attention-passing model that prevents error propagation, as well as several model variants. The best proposed model outperforms all other tested models and is much more data efficient than the direct model, allowing this model to compete with cascaded models even under realistic assumptions with auxiliary data available. Analysis showed that there seems to be a trade-off between data efficiency and error propagation.

An interesting avenue for future work is the integration of techniques from Chapter 5 into the described two-stage or attention-passing models. The first technique that addressed robustness (Section 5.1) would be relatively straightforward to integrate, by introducing substitutions, insertions, and deletions to the attention-passing model's intermediate source text decoder labels during training. Integrating the second technique, lattice translation (Section 5.2), is also conceivable, though more involved. This would involve creating a lattice representation based on the first-stage decoder's output. Creating such lattices would require heuristics, because RNN decoders strictly speaking do not allow for path recombination. The second-stage model component would then translate the complete lattice into the target text, as described in this thesis.

Besides these suggestions, further avenues for future work include testing better ASR attention models, examining the effect of adding out-of-domain ASR data, and exploring further model variants.

Chapter 7

Discussion

7.1 Overview and Comparison

We have introduced several variants of direct and cascaded models in the context of speech translation. To contrast these against one another, it is instructive to recall the modeling assumptions and decompositions taken by each. As before, let T denote the translation, X the audio signal, S the source side transcript, and $\mathcal{H}$ the hypothesis space of all possible transcripts.

We started our explorations in Chapter 4 by looking at direct models for speech recognition. In particular, encoder-decoder models allow us to model $P(S \mid X)$ directly, an important prerequisite on our quest toward end-to-end speech translation. Besides analyzing state-of-the-art recognition models, we proposed a new self-attentional architecture that successfully improves upon two major drawbacks of many direct neural modeling techniques, namely speed and interpretability.

Equipped with this model, we moved to our main task, speech translation. Chapter 5 looked at cascaded scenarios which are modeled as

$$Pr(T \mid X) = \sum_{S \in \mathcal{H}} Pr(T \mid S, X) Pr(S \mid X) \approx \sum_{S \in \mathcal{H}} p_{\text{asr}}(T \mid S) p_{\text{mt}}(S \mid X).$$

This decomposition expresses that cascaded speech translation requires estimating two distinct models, a speech recognizer $p_{\text{asr}}(S \mid X)$ and a machine translation model $p_{\text{mt}}(T \mid S)$. In this chapter we assumed an already existing model $p_{\text{asr}}(S \mid X)$, and devised effective ways for modeling $p_{\text{mt}}(T \mid S)$ as well as for approximating the computationally prohibitive sum over the ASR hypothesis space. Our first solution was

a noise-robust sequence-to-sequence model p_{robust} following the modeling assumptions

$$\sum_{S \in \mathcal{H}} Pr(T \mid S) Pr(S \mid X) \approx Pr(T \mid S') Pr(S' \mid X) \propto p_{\text{robust}}(T \mid S'),$$

where $S' = \text{argmax}_{S \in \mathcal{H}} p_{\text{asr}}(S \mid X)$ (the ASR 1-best output) replaces the exhaustive sum. This method is very simple to implement, but was outperformed by our second approach, the lattice-to-sequence model p_{lat2seq}. Here, our model followed

$$Pr(T \mid X) = \sum_{S \in \mathcal{H}} Pr(T \mid S, X) Pr(S \mid X) \approx \sum_{S \in \mathcal{L}} Pr(T \mid S) P(S \mid X) = p_{\text{lat2seq}}(T \mid \mathcal{L}).$$

The lattice $\mathcal{L}$ approximates the hypothesis space, and the sum is implicitly handled by the lattice-to-sequence model.

Up to this point, we have investigated ways to tighten the speech translation cascade using neural models, but have still trained ASR and MT models separately. In Chapter 6 we proceeded to the one-model approach by exploiting the fact that encoder-decoder models are flexible enough to model $Pr(T \mid X)$ directly. We saw that this direct modeling approach can work when suitable data is available and that auxiliary ASR and MT data can be incorporated through multi-task training that jointly trains auxiliary models with partly overlapping parameters. However, we have also discussed the inconclusive results in prior work on whether the direct approach outperforms the cascaded approach, and were able to explain this by the increased data requirements of the former. In addition, we experimentally demonstrated that the suboptimal data efficiency is a major disadvantage. In particular, direct models are poor at exploiting auxiliary data and only perform well when large amounts of end-to-end training data are available, which is rarely the case in practice.

We therefore identified two-stage models as a promising alternative. We showed empirically that two-stage models improve data efficiency, but unfortunately re-introduce error propagation into end-to-end models. We showed how to solve both issues at once by obscuring the discrete source text representation from the translation stage, and instead passing on the more robust source-text attentions. This approach combines the modeling advantages of all methods considered in the chapter.

At this point, the question of which of all the investigated approaches in this thesis should be used in practice arises. First, the noise-robust model's gains are smaller than those of the other proposed approaches, but the method is nonetheless appealing because it is extremely easy to implement, yields consistent gains across very different model settings, and does not require special training data or special ASR outputs

(such as lattices). The lattice-to-sequence model, in contrast, leads to much greater gains. In fact, these numbers are among the strongest in this thesis and approach those of our best one-model approach that uses additional MT training data. It also benefits immediately from advancements to the (HMM-based) ASR model, suggesting that this model may be preferable in cases where a strong ASR system exists. However, developing a strong HMM-based ASR model can be a major burden, and changes to the ASR model require laborious updates of the training lattices and retraining of the lattice-to-sequence model.

The one-model approach, in turn, is conceptually simple and appealing from a modeling point of view because it allows joint parameter estimation and suffers less from error propagation. Our proposed method allows overcoming its main weakness and makes it very data-efficient. It produced the strongest results among all models in this thesis by being able to exploit auxiliary translation data. The fact that the numbers were still behind the lattice-to-sequence model under equal data conditions can be explained by its somewhat inferior ASR modeling component. In the lattice-to-sequence model, a complicated and highly engineered traditional HMM model, developed by many contributors from the research community, was used [PKL+13]. In contrast, our end-to-end models used a completely self-developed ASR component, inevitably involving much less tuning and engineering. We may think of the lattice-to-sequence model as using a much larger *bag-of-tricks* than the end-to-end model. This is owed to the considerable labor and computational resources required to increase the bag-of-tricks. However, literature indicates that encoder-decoder based ASR models may be able to compete with traditional models [CSW+18]. Improvements on this front are likely to benefit our one-model approaches in an equal fashion. It is therefore reasonable to assume that the one-model approach will be able to compete with and potentially outperform cascaded approaches such as the lattice-to-sequence model in the future.

Besides the engineering effort related to the bag-of-tricks, the other decisive factor is available data. In particular, when no direct speech translation data is available, more traditional cascaded approaches remain the only viable choice. Once a certain critical amount of such data becomes available, end-to-end modeling techniques as developed in this thesis can become the method of choice.

7.2 Future Work

7.2.1 Streaming

In practical scenarios, it is necessary to distinguish the pre-segmented and the streaming scenario. In the **pre-segmented** case, followed throughout this thesis, we assume that audio inputs correspond to linguistically sensible sentence-like units which we refer to as utterances. These occur naturally in many use-cases; for example, a voice query is usually made in the form of a single utterance, and in simple dialog situations people may speak a single utterance and then insert a break to listen to a reply. In the **streaming** case, we assume a continuous stream of audio, without extrinsic information on potential segment or utterance boundaries. This occurs in situations such as public speeches and lectures, as well as more advanced dialogues where speakers may utter a series of continuous utterances before trading turns. It is relatively straight forward to turn the streaming scenario into a pre-segmented scenario by applying an audio segmenter based on detection of silence and other features. However, this may not be ideal, because silences often do not correspond to linguistically plausible utterance breaks and therefore may compromise translation quality due to missing context. In a cascade approach it is possible to re-segment the output of the speech recognizer before feeding it to the translation component, but in more tightly coupled models this is not always possible and it would therefore by desirable to find a more principled solution to this problem. Dealing with streams directly is challenging on several levels. Computationally, we can only fit a limited amount of audio in memory, and some components may suffer from super-linear runtime. From a modeling perspective, certain statistical model such as bidirectional recurrent networks require a full segment including future context. Linguistically, it is not always clear where to place a clean-cut utterance boundary, given that spoken language is often malformed and not grammatically correct.

7.2.2 Creating Data Resources

This thesis demonstrated several ways for exploiting end-to-end speech translation data, i.e. audio recordings paired with translations and potentially transcriptions. Such data is still rather scarce: The only two larger publicly available data sets are the Fisher-Callhome Spanish-English corpus [PKL+13] and the LibriSpeech English-French corpus [KBK18]. However, it is possible to exploit further data that is already available in some form, for example English TED talks [TED12] for which translations in many

languages exist, European parliamentary speeches that exist in several languages and have been translated into several European languages, and audio books by extending the approach of Kocabiyikoglu et al. [KBK18] to other languages.

In cases where no existing data resource can be found, one may resort to cost-efficient approaches to collecting data [SNF+13, SSN+14, SNNW14a, SNNW14b, SNNW16, SNN+17] and conducting evaluation [SNN+16] which attempt to minimize the amount of work that must be performed by humans.

7.2.3 Low resource

As indicated in the introduction, one of the situations where speech translation is tremendously useful is when there are users with no access to writing (e.g. for lack of education or lack of a writing system), or without standardized writing system. Unfortunately, these situations tend to involve the languages for which little or no useful data is available, posing a major challenge for data-driven approaches as described in this thesis. To overcome this, solutions on several levels have to be found and combined. This may include more efficient data collection [SNN+17], exploiting multilingual data resources, translation by non-bilingual speakers [HBR10], and devising language games [WLM16].

Chapter 8

Conclusion

In this thesis, we examined several ways to improve speech translation, a challenging but important task with many practical applications, through neural modeling techniques. We identified the problem of error propagation as a central challenge in speech translation. This challenge motivates our efforts toward end-to-end approaches and tighter integration of components up to the point where the complete speech translation pipeline is addressed by a single model. We identified neural encoder-decoder models as a promising fit for tighter integration and direct modeling.

We started our quest toward end-to-end speech translation by analyzing the state-of-the-art for encoder-decoder based speech recognition. We showed that it performs reasonably well to tackle this task, despite still lacking behind somewhat when compared to highly engineered traditional methods. We then introduced a new self-attentional acoustic model. We showed that its parallelizable computations improve computation speed significantly on modern GPU hardware. We also showed that our model improves interpretability of the acoustic model component, in part thanks to our introduced trainable Gaussian masking strategy. Self-attentional models therefore address two of the main drawbacks in common neural end-to-end models, poor computation speed and poor interpretability.

We then turned to our main task, speech translation. The first set of contributions targets the cascaded speech translation scenario which features separately trained ASR and MT components. The first and simplest method we introduced improves the robustness of the translation component when exposed to errors from the speech recognition component. This proposed data noising method aims to reduce the impact of error propagation by training the model to deal with such errors and therefore preventing error compounding and reducing the mismatch between training and test

situations. The observed gains are small, but consistently generalize even to large-scale models. The second contribution removes a limitation of the previous approach, namely that the translation model only has access to the error-prone first-best output of the speech recognizer. Instead, a rich lattice structure that efficiently represents the speech recognition hypothesis space is passed to the translation model. We show how to extend encoder-decoder models to be able to directly translate such lattices, and observe substantial gains by doing so. We identified the integration of lattice confidence scores as an important contributing factor to the success of this approach.

Our final method and the central contribution of this thesis is an end-to-end model for speech translation. Here, the task is tackled by a single model in which all parameters are trained jointly. The simplest approach, a direct encoder-decoder model, had been investigated in prior work but with inconclusive results when compared to traditional cascaded models. We provided evidence that this is explained by the increased need for suitable speech translation data for such models, while in practice such data is scarce. Moreover, we demonstrated that these models are poor at exploiting the much more abundantly available ASR and MT data. We identified *two-stage models* as a remedy. Two-stage models compute separate audio and translation alignments, introduce a stronger inductive bias to guide the training process, and are much more effective at exploiting ASR and MT data. Unfortunately, they also suffer from the problem of error propagation. We therefore introduced a novel technique to prevent such error propagation, resulting in the best accuracy among all tested models. This shows for the first time that end-to-end modeling techniques are a generalizably promising and practically relevant approach to speech translation. We expect that its main limitation, the still inferior performance of the ASR modeling component, can be removed by further engineering and tuning efforts, which we identify as important future work.

Bibliography

[AAB06] P. D. Aguero, Jordi Adell, and Antonio Bonafonte. Prosody Generation for Speech-to-Speech Translation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2006. 15

[AC18] Antonios Anastasopoulos and David Chiang. Tied Multitask Learning for Neural Speech Translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA, 2018. 76, 89, 97

[AKESH00] Raja Al-Khanji, Said El-Shiyab, and Riyadh Hussein. On the Use of Compensatory Strategies in Simultaneous Interpretation. *Meta: Journal des traducteurs*, 45(3):548, 2000. 54

[AMB+16] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Many Languages, One Parser. *Transactions of the Association for Computational Linguistics (TACL)*, 4:431–444,, 2016. 85

[AOB12] Gopala Krishna Anumanchipalli, Luis C. Oliveira, and Alan W. Black. Intent transfer in speech-to-speech machine translation. In *Workshop on Spoken Language Technology (SLT)*, pages 153–158. IEEE, 2012. 15

[Bak75] James Baker. The DRAGON system - An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975. 11

[BB18] Yonatan Belinkov and Yonatan Bisk. Synthetic and Natural Noise Both Break Neural Machine Translation. In *International Conference on Learning Representations (ICLR)*, 2018. 46, 57

[BBF+02] Alan W. Black, Ralf D. Brown, Robert Frederking, Kevin Lenzo, John Moody, Alexander Rudnicky, Rita Singh, and Eric Steinbrecher. Rapid

Development of Speech-to-Speech Translation Systems. In *International Conference on Spoken Language Processing (ICSLP)*, pages 1–4, 2002. 16

[BBKP18] Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. End-to-End Automatic Speech Translation of Audiobooks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018. 76, 77, 82, 89, 97

[BCB15] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Representation Learning (ICLR)*, San Diego, USA, 2015. 2, 3, 6, 7, 41, 46, 57, 79

[BCS+16] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philémon Brakel, and Yoshua Bengio. End-To-End Attention-Based Large Vocabulary Speech Recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. 11, 42

[BDVJ03] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. 10

[BF05] Nicola Bertoldi and Marcello Federico. A New Decoder for Spoken Language Translation Based on Confusion Networks. In *Automatic Speech Recognition & Understanding (ASRU)*, pages 86–91, 2005. 17, 72

[BJM16] Piotr Bojanowski, Armand Joulin, and Tomáš Mikolov. Alternative Structures for Character-Level RNNs. In *International Conference on Learning Representations (ICLR)*, 2016. 10

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffry E. Hinton. Layer Normalization. *arXiv:1607.06450*, 2016. 35

[BKL+18] Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. Low-Resource Speech-to-Text Translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, 2018. 97

[BKL+19] Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. Pre-training on high-resource speech recognition

improves low-resource speech-to-text translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 82, 97

[BPPM93] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993. 17

[BR01] Srinivas Bangalore and Giuseppe Riccardi. A Finite-State Approach to Machine Translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, Pittsburgh, USA, 2001. 16

[Buc01] Gary Buck. An overview of listening comprehension. In *Assessing Listening*, chapter 1. Cambridge University Press, 2001. 76

[BW91] Ulrich Bodenhausen and Alex Waibel. The Tempo 2 Algorithm: Adjusting Time-Delays By Supervised Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 155–161, 1991. 42

[CBKO06] Chris Callison-Burch, Philipp Koehn, and Miles Osborne. Improved statistical machine translation using paraphrases. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 17–24, New York City, USA, 2006. 56

[CBS+15] Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585, 2015. 3, 94

[CD87] Wallace Chafe and Jane Danielwicz. Properties of spoken and written language. Technical Report 5, National Center for the Study of Writing and Literacy, 1987. 15

[CDL16] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA, 2016. 9, 33, 41

[CFH+13] Eunah Cho, Christian Fügen, Teresa Hermann, Kevin Kilgour, Mohammed Mediani, Christian Mohr, Jan Niehues, Kay Rottmann,

Christian Saam, Sebastian Stüker, and Alex Waibel. A real-world system for simultaneous translation of German lectures. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 3473–3477, 2013. 1, 17

[CG98] Stanley F Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Association for Computational Linguistic (ACL)*, 1998. 12

[CHHL17] Pin-Jung Chen, I-Hung Hsu, Yi-Yao Huang, and Hung-Yi Lee. Mitigating the Impact of Speech Recognition Errors on Chatbot using Sequence-to-sequence Model. *arXiv:1709.07862*, 2017. 47

[CJ17] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 523–527, 2017. 27, 32

[CJLV16] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2016. 3, 11, 13, 25, 26

[CKF+16] Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. Bilingual Methods for Adaptive Training Data Selection for Machine Translation. In *Association for the Machine Translation in Americas (AMTA)*, 2016. 46, 56

[CMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*, 2014. 57

[CMW04] Christopher Cieri, David Miller, and Kevin Walker. The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text. In *Language Resources and Evaluation (LREC)*, pages 69–71, 2004. 79

[CMX11] Constantine Caramanis, Shie Mannor, and Huan Xu. Robust Optimization in Machine Learning. In *Optimization for Machine Learning*. The MIT Press, 2011. 46

[CNO+04] Francisco Casacuberta, Hermann Ney, Franz Josef Och, Enrique Vidal, J. M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C. Martínez, S. Molau, F. Nevado, M. Pastor, D. Picó, A. Sanchis, and C. Tillmann. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18(1):25–47, 2004. 16, 57, 72

[CNW14] Eunah Cho, Jan Niehues, and Alex Waibel. Tight Integration of Speech Disfluency Removal into SMT. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 1689–1699, Gothenburg, Sweden, 2014. 72

[CNW17] Eunah Cho, Jan Niehues, and Alex Waibel. NMT-based Segmentation and Punctuation Insertion for Real-time Spoken Language Translation. *Proc. Interspeech 2017*, pages 2645–2649, 2017. 3

[Cos71] J. Costello. On the number of points in regular discrete simplex. *IEEE Transactions on Information Theory*, 17(2):211–212, 1971. 48

[CSW+18] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Katya Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art Speech Recognition With Sequence-to-Sequence Models. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018. 32, 101

[CTM+18] Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. Towards Robust Neural Machine Translation. *Arxiv*, 2018. 57

[DAC+16] Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. An Attentional Model for Speech Translation Without Transcription. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 949–959, San Diego, USA, 2016. 76, 79, 97

[DMR08] Christopher Dyer, Smaranda Muresan, and Philip Resnik. Generalizing Word Lattice Translation. Technical Report LAMP-TR-149, University of Maryland, Institute For Advanced Computer Studies, 2008. 57, 59, 72

[DN17] Michael Denkowski and Graham Neubig. Stronger Baselines for Trustable Results in Neural Machine Translation. In *The First Workshop on Neural Machine Translation*, Vancouver, Canada, 2017. 51

[DSN18] Truong Quoc Do, Sakriani Sakti, and Satoshi Nakamura. Sequence-to-Sequence Models for Emphasis Speech Translation. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(10), 2018. 15

[EHT16] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-Sequence Attentional Neural Machine Translation. In *Association for Computational Linguistic (ACL)*, pages 823–833, Berlin, Germany, 2016. 60

[Füg08] Christian Fügen. *A System for Simultaneous Translation of Lectures and Speeches*. PhD thesis, University of Karlsruhe, 2008. 17

[GAAD+18] Pierre Godard, Gilles Adda, Martine Adda-Decker, J. Benjumea, L. Besacier, J. Cooper-Leavitt, G-N. Kouarata, L. Lamel, H. Maynard, M. Mueller, A. Rialland, S. Stueker, F. Yvon, and M. Zanon-Boito. A Very Low Resource Language Speech Corpus for Computational Language Documentation Experiments. In *Language Resources and Evaluation (LREC)*, 2018. 18

[GAG+17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning. In *International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017. 9, 36

[GG16] Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Neural Information Processing Systems Conference (NIPS)*, pages 1019–1027, Barcelona, Spain, 2016. 26, 27, 38, 51, 89

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6645–6649, Vancouver, Canada, 2013. IEEE. 11, 13

[HB78] Robert L. Hemminger and Lowell W. Beineke. Line graphs and line digraphs. In *Selected Topics in Graph Theory*, pages 271–305. Academic Press Inc., 1978. 61

[HBR10] Chang Hu, Benjamin B Bederson, and Philip Resnik. Translation by iterative collaboration between monolingual users. In *Graphics Interface*, pages 39–46. Canadian Information Processing Society, 2010. 103

[HDA11] Xiaodong He, Li Deng, and Alex Acero. Why word error rate is not a good metric for speech recognizer training for the speech translation task? *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5632–5635, 2011. 17

[HNvG17] Georg Heigold, Günter Neumann, and Josef van Genabith. How Robust Are Character-Based Word Embeddings in Tagging and MT Against Wrod Scramlbing or Randdm Nouse? *arXiv:1704.04441*, 2017. 46, 56

[Hoc98] Sepp Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998. 8

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997. 7, 8

[Hun87] Melvyn J Hunt. Delayed decisions in speech recognition - the case of formants. *Pattern Recognition Letters*, 6(2):121–137, 1987. 2

[IC17] Jinbae Im and Sungzoon Cho. Distance-based Self-Attention Network for Natural Language Inference. *arXiv:1712.02047*, 2017. 41

[IKS+13] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa. Reverberant speech recognition based on denoising autoencoder. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 3512–3516, Lyon, France, 2013. 56

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015. 27

BIBLIOGRAPHY

[JBM75] Frederick Jelinek, Lalit R. Bahl, and Robert L. Mercer. Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975. 11

[JC92] Marcel Adam Just and Patricia A Carpenter. A Capacity Theory of Comprehension: Individual Differences in Working Memory. *Psychological Review*, 99(January):122–149, 1992. 76

[JH13] Navdeep Jaitly and Geoffrey Hinton. Vocal tract length perturbation (VTLP) improves speech recognition. In *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech, and Language Processing*, Atlanta, USA, 2013. 56

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning (ICML)*, Lille, France, 2015. 40, 65

[KB13] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709, Seattle, Washington, USA, 2013. 2, 6, 46, 57

[KB14] Diederik P. Kingma and Jimmy L. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014. 26, 65, 89

[KBK18] Ali Can Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. Augmenting Librispeech with French Translations: A Multimodal Corpus for Direct Speech Translation Evaluation. In *Language Resources and Evaluation (LREC)*, Miyazaki, Japan, 2018. 18, 79, 102, 103

[KBT+15] Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. A Coarse-Grained Model for Optimal Coupling of ASR and SMT Systems for Speech Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1902–1907, Lisbon, Portugal, 2015. 17

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. In *Association for Computational Linguistics (ACL)*, pages 655–665, 2014. 9

[Kim14] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 9

[KK17] Philipp Koehn and Rebecca Knowles. Six Challenges for Neural Machine Translation. *arXiv:1706.03872*, 2017. 56

[Koe17] Philipp Koehn. Neural Machine Translation. *arXiv:1709.07809*, 2017. 46, 56

[KOM03] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical Phrase-based Translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 48–54, Edmonton, USA, 2003. 17

[KSN17] Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. Structured-based Curriculum Learning for End-to-end English-Japanese Speech Translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2630–2634, 2017. 76, 78, 83, 89, 97

[KTS+13] Takatomo Kano, Shinnosuke Takamichi, Sakriani Sakti, Graham Neubig, Tomoki Toda, and Satoshi Nakamura. Generalizing continuous-space translation of paralinguistic information. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2614–2618, 2013. 15

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989. 9

[LFdS+17] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A Structured Self-attentive Sentence Embedding. In *International Conference on Representation Learning (ICLR)*, Toulon, France, 2017. 9, 33, 41

[LGD+16] Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. LatticeRnn: Recurrent Neural Networks over Lattices. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 695–699, San Francisco, USA, 2016. 58, 60, 72

BIBLIOGRAPHY

[LGG+95] Alon Lavie, Donna Gates, Marsal Gavalda, Laura Mayfield, Alex Waibel, and Lori Levin. Multi-lingual Translation of Spontaneously Spoken Language in a Limited Domain. In *Conference of the Association for Machine Translation in the Americas*, pages 252–255, 1995. 15, 16

[LGGP03] Fu-hua Liu, Liang Gu, Yuqing Gao, and Michael Picheny. Use of Statistical N-Gram Models in Natural Language Generation for Machine Translation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 636–639, 2003. 15, 17

[LLL07] Donghyeon Lee, Jonghoon Lee, and Gary Geunbae Lee. POSSLT: a Korean to English spoken language translation system. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 7–8. Association for Computational Linguistics, 2007. 16

[LMJ17] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding Neural Networks through Representation Erasure. *arXiv:1612.08220*, 2017. 56

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal, 2015. 7, 26

[MD14] James L Morgan and Katherine Demuth. *Signal to syntax: Bootstrapping from speech to grammar in early acquisition.* Psychology Press, 2014. 2

[MHGK14] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*, 27:2204–2212, 2014. 41

[MHN08] Evgeny Matusov, Björn Hoffmeister, and Hermann Ney. ASR word lattice translation with exhaustive reordering is possible. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2342–2345, Brisbane, Australia, 2008. 17, 57, 72

[MMN06] Evgeny Matusov, Arne Mauser, and Hermann Ney. Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 158–165, Kyoto, Japan, 2006. 17

[MN18] Paul Michel and Graham Neubig. MTNT: A Testbed for Machine Translation of Noisy Text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 57

[MNS05] Evgeny Matusov, Hermann Ney, and Ralph Schluter. Phrase-based translation of speech recognizer word lattices using loglinear model combination. *Automatic Speech Recognition and Understanding (ASRU)*, pages 110–115, 2005. 17

[Moo52] P. G. Moore. The Estimation of the Poisson Parameter from a Truncated Distribution. *Biometrika*, 39(3/4):247–251, 1952. 48

[MSD+12] Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword Language Modeling with Neural Networks. Technical report, Unpublished Manuscript, 2012. 10

[NBC+16] Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. Reward Augmented Maximum Likelihood for Neural Structured Prediction. In *Neural Information Processing Systems Conference (NIPS)*, pages 1723–1731, Barcelona, Spain, 2016. 47

[NC17] Jan Niehues and Eunah Cho. Exploiting Linguistic Resources for Neural Machine Translation Using Multi-task Learning. In *Conference on Machine Translation (WMT)*, pages 80–89, Copenhagen, Denmark, 2017. 82

[NC18] Toan Q. Nguyen and David Chiang. Improving Lexical Choice in Neural Machine Translation. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA, 2018. 27, 38, 89

[NDG+17] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*, 2017. 26, 29, 65

BIBLIOGRAPHY

[Ney99] Hermann Ney. Speech Translation: Coupling of Recognition and Translation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 517–520, Phoenix, USA, 1999. 16, 57, 72

[NGD17] Graham Neubig, Yoav Goldberg, and Chris Dyer. On-the-fly Operation Batching in Dynamic Computation Graphs. In *Neural Information Processing Systems Conference (NIPS)*, Long Beach, USA, 2017. 64

[NPH+18] Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. Low-Latency Neural Speech Translation. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, 2018. 17

[ONS+14] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Optimizing Segmentation Strategies for Simultaneous Speech Translation. In *Association for Computational Linguistics (ACL)*, pages 551–556, Baltimore, USA, 2014. 17

[Ost99] Mari Ostendorf. Moving beyond the 'beads-on-a-string'model of speech. In *Automatic Speech Recognition & Understanding (ASRU)*, pages 79–84, 1999. 32

[Pau10] Matthias Paulik. *Learning Speech Translation from Interpretation.* PhD thesis, University of Karlsruhe, 2010. 18

[PB92] Douglas B. Paul and Janet M. Baker. The design for the Wall Street Journal-based CSR corpus. In *Workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992. 19

[PCPK15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an ASR corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, Brisbane, Australia, 2015. 79

[PGB+11] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, and Others. The Kaldi speech recognition toolkit. In *Workshop on Automatic Speech Recognition & Understanding (ASRU)*, 2011. 30

[PGJ+07] Alicia Pérez, Victor Guijarrubia, Raquel Justo, M. Inés Torres, and Francisco Casacuberta. A comparison of linguistically and statistically enhanced models for speech-to-speech machine translation. In *International Workshop on Spoken Language Translation (IWSLT) 2007*, 2007. 16

[PHG+18] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. A Time-Restricted Self-Attention Layer for ASR. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018. 42

[PKL+13] Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg, Germany, 2013. 18, 21, 46, 59, 65, 79, 88, 101, 102

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Association for Computational Linguistic (ACL)*, pages 311–318, Philadephia, Pennsylvania, 2002. 23

[PSS+17] Ngoc-quan Pham, Matthias Sperber, Elizabeth Salesky, Thanh-le Ha, Jan Niehues, and Alex Waibel. KIT's Multilingual Neural Machine Translation Systems for IWSLT 2017. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 42–47, 2017. 4, 55

[PTDU16] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2249–2255, Austin, USA, 2016. 9, 33, 41

[QFC05] Vu Hai Quan, Marcello Federico, and Mauro Cettolo. Integrated N-best Re-ranking for Spoken Language Translation. In *Interspeech*, pages 3181–3184, 2005. 16

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989. 62

BIBLIOGRAPHY

[RDBF17] Nicholas Ruiz, Mattia Antonino Di Gangi, Nicola Bertoldi, and Marcello Federico. Assessing the Tolerance of Neural Machine Translation Systems Against Speech Recognition Errors. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2635–2639, Stockholm, Sweden, 2017. 18, 46

[RDE14] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. In *International Conference on Language Resources and Evaluation (LREC)*, pages 3935–3939, 2014. 20, 30

[RDJ05] Jennifer M. Rodd, Matthew H. Davis, and Ingrid S. Johnsrude. The Neural Mechanisms of Speech Comprehension: fMRI Studies of Semantic Ambiguity. *Cerebral Cortex*, 15(8):1261–1269, 2005. 76

[RGLF15] Nicholas Ruiz, Qin Gao, William Lewis, and Marcello Federico. Adapting Machine Translation Models toward Misrecognized Speech with Text-to-Speech Pronunciation Rules and Acoustic Confusability. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2247–2251, Dresden, Germany, 2015. 18, 47, 50, 56

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: PTR Prentice Hall, 1993. 11

[Sel78] Danica Seleskovitch. *Interpreting for International Conferences: Problems of Language and Communication*. Pen & Booth, Washington D.C., 1978. 76

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. 12

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Association for Computational Linguistic (ACL)*, 2016. 10

[SHG+14] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Conference on Information and Knowledge Management (CIKM)*, pages 101–110, 2014. 9

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 46

[SJVS04] Shirin Saleem, Szu-Chen Jou, Stephan Vogel, and Tanja Schultz. Using Word Lattice Information for a Tighter Coupling in Speech Translation Systems. In *International Conference on Spoken Language Processing (ICSLP)*, pages 41–44, Jeju Island, Korea, 2004. 17, 57, 72

[SKM+12] Sebastian Stüker, Florian Kraft, Christian Mohr, Teresa Herrmann, Eunah Cho, and Alex Waibel. The KIT Lecture Corpus for Speech Translation. In *Language Resources and Evaluation (LREC)*, pages 3409–3414, Istanbul, Turkey, 2012. 18

[SMFW01] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 214–217, Madonna di Campiglio, Italy, 2001. 30

[SNF+13] Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel. Efficient Speech Transcription Through Respeaking. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 1087–1091, Lyon, France, 2013. 103

[SNN+16] Matthias Sperber, Graham Neubig, Jan Niehues, Sebastian Stüker, and Alex Waibel. Lightly Supervised Quality Estimation. In *International Conference on Computational Linguistics (COLING)*, pages 3103–3113, 2016. 103

[SNN+17] Matthias Sperber, Graham Neubig, Jan Niehues, Satoshi Nakamura, and Alex Waibel. Transcribing Against Time. *Speech Communication*, 93C:20–30, 2017. 103

[SNN+18] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. Self-Attentional Acoustic Models. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Hyderabad, India, 2018. 4

BIBLIOGRAPHY

[SNNW14a] Matthias Sperber, Graham Neubig, Satoshi Nakamura, and Alex Waibel. On-the-Fly User Modeling for Cost-Sensitive Correction of Speech Transcripts. In *Spoken Language Technology Workshop (SLT)*, pages 460–465, Lake Tahoe, USA, 2014. 103

[SNNW14b] Matthias Sperber, Graham Neubig, Satoshi Nakamura, and Alex Waibel. SESLA Transcriber: A Speech Transcription Tool That Adapts To Your Skill And Time Budget. In *Spoken Language Technology Workshop (SLT)*, Lake Tahoe, USA, 2014. 103

[SNNW16] Matthias Sperber, Graham Neubig, Satoshi Nakamura, and Alex Waibel. Optimizing Computer-Assisted Transcription Quality with Iterative User Interfaces. In *Language Resources and Evaluation (LREC)*, Portorož, Slovenia, 2016. 103

[SNNW17] Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. Neural Lattice-to-Sequence Models for Uncertain Inputs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1380–1389, Copenhagen, Denmark, 2017. 4

[SNNW19] Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation. *Transactions of the Association for Computational Linguistics (TACL)*, 2019. 4

[SNW17] Matthias Sperber, Jan Niehues, and Alex Waibel. Toward Robust Neural Machine Translation for Noisy Input Sequences. In *International Workshop on Spoken Language Translation (IWSLT)*, Tokyo, Japan, 2017. 4

[SPN+18] Matthias Sperber, Ngoc Quan Pham, Thai Son Nguyen, Jan Niehues, Markus Müller, Thanh-Le Ha, Sebastian Stüker, and Alex Waibel. KIT's IWSLT 2018 SLT Translation System. In *International Workshop on Spoken Language Translation (IWSLT)*, 2018. 4, 55

[SSB15] Hasim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, 2015. 32

[SSN07] Eiichiro Sumita, Tohru Shimizu, and Satoshi Nakamura. NICT-ATR speech-to-speech translation system. In *Association for Computational Linguistic (ACL)*, pages 25–28. Association for Computational Linguistics, 2007. 16

[SSN+14] Matthias Sperber, Mirjam Simantzik, Graham Neubig, Satoshi Nakamura, and Alex Waibel. Segmentation for Efficient Supervised Language Annotation with an Explicit Cost-Utility Tradeoff. *Transactions of the Association for Computational Linguistics (TACL)*, 2(April):169–180, 2014. 103

[SSRB17] Hasim Sak, Matt Shannon, Kanishka Rao, and Francoise Beaufays. Recurrent Neural Aligner : An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 1298–1302, 2017. 11

[ST04] Noah Smith and Roy Tromble. Sampling uniformly from the unit simplex. Technical report, Johns Hopkins University, 2004. 48, 50

[STX+17] Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation. In *Conference on Artificial Intelligence (AAAI)*, pages 3302–3308, San Francisco, USA, 2017. 57, 58, 59, 60, 66, 72

[SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 464–468, New Orleans, USA, 2018. 42

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, USA, 2016. 27, 38, 89

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, Montréal, Canada, 2014. 2, 6, 46, 57

[SZL+18] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding. In *Conference on Artificial Intelligence (AAAI)*, New Orleans, USA, 2018. 41

[TED12] TED. Talks on technology, entertainment, and design. http://www.ted.com, 2012. 17, 20, 102

[TGN13] Andreas Tsiartas, Panayiotis G Georgiou, and Shrikanth Narayanan. Toward transfer of acoustic cues of emphasis across languages. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 3483–3486, 2013. 15

[Tie09] Jörg Tiedemann. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. *Recent Advances in Natural Language Processing*, V:237–248, 2009. 92

[TLS+17] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. Neural Machine Translation with Reconstruction. In *Conference on Artificial Intelligence (AAAI)*, 2017. 78, 80, 89, 97

[TMD14] Yulia Tsvetkov, Florian Metze, and Chris Dyer. Augmenting translation models with simulated acoustic confusions for improved spoken language translation. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 616–625, Gothenburg, Sweden, 2014. 18, 47, 50, 56

[TMS+98] Toshiyuki Takezawa, Tsuyoshi Morimoto, Yoshinori Sagisaka, Nick Campbell, Hitoshi Iida, Fumiaki Sugaya, Akio Yokoo, and Seiichi Yamamoto. A Japanese-to-English Speech Translation System: ATR-MATRIX. In *International Conference on Spoken Language (ICSLP)*, 1998. 16

[TSM15] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Association for Computational Linguistic (ACL)*, pages 1556–1566, Beijing, China, 2015. 58, 59, 60

[TSN17] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Local Monotonic Attention Mechanism for End-to-End Speech Recognition.

In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 431–440, 2017. 32, 42, 94

[TSW+17] Shubham Toshniwal, Tara N. Sainath, Ron J. Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. Multilingual Speech Recognition With A Single End-To-End Model. *arXiv*, 2017. 32

[TTLL17] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm, Sweden, 2017. 32, 97

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018. 42

[Vid97] Enrique Vidal. Finite-State Speech-to-Speech Translation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 111–114, Munich, Germany, 1997. 16

[VKK+15] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a Foreign Language. In *Neural Information Processing Systems Conference (NIPS)*, Montréal, Canada, 2015. 3

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Neural Information Processing Systems Conference (NIPS)*, pages 5998–6008, Long Beach, USA, 2017. 9, 10, 33, 34, 36, 41

[WBW07] Alex Waibel, Keni Bernardin, and Matthias Wölfel. Computer-Supported Human-Human Multilingual Communication. *50 years of artificial intelligence*, pages 271–287, 2007. 1

[WCE+93] M Woszczyna, N Coccaro, A Eisele, A Lavie, A McNair, T Polzin, I Rogina, C P Rose, T Sloboda, M Tomita, J Tsutsumi, N Aoki-Waibel, A Waibel, and W Ward. Recent advances in {JANUS}: A Speech Translation System. In *Workshop on Human Language Technology (HLT)*, pages 211–216, 1993. 15

BIBLIOGRAPHY

[WCJ+17] Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Stockholm, Sweden, 2017. 76, 77, 82, 89, 97

[WHH+87] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. Technical Report October 30, ATR Interpreting Telephony Research Laboratories, 1987. 9

[WHH+89] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989. 9

[WHK+18] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPnet: End-to-End Speech Processing Toolkit. *arXiv:1804.00015*, 2018. 29, 30

[WJM+91] Alex Waibel, Ajay N. Jain, Arthur E. McNair, Hiroaki Saito, Alexander G. Hauptmann, and Joe Tebelskis. JANUS: a speech-to-speech translation system using connectionist and symbolic processing strategies. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 793–796, Toronto, Canada, 1991. 15

[WLM16] Sida I. Wang, Percy Liang, and Christopher D. Manning. Learning Language Games through Interaction. *arXiv preprint arXiv:1606.02447*, 2016. 103

[WW91] Ye-yi Wang and Alex Waibel. A Connectionist Model for Dialog Processing. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toronto, Canada, 1991. 15

[WW98] Ye-Yi Wang and Alex Waibel. Modeling with structures in statistical machine translation. In *Association for Computational Linguistics and International Conference on Computational Linguistics (COLING-ACL)*, volume 2, pages 1357–1363, Montréal, Canada, 1998. 16

[XTW+17] Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Deliberation Networks: Sequence Generation. In *Neural Information Processing Systems Conference (NIPS)*, Long Beach, USA, 2017. 97

[XWL+17] Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. Data Noising as Smoothing in Neural Network Language Models. In *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017. 52, 56

[YBD+17] Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. The Neural Noisy Channel. In *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017. 46

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017. 56

[ZCJ17] Yu Zhang, William Chan, and Navdeep Jaitly. Very Deep Convolutional Networks for End-to-End Speech Recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. 27

[ZKYL05] Ruiqiang Zhang, Genichiro Kikui, Hirofumi Yamamoto, and Wai-Kit Lo. A Decoding Algorithm for Word Lattice Translation in Speech Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 23–29, Pittsburgh, USA, 2005. 17, 57, 72

[ZSN+18] Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller, Ngoc-quan Pham, Sebastian Stüker, and Alex Waibel. Open Source Toolkit for Speech to Text Translation. *The Prague Bulletin of Mathematical Linguistics*, 111(October):125–135, 2018. 4, 29